Atmic Networks

CERTIFICATION EXAM SUPPORT

EXAM PREPARATION MATERIAL

Practice files designed to help you understand exam patterns, question styles and the overall assessment format.

IMPORTANT NOTE

> **Please note that these files may not be up to date. However, the questions will help you understand the exam format and typical question patterns.**

Stay connected with our support team
for the latest file updates and exam-related information.

www.atmicnetworks.com

Knowledge that moves beyond memorization.

www.atmicnetworks.com

Question: 1

As the lead ML Engineer for your company, you are responsible for building ML models to digitize scanned customer forms. You have developed a TensorFlow model that converts the scanned images into text and stores them in Cloud Storage. You need to use your ML model on the aggregated data collected at the end of each day with minimal manual intervention. What should you do?

A. Use the batch prediction functionality of AI Platform
B. Create a serving pipeline in Compute Engine for prediction
C. Use Cloud Functions for prediction each time a new data point is ingested
D. Deploy the model on AI Platform and create a version of it for online inference.

Answer: A

Explanation:
Batch prediction is the process of using an ML model to make predictions on a large set of data points. Batch prediction is suitable for scenarios where the predictions are not time-sensitive and can be done in batches, such as digitizing scanned customer forms at the end of each day. Batch prediction can also handle large volumes of data and scale up or down the resources as needed. AI Platform provides a batch prediction service that allows users to submit a job with their TensorFlow model and input data stored in Cloud Storage, and receive the output predictions in Cloud Storage as well. This service requires minimal manual intervention and can be automated with Cloud Scheduler or Cloud Functions. Therefore, using the batch prediction functionality of AI Platform is the best option for this use case.
Reference:
Batch prediction overview
Using batch prediction

Question: 2

You work for a global footwear retailer and need to predict when an item will be out of stock based on historical inventory dat
a. Customer behavior is highly dynamic since footwear demand is influenced by many different factors. You want to serve models that are trained on all available data, but track your performance on specific subsets of data before pushing to production. What is the most streamlined and reliable way to perform this validation?
A. Use the TFX ModelValidator tools to specify performance metrics for production readiness

B. Use k-fold cross-validation as a validation strategy to ensure that your model is ready for production.
C. Use the last relevant week of data as a validation set to ensure that your model is performing accurately on current data
D. Use the entire dataset and treat the area under the receiver operating characteristics curve (AUC ROC) as the main metric.

Answer: A

Explanation:
TFX ModelValidator is a tool that allows you to compare new models against a baseline model and evaluate their performance on different metrics and data slices1. You can use this tool to validate your models before deploying them to production and ensure that they meet your expectations and requirements.
k-fold cross-validation is a technique that splits the data into k subsets and trains the model on k-1 subsets while testing it

on the remaining subset. This is repeated k times and the average performance is reported2. This technique is useful for estimating the generalization error of a model, but it does not account for the dynamic nature of customer behavior or the potential changes in data distribution over time.

Using the last relevant week of data as a validation set is a simple way to check the model's performance on recent data, but it may not be representative of the entire data or capture the longterm trends and patterns. It also does not allow you to compare the model with a baseline or evaluate it on different data slices.

Using the entire dataset and treating the AUC ROC as the main metric is not a good practice because it does not leave any data for validation or testing. It also assumes that the AUC ROC is the only metric that matters, which may not be true for your business problem. You may want to consider other metrics such as precision, recall, or revenue.

Question: 3

You work on a growing team of more than 50 data scientists who all use AI Platform. You are designing a strategy to organize your jobs, models, and versions in a clean and scalable way. Which strategy should you choose?

A. Set up restrictive I AM permissions on the AI Platform notebooks so that only a single user or group can access a given instance.

B. Separate each data scientist's work into a different project to ensure that the jobs, models, and versions created by each data scientist are accessible only to that user.

C. Use labels to organize resources into descriptive categories. Apply a label to each created resource so that users can filter the results by label when viewing or monitoring the resources

D. Set up a BigQuery sink for Cloud Logging logs that is appropriately filtered to capture information about AI Platform resource usage In BigQuery create a SQL view that maps users to the resources they are using.

Answer: C

Explanation:

Labels are key-value pairs that can be attached to any AI Platform resource, such as jobs, models, versions, or endpoints1. Labels can help you organize your resources into descriptive categories, such as project, team, environment, or purpose. You can use labels to filter the results when you list or monitor your resources, or to group them for billing or quota purposes2. Using labels is a simple and scalable way to manage your AI Platform resources without creating unnecessary complexity or overhead. Therefore, using labels to organize resources is the best strategy for this use case.

Reference:

Using labels

Filtering and grouping by labels

Question: 4

During batch training of a neural network, you notice that there is an oscillation in the loss. How should you adjust your model to ensure that it converges?

A. Increase the size of the training batch

B. Decrease the size of the training batch

C. Increase the learning rate hyperparameter

D. Decrease the learning rate hyperparameter

Answer: D

Explanation:
Oscillation in the loss during batch training of a neural network means that the model is overshooting the optimal point of the loss function and bouncing back and forth. This can prevent the model from converging to the minimum loss value. One of the main reasons for this phenomenon is that the learning rate hyperparameter, which controls the size of the steps that the model takes along the gradient, is too high. Therefore, decreasing the learning rate hyperparameter can help the model take smaller and more precise steps and avoid oscillation. This is a common technique to improve the stability and performance of neural network training12.

Reference:
Interpreting Loss Curves
Is learning rate the only reason for training loss oscillation after few epochs?

Question: 5

You are building a linear model with over 100 input features, all with values between -1 and 1. You suspect that many features are non-informative. You want to remove the non-informative features from your model while keeping the informative ones in their original form. Which technique should you use?

A. Use Principal Component Analysis to eliminate the least informative features.
B. Use L1 regularization to reduce the coefficients of uninformative features to 0.
C. After building your model, use Shapley values to determine which features are the most informative.
D. Use an iterative dropout technique to identify which features do not degrade the model when removed.

Answer: B

Explanation:
E. regularization, also known as Lasso regularization, adds the sum of the absolute values of the model's coefficients to the loss function1. It encourages sparsity in the model by shrinking some coefficients to precisely zero2. This way, L1 regularization can perform feature selection and remove the non-informative features from the model while keeping the informative ones in their original form. Therefore, using L1 regularization is the best technique for this use case.

Reference:
Regularization in Machine Learning - GeeksforGeeks
Regularization in Machine Learning (with Code Examples) - Dataquest
F. And L2 Regularization Explained & Practical How To Examples
G. and L2 as Regularization for a Linear Model

Question: 6

Your team has been tasked with creating an ML solution in Google Cloud to classify support requests for one of your platforms. You analyzed the requirements and decided to use TensorFlow to build the classifier so that you have full control of the model's code, serving, and deployment. You will use Kubeflow pipelines for the ML platform. To save time, you want to build on existing resources and use managed services instead of building a completely new model. How should you build the classifier?

A. Use the Natural Language API to classify support requests

B. Use AutoML Natural Language to build the support requests classifier

C. Use an established text classification model on AI Platform to perform transfer learning

D. Use an established text classification model on AI Platform as-is to classify support requests

Answer: C

Explanation:

Transfer learning is a technique that leverages the knowledge and weights of a pre-trained model and adapts them to a new task or domain1. Transfer learning can save time and resources by avoiding training a model from scratch, and can also improve the performance and generalization of the model by using a larger and more diverse dataset2. AI Platform provides several established text classification models that can be used for transfer learning, such as BERT, ALBERT, or XLNet3. These models are based on state-of-the-art natural language processing techniques and can handle various text classification tasks, such as sentiment analysis, topic classification, or spam detection4. By using one of these models on AI Platform, you can customize the model's code, serving, and deployment, and use Kubeflow pipelines for the ML platform. Therefore, using an established text classification model on AI Platform to perform transfer learning is the best option for this use case.

Reference:

Transfer Learning - Machine Learning's Next Frontier

A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning

Text classification models

Text Classification with Pre-trained Models in TensorFlow

Question: 7

Your team is working on an NLP research project to predict political affiliation of authors based on articles they have written. You have a large training dataset that is structured like this:

```
AuthorA:Political Party A
    TextAl: [SentenceAll, SentenceA12, SentenceA13, ...]
    TextA2: [SentenceA21, SentenceA22, SentenceA23, ...]

AuthorB:Political Party B
    TextBl: [SentenceBll, SentenceB12, SentenceB13, ...]
    TextB2: [SentenceB21, SentenceB22, SentenceB23, ...]

AuthorC:Political Party B
    TextCl: [SentenceCll, SentenceC12, SentenceC13, ...]
    TextC2: [SentenceC21, SentenceC22, SentenceC23, ...J

AuthorD:Political Party A
    TextDl: [SentenceDll, SentenceD12, SentenceD13, ...]
    TextD2: [SentenceD21, SentenceD22, SentenceD23, ...]
```

You followed the standard 80%-10%-10% data distribution across the training, testing, and evaluation subsets. How should you distribute the training examples across the train-test-eval subsets while maintaining the 80-10-10 proportion?

A)

Distribute texts randomly across the tram-test-eval subsets

Train set [TextAl TextB2...]

Test set (TextA2, TextCl, TextD2]

Eval set [TextBl, TextC2 TextDl,]

B)

Distribute authors randomly across the train-test-eval subsets (*)

Train set [TextAI, TextA2, TextDI, TextD2,..]

Test set [TextBI, TextB2 ..]

Eval set [TextCI, TextC2, |

C)

Distribute sentences randomly across the train-test-eval subsets

Train set [SentenceAU, SentenceA21, Sentence B11, SentenceB21, SentenceCH. SentenceD21,...] Test set [SentenceAl2, SentenceA22 Sentence B12 SentenceC22, Sentenced2 SentenceD22,] Eval set [SentenceA13, SentenceA23, Sentence 813, SentenceC23, SentenceC13, SentenceD31, |

D)

Distribute paragraphs of texts (i e, chunks of consecutive sentences) across the train-test-eval subsets

Tram set [SentenceAU, SentenceA12, Sentence D11, SentenceD12,.]

Test set [SentenceAl 3, SentenceB13 Sentence B21, SentenceD23, SentenceC12, SentenceD13,] Eval set [SentenceAU, SentenceA22 Sentence B13, SentenceD22 SentenceC23, SentenceDU]

A. Option A
B. Option B
C. Option C
D. Option D

Answer: C

Explanation:

The best way to distribute the training examples across the train-test-eval subsets while maintaining the 80-10-10 proportion is to use option C. This option ensures that each subset contains a balanced and representative sample of the different classes (Democrat and Republican) and the different authors. This way, the model can learn from a diverse and comprehensive set of articles and avoid overfitting or underfitting. Option C also avoids the problem of data leakage, which occurs when the same author appears in more than one subset, potentially biasing the model and inflating its performance. Therefore, option C is the most suitable technique for this use case.

Question: 8

Your data science team needs to rapidly experiment with various features, model architectures, and hyperparameters. They need to track the accuracy metrics for various experiments and use an API to query the metrics over time. What

should they use to track and report their experiments while minimizing manual effort?

A. Use Kubeflow Pipelines to execute the experiments Export the metrics file, and query the results using the Kubeflow Pipelines API.

B. Use AI Platform Training to execute the experiments Write the accuracy metrics to BigQuery, and query the results using the BigQueryAPI.

C. Use AI Platform Training to execute the experiments Write the accuracy metrics to Cloud Monitoring, and query the results using the Monitoring API.

D. Use AI Platform Notebooks to execute the experiments. Collect the results in a shared Google Sheets file, and query the results using the Google Sheets API

Answer: C

Explanation:

AI Platform Training is a service that allows you to run your machine learning experiments on Google Cloud using various features, model architectures, and hyperparameters. You can use AI Platform Training to scale up your experiments, leverage distributed training, and access specialized hardware such as GPUs and TPUs1. Cloud Monitoring is a service that collects and analyzes metrics, logs, and traces from Google Cloud, AWS, and other sources. You can use Cloud Monitoring to create dashboards, alerts, and reports based on your data2. The Monitoring API is an interface that allows you to programmatically access and manipulate your monitoring data3.

By using AI Platform Training and Cloud Monitoring, you can track and report your experiments while minimizing manual effort. You can write the accuracy metrics from your experiments to Cloud Monitoring using the AI Platform Training Python package4. You can then query the results using the Monitoring API and compare the performance of different experiments. You can also visualize the metrics in the Cloud Console or create custom dashboards and alerts5. Therefore, using AI Platform Training and Cloud Monitoring is the best option for this use case.

Reference:

AI Platform Training documentation

Cloud Monitoring documentation

Monitoring API overview

Using Cloud Monitoring with AI Platform Training

Viewing evaluation metrics

Question: 9

You are an ML engineer at a bank that has a mobile application. Management has asked you to build an ML-based biometric authentication for the app that verifies a customer's identity based on their fingerprint. Fingerprints are considered highly sensitive personal information and cannot be downloaded and stored into the bank databases. Which learning strategy should you recommend to train and deploy this ML model?

A. Differential privacy

B. Federated learning

C. MD5 to encrypt data

D. Data Loss Prevention API

Answer: B

Explanation:

Federated learning is a machine learning technique that enables organizations to train AI models on decentralized data without centralizing or sharing it1. It allows data privacy, continual learning, and better performance on end-user devices2. Federated learning works by sending the model parameters to the devices, where they are updated locally on the device's data, and then aggregating the updated parameters on a central server to form a global model3. This way, the data never leaves the device and the model can learn from a large and diverse dataset.

Federated learning is suitable for the use case of building an ML-based biometric authentication for the bank's mobile app that verifies a customer's identity based on their fingerprint. Fingerprints are considered highly sensitive personal information and cannot be downloaded and stored into the bank databases. By using federated learning, the bank can train and deploy an ML model that can recognize fingerprints without compromising the data privacy of the customers. The model can also adapt to the variations and changes in the fingerprints over time and improve its accuracy and reliability. Therefore, federated learning is the best learning strategy for this use case.

Question: 10

You are building a linear regression model on BigQuery ML to predict a customer's likelihood of purchasing your company's products. Your model uses a city name variable as a key predictive component. In order to train and serve the model, your data must be organized in columns. You want to prepare your data using the least amount of coding while maintaining the predictable variables. What should you do?

A. Create a new view with BigQuery that does not include a column with city information

B. Use Dataprep to transform the state column using a one-hot encoding method, and make each city a column with binary values.

C. Use Cloud Data Fusion to assign each city to a region labeled as 1, 2, 3, 4, or 5r and then use that number to represent the city in the model.

D. Use TensorFlow to create a categorical variable with a vocabulary list Create the vocabulary file, and upload it as part of your model to BigQuery ML.

Answer: B

Explanation:

One-hot encoding is a technique that converts categorical variables into numerical variables by creating dummy variables for each possible category. Each dummy variable has a value of 1 if the original variable belongs to that category, and 0 otherwise1. One-hot encoding can help linear regression models to capture the effect of different categories on the target variable without imposing any ordinal relationship among them2. Dataprep is a service that allows you to explore, clean, and transform your data for analysis and machine learning. You can use Dataprep to apply one-hot encoding to your city name variable and make each city a column with binary values3. This way, you can prepare your data using the least amount of coding while maintaining the predictive variables. Therefore, using Dataprep to transform the state column using a one-hot encoding method is the best option for this use case.

Reference:

One Hot Encoding: A Beginner's Guide

One-Hot Encoding in Linear Regression Models Dataprep documentation

Question: 11

You work for a toy manufacturer that has been experiencing a large increase in demand. You need to build an ML model to reduce the amount of time spent by quality control inspectors checking for product defects. Faster defect detection is a priority. The factory does not have reliable Wi-Fi. Your company wants to implement the new ML model as soon as

possible. Which model should you use?

A. AutoML Vision model
B. AutoML Vision Edge mobile-versatile-1 model
C. AutoML Vision Edge mobile-low-latency-1 model
D. AutoML Vision Edge mobile-high-accuracy-1 model

Answer: C

Explanation:
AutoML Vision Edge is a service that allows you to create custom image classification and object detection models that can run on edge devices, such as mobile phones, tablets, or IoT devices1. AutoML Vision Edge offers four types of models that vary in size, accuracy, and latency: mobile-versatile-1, mobile-low-latency-1, mobile-high-accuracy-1, and mobile-core-ml-low-latency- 12. Each model has its own trade-offs and use cases, depending on the device specifications and the application requirements.
For the use case of building an ML model to reduce the amount of time spent by quality control inspectors checking for product defects, the best model to use is the AutoML Vision Edge mobile- low- latency-1 model. This model is optimized for fast inference on mobile devices, with a latency of less than 50 milliseconds on a Pixel 1 phone2. Faster defect detection is a priority for the toy manufacturer, and the factory does not have reliable Wi-Fi, so a low-latency model that can run on the device without internet connection is ideal. The mobile-low-latency-1 model also has a small size of less than 4 MB, which makes it easy to deploy and update2. The mobile-low-latency-1 model has a slightly lower accuracy than the mobile-high-accuracy-1 model, but it is still suitable for most image classification tasks2. Therefore, the AutoML Vision Edge mobile-low-latency-1 model is the best option for this use case.
Reference:
AutoML Vision Edge documentation
AutoML Vision Edge model types

Question: 12

You are going to train a DNN regression model with Keras APIs using this code:

```
model - tf.keras.Sequential()
model.add(tf.keras.layers.Dense(
        256,
        use_bias-True,
        activation-’relu',
        kernel initializer—None,
        kernel regularizer=None, input_shape=(500,)))
model.add(tf.keras.layers.Dropout(rate-0. 25)) model.add(tf.keras.layers.Dense(
        128,     use bias=True, activation='relu', kernel initializer-’uniform', kernel
        regularize!?—' 12’))
model.add(tf.keras.layers.Dropout(rate-0.25)) model.add(tf.keras.layers.Dense(
```

2, use bias-False, activation-'softmax')) model.compile(loss-'mse*)

How many trainable weights does your model have? (The arithmetic below is correct.)

A. 501*256+257*128+2 161154
B. 500*256+256*128+128*2 161024
C. 501*256+257*128+128*2161408
D. 500*256*0 25+256*128*0 25+128*2 40448

Answer: B

Explanation:
The number of trainable weights in a DNN regression model with Keras APIs can be calculated by multiplying the number of input units by the number of output units for each layer, and adding the number of bias units for each layer. The bias units are usually equal to the number of output units, except for the last layer, which does not have bias units if the activation function is softmax1. In this code, the model has three layers: a dense layer with 256 units and relu activation, a dropout layer with 0.25 rate, and a dense layer with 2 units and softmax activation. The input shape is 500.
Therefore, the number of trainable weights is:
For the first layer: 500 input units * 256 output units + 256 bias units 128256
For the second layer: The dropout layer does not have any trainable weights, as it only randomly sets some of the input units to zero to prevent overfitting2.
For the third layer: 256 input units * 2 output units + 0 bias units 512
The total number of trainable weights is 128256 + 512 161024. Therefore, the correct answer is B. Reference:
How to calculate the number of parameters for a Convolutional Neural Network?
Dropout (keras.io)

Question: 13

You recently joined a machine learning team that will soon release a new project. As a lead on the project, you are asked to determine the production readiness of the ML components. The team has already tested features and data, model development, and infrastructure. Which additional readiness check should you recommend to the team?

A. Ensure that training is reproducible
B. Ensure that all hyperparameters are tuned
C. Ensure that model performance is monitored
D. Ensure that feature expectations are captured in the schema

Answer: C

Explanation:
Monitoring model performance is an essential part of production readiness, as it allows the team to detect and address any issues that may arise after deployment, such as data drift, model degradation, or errors.
Other Options:
A . Ensuring that training is reproducible is important for model development, but not necessarily for production readiness. Reproducibility helps the team to track and compare different experiments, but it does not guarantee that the model will perform well in production.
B . Ensuring that all hyperparameters are tuned is also important for model development, but not sufficient for production

readiness. Hyperparameter tuning helps the team to find the optimal configuration for the model, but it does not account for the dynamic and changing nature of the production environment.

D . Ensuring that feature expectations are captured in the schema is a part of testing features and data, which the team has already done. The schema defines the expected format, type, and range of the features, and helps the team to validate and preprocess the data.

Question: 14

You recently designed and built a custom neural network that uses critical dependencies specific to your organization's framework. You need to train the model using a managed training service on Google Cloud. However, the ML framework and related dependencies are not supported by AI Platform Training. Also, both your model and your data are too large to fit in memory on a single machine. Your ML framework of choice uses the scheduler, workers, and servers distribution structure. What should you do?

A. Use a built-in model available on AI Platform Training
B. Build your custom container to run jobs on AI Platform Training
C. Build your custom containers to run distributed training jobs on AI Platform Training
D. Reconfigure your code to a ML framework with dependencies that are supported by AI Platform Training

Answer: C

Explanation:

AI Platform Training is a service that allows you to run your machine learning training jobs on Google Cloud using various features, model architectures, and hyperparameters. You can use AI Platform Training to scale up your training jobs, leverage distributed training, and access specialized hardware such as GPUs and TPUs1. AI Platform Training supports several pre-built containers that provide different ML frameworks and dependencies, such as TensorFlow, PyTorch, scikit-learn, and XGBoost2. However, if the ML framework and related dependencies that you need are not supported by the pre-built containers, you can build your own custom containers and use them to run your training jobs on AI Platform Training3.

Custom containers are Docker images that you create to run your training application. By using custom containers, you can specify and pre-install all the dependencies needed for your application, and have full control over the code, serving, and deployment of your model4. Custom containers also enable you to run distributed training jobs on AI Platform Training, which can help you train large- scale and complex models faster and more efficiently5. Distributed training is a technique that splits the training data and computation across multiple machines, and coordinates them to update the model parameters. AI Platform Training supports two types of distributed training: parameter server and collective all-reduce. The parameter server architecture consists of a set of workers that perform the computation, and a set of servers that store and update the model parameters. The collective allreduce architecture consists of a set of workers that perform the computation and synchronize the model parameters among themselves. Both architectures also have a scheduler that coordinates the workers and servers.

For the use case of training a custom neural network that uses critical dependencies specific to your organization's framework, the best option is to build your custom containers to run distributed training jobs on AI Platform Training. This option allows you to use the ML framework and dependencies of your choice, and train your model on multiple machines without having to manage the infrastructure. Since your ML framework of choice uses the scheduler, workers, and servers distribution structure, you can use the parameter server architecture to run your distributed training job on AI Platform Training. You can specify the number and type of machines, the custom container image, and the training application arguments when you submit your training job. Therefore, building your custom containers to run distributed training jobs on AI Platform Training is the best option for this use case.

Reference:
AI Platform Training documentation
Pre-built containers for training
Custom containers for training
Custom containers overview | Vertex AI | Google Cloud
Distributed training overview
[Types of distributed training]
[Distributed training architectures]
[Using custom containers for training with the parameter server architecture]

Question: 15

You are an ML engineer in the contact center of a large enterprise. You need to build a sentiment analysis tool that predicts customer sentiment from recorded phone conversations. You need to identify the best approach to building a model while ensuring that the gender, age, and cultural differences of the customers who called the contact center do not impact any stage of the model development pipeline and results. What should you do?

A. Extract sentiment directly from the voice recordings
B. Convert the speech to text and build a model based on the words
C. Convert the speech to text and extract sentiments based on the sentences
D. Convert the speech to text and extract sentiment using syntactical analysis

Answer: C

Explanation:
Sentiment analysis is the process of identifying and extracting the emotions, opinions, and attitudes expressed in a text or speech. Sentiment analysis can help businesses understand their customers' feedback, satisfaction, and preferences. There are different approaches to building a sentiment analysis tool, depending on the input data and the output format. Some of the common approaches are:
Extracting sentiment directly from the voice recordings: This approach involves using acoustic features, such as pitch, intensity, and prosody, to infer the sentiment of the speaker. This approach can capture the nuances and subtleties of the vocal expression, but it also requires a large and diverse dataset of labeled voice recordings, which may not be easily available or accessible.
Moreover, this approach may not account for the semantic and contextual information of the speech, which can also affect the sentiment.
Converting the speech to text and building a model based on the words: This approach involves using automatic speech recognition (ASR) to transcribe the voice recordings into text, and then using lexical features, such as word frequency, polarity, and valence, to infer the sentiment of the text. This approach can leverage the existing text-based sentiment analysis models and tools, but it also introduces some challenges, such as the accuracy and reliability of the ASR system, the ambiguity and variability of the natural language, and the loss of the acoustic information of the speech. Converting the speech to text and extracting sentiments based on the sentences: This approach involves using ASR to transcribe the voice recordings into text, and then using syntactic and semantic features, such as sentence structure, word order, and meaning, to infer the sentiment of the text. This approach can capture the higher-level and complex aspects of the natural language, such as negation, sarcasm, and irony, which can affect the sentiment. However, this approach also requires more sophisticated and advanced natural language processing techniques, such as parsing, dependency analysis, and semantic role labeling, which may not be readily available or easy to implement.
Converting the speech to text and extracting sentiment using syntactical analysis: This approach involves using ASR to transcribe the voice recordings into text, and then using syntactical analysis, such as part-of-speech tagging, phrase chunking, and constituency parsing, to infer the sentiment of the text. This approach can identify the

grammatical and structural elements of the natural language, such as nouns, verbs, adjectives, and clauses, which can indicate the sentiment. However, this approach may not account for the pragmatic and contextual information of the speech, such as the speaker's intention, tone, and situation, which can also influence the sentiment.

For the use case of building a sentiment analysis tool that predicts customer sentiment from recorded phone conversations, the best approach is to convert the speech to text and extract sentiments based on the sentences. This approach can balance the trade-offs between the accuracy, complexity, and feasibility of the sentiment analysis tool, while ensuring that the gender, age, and cultural differences of the customers who called the contact center do not impact any stage of the model development pipeline and results. This approach can also handle different types and levels of sentiment, such as polarity (positive, negative, or neutral), intensity (strong or weak), and emotion (anger, joy, sadness, etc.). Therefore, converting the speech to text and extracting sentiments based on the sentences is the best approach for this use case.

Question: 16

You work for an advertising company and want to understand the effectiveness of your company's latest advertising campaign. You have streamed 500 MB of campaign data into BigQuery. You want to query the table, and then manipulate the results of that query with a pandas dataframe in an AI Platform notebook. What should you do?

A. Use AI Platform Notebooks' BigQuery cell magic to query the data, and ingest the results as a pandas dataframe
B. Export your table as a CSV file from BigQuery to Google Drive, and use the Google Drive API to ingest the file into your notebook instance
C. Download your table from BigQuery as a local CSV file, and upload it to your AI Platform notebook instance Use pandas. read_csv to ingest the file as a pandas dataframe
D. From a bash cell in your AI Platform notebook, use the bq extract command to export the table as a CSV file to Cloud Storage, and then use gsutii cp to copy the data into the notebook Use pandas. read_csv to ingest the file as a pandas dataframe

Answer: A

Explanation:

AI Platform Notebooks is a service that provides managed Jupyter notebooks for data science and machine learning. You can use AI Platform Notebooks to create, run, and share your code and analysis in a collaborative and interactive environment1. BigQuery is a service that allows you to analyze large-scale and complex data using SQL queries. You can use BigQuery to stream, store, and query your data in a fast and cost-effective way2. Pandas is a popular Python library that provides data structures and tools for data analysis and manipulation. You can use pandas to create, manipulate, and visualize dataframes, which are tabular data structures with rows and columns3. AI Platform Notebooks provides a cell magic, %%bigquery, that allows you to run SQL queries on BigQuery data and ingest the results as a pandas dataframe. A cell magic is a special command that applies to the whole cell in a Jupyter notebook. The %%bigquery cell magic can take various arguments, such as the name of the destination dataframe, the name of the destination table in

BigQuery, the project ID, and the query parameters4. By using the %%bigquery cell magic, you can query the data in BigQuery with minimal code and manipulate the results with pandas in AI Platform Notebooks. This is the most convenient and efficient way to achieve your goal.

The other options are not as good as option A, because they involve more steps, more code, and more manual effort. Option B requires you to export your table as a CSV file from BigQuery to Google Drive, and then use the Google Drive API to ingest the file into your notebook instance. This option is cumbersome and time-consuming, as it involves moving the data across different services and formats. Option C requires you to download your table from BigQuery as a local CSV file, and then upload it to your AI Platform notebook instance. This option is also inefficient and impractical, as it involves

downloading and uploading large files, which can take a long time and consume a lot of bandwidth. Option D requires you to use a bash cell in your AI Platform notebook to export the table as a CSV file to Cloud Storage, and then copy the data into the notebook. This option is also complex and unnecessary, as it involves using different commands and tools to move the data around. Therefore, option A is the best option for this use case.

Reference:

AI Platform Notebooks documentation

BigQuery documentation pandas documentation

Using Jupyter magics to query BigQuery data

Question: 17

You have trained a model on a dataset that required computationally expensive preprocessing operations. You need to execute the same preprocessing at prediction time. You deployed the model on AI Platform for high-throughput online prediction. Which architecture should you use?

A. • Validate the accuracy of the model that you trained on preprocessed data
- Create a new model that uses the raw data and is available in real time
- Deploy the new model onto AI Platform for online prediction

B. • Send incoming prediction requests to a Pub/Sub topic
- Transform the incoming data using a Dataflow job
- Submit a prediction request to AI Platform using the transformed data
- Write the predictions to an outbound Pub/Sub queue

C. • Stream incoming prediction request data into Cloud Spanner
- Create a view to abstract your preprocessing logic.
- Query the view every second for new records
- Submit a prediction request to AI Platform using the transformed data
- Write the predictions to an outbound Pub/Sub queue.

D. • Send incoming prediction requests to a Pub/Sub topic
- Set up a Cloud Function that is triggered when messages are published to the Pub/Sub topic.
- Implement your preprocessing logic in the Cloud Function
- Submit a prediction request to AI Platform using the transformed data
- Write the predictions to an outbound Pub/Sub queue

Answer: D

Explanation:

Option A is incorrect because creating a new model that uses the raw data and is available in real time would require retraining the model and deploying it again, which is not efficient or scalable. Option B is incorrect because using a Dataflow job to transform the incoming data would introduce unnecessary latency and complexity for online prediction, which requires fast and simple processing. Option C is incorrect because using Cloud Spanner to stream and query the incoming data would incur high costs and overhead for online prediction, which does not need a relational database. Option D is correct because using a Cloud Function to preprocess the data and submit a prediction request to AI Platform is a simple and scalable solution for online prediction, which leverages the serverless and event-driven features of Cloud Functions.

www.atmicnetworks.com

Question: 18

You are building a model to predict daily temperatures. You split the data randomly and then transformed the training and test datasets. Temperature data for model training is uploaded hourly. During testing, your model performed with 97% accuracy; however, after deploying to production, the model's accuracy dropped to 66%. How can you make your production model more accurate?

A. Normalize the data for the training, and test datasets as two separate steps.
B. Split the training and test data based on time rather than a random split to avoid leakage
C. Add more data to your test set to ensure that you have a fair distribution and sample for testing D. Apply data transformations before splitting, and cross-validate to make sure that the transformations are applied to both the training and test sets.

Answer: B

Explanation:
When building a model to predict daily temperatures, it is important to split the training and test data based on time rather than a random split. This is because temperature data is likely to have temporal dependencies and patterns, such as seasonality, trends, and cycles. If the data is split randomly, there is a risk of data leakage, which occurs when information from the future is used to train or validate the model. Data leakage can lead to overfitting and unrealistic performance estimates, as the model may learn from data that it should not have access to. By splitting the data based on time, such as using the most recent data as the test set and the older data as the training set, the model can be evaluated on how well it can forecast future temperatures based on past data, which is the realistic scenario in production. Therefore, splitting the data based on time rather than a random split is the best way to make the production model more accurate.

Question: 19

You have a demand forecasting pipeline in production that uses Dataflow to preprocess raw data prior to model training and prediction. During preprocessing, you employ Z-score normalization on data stored in BigQuery and write it back to BigQuery. New training data is added every week. You

want to make the process more efficient by minimizing computation time and manual intervention. What should you do?

A. Normalize the data using Google Kubernetes Engine
B. Translate the normalization algorithm into SQL for use with BigQuery
C. Use the normalizer_fn argument in TensorFlow's Feature Column API
D. Normalize the data with Apache Spark using the Dataproc connector for BigQuery

Answer: B

Explanation:
Z-score normalization is a technique that transforms the values of a numeric variable into standardized units, such that the mean is zero and the standard deviation is one. Z-score normalization can help to compare variables with different scales and ranges, and to reduce the effect of outliers and skewness. The formula for z-score normalization is: z (x - mu) / sigma where x is the original value, mu is the mean of the variable, and sigma is the standard deviation of the variable.
Dataflow is a service that allows you to create and run data processing pipelines on Google Cloud. You can use Dataflow to preprocess raw data prior to model training and prediction, such as applying z-score normalization on data stored in BigQuery. However, using Dataflow for this task may not be the most efficient option, as it involves reading and writing

data from and to BigQuery, which can be time-consuming and costly. Moreover, using Dataflow requires manual intervention to update the pipeline whenever new training data is added.

A more efficient way to perform z-score normalization on data stored in BigQuery is to translate the normalization algorithm into SQL and use it with BigQuery. BigQuery is a service that allows you to analyze large-scale and complex data using SQL queries. You can use BigQuery to perform z-score normalization on your data using SQL functions such as AVG(), STDDEV_POP(), and OVER(). For example, the following SQL query can normalize the values of a column called temperature in a table called weather:

SELECT (temperature - AVG(temperature) OVER ()) / STDDEV_POP(temperature) OVER () AS normalized_temperature FROM weather;

By using SQL to perform z-score normalization on BigQuery, you can make the process more efficient by minimizing computation time and manual intervention. You can also leverage the scalability and performance of BigQuery to handle large and complex datasets. Therefore, translating the normalization algorithm into SQL for use with BigQuery is the best option for this use case.

Question: 20

You were asked to investigate failures of a production line component based on sensor readings. After receiving the dataset, you discover that less than 1% of the readings are positive examples representing failure incidents. You have tried to train several classification models, but none of them converge. How should you resolve the class imbalance problem?

A. Use the class distribution to generate 10% positive examples
B. Use a convolutional neural network with max pooling and softmax activation
C. Downsample the data with upweighting to create a sample with 10% positive examples
D. Remove negative examples until the numbers of positive and negative examples are equal

Answer: C

Explanation:

The class imbalance problem is a common challenge in machine learning, especially in classification tasks. It occurs when the distribution of the target classes is highly skewed, such that one class (the majority class) has much more examples than the other class (the minority class). The minority class is often the more interesting or important class, such as failure incidents, fraud cases, or rare diseases. However, most machine learning algorithms are designed to optimize the overall accuracy, which can be biased towards the majority class and ignore the minority class. This can result in poor predictive performance, especially for the minority class.

There are different techniques to deal with the class imbalance problem, such as data-level methods, algorithm-level methods, and evaluation-level methods1. Data-level methods involve resampling the original dataset to create a more balanced class distribution. There are two main types of data-level methods: oversampling and undersampling. Oversampling methods increase the number of examples in the minority class, either by duplicating existing examples or by generating synthetic examples. Undersampling methods reduce the number of examples in the majority class, either by randomly removing examples or by using clustering or other criteria to select representative examples. Both oversampling and undersampling methods can be combined with upweighting or downweighting, which assign different weights to the examples according to their class frequency, to further balance the dataset.

For the use case of investigating failures of a production line component based on sensor readings, the best option is to downsample the data with upweighting to create a sample with 10% positive examples. This option involves randomly removing some of the negative examples (the majority class) until the ratio of positive to negative examples is 1:9, and then assigning higher weights to the positive examples to compensate for their low frequency. This option can create a more balanced dataset that can improve the performance of the classification models, while preserving the diversity and

representativeness of the original data. This option can also reduce the computation time and memory usage, as the size of the dataset is reduced. Therefore, downsampling the data with upweighting to create a sample with 10% positive examples is the best option for this use case. Reference:

A Systematic Study of the Class Imbalance Problem in Convolutional Neural Networks

Question: 21

You need to design a customized deep neural network in Keras that will predict customer purchases based on their purchase history. You want to explore model performance using multiple model architectures, store training data, and be able to compare the evaluation metrics in the same dashboard. What should you do?

A. Create multiple models using AutoML Tables
B. Automate multiple training runs using Cloud Composer
C. Run multiple training jobs on Al Platform with similar job names
D. Create an experiment in Kubeflow Pipelines to organize multiple runs

Answer: D

Explanation:

Kubeflow Pipelines is a service that allows you to create and run machine learning workflows on Google Cloud using various features, model architectures, and hyperparameters. You can use Kubeflow Pipelines to scale up your workflows, leverage distributed training, and access specialized hardware such as GPUs and TPUs1. An experiment in Kubeflow Pipelines is a workspace where you can try different configurations of your pipelines and organize your runs into logical groups. You can use experiments to compare the performance of different models and track the evaluation metrics in the same dashboard2.

For the use case of designing a customized deep neural network in Keras that will predict customer purchases based on their purchase history, the best option is to create an experiment in Kubeflow Pipelines to organize multiple runs. This option allows you to explore model performance using multiple model architectures, store training data, and compare the evaluation metrics in the same dashboard. You can use Keras to build and train your deep neural network models, and then package them as pipeline components that can be reused and combined with other components. You can also use Kubeflow Pipelines SDK to define and submit your pipelines programmatically, and use Kubeflow Pipelines UI to monitor and manage your experiments. Therefore, creating an experiment in Kubeflow Pipelines to organize multiple runs is the best option for this use case.

Reference:

Kubeflow Pipelines documentation

Experiment | Kubeflow

Question: 22

Your team needs to build a model that predicts whether images contain a driver's license, passport, or credit card. The data engineering team already built the pipeline and generated a dataset composed of 10,000 images with driver's licenses, 1,000 images with passports, and 1,000 images with credit cards. You now have to train a model with the following label map: ['driversjicense', 'passport', 'credit_card']. Which loss function should you use?

A. Categorical hinge
B. Binary cross-entropy

C. Categorical cross-entropy
D. Sparse categorical cross-entropy

Answer: C

Explanation:
Categorical cross-entropy is a loss function that is suitable for multi-class classification problems, where the target variable has more than two possible values. Categorical cross-entropy measures the difference between the true probability distribution of the target classes and the predicted probability distribution of the model. It is defined as:
L - sum(y_i * log(p_i))
where y_i is the true probability of class i, and p_i is the predicted probability of class i. Categorical cross-entropy penalizes the model for making incorrect predictions, and encourages the model to assign high probabilities to the correct classes and low probabilities to the incorrect classes. For the use case of building a model that predicts whether images contain a driver's license, passport, or credit card, categorical cross-entropy is the appropriate loss function to use. This is because the problem is a multi-class classification problem, where the target variable has three possible values: ['drivers_license', 'passport', 'credit_card']. The label map is a list that maps the class names to the class indices, such that 'drivers_license' corresponds to index 0, 'passport' corresponds to index 1, and 'credit_card' corresponds to index 2. The model should output a probability distribution over the three classes for each image, and the categorical cross-entropy loss function should compare the output with the true labels. Therefore, categorical cross-entropy is the best loss function for this use case.

Question: 23

You are an ML engineer at a global car manufacturer. You need to build an ML model to predict car sales in different cities around the world. Which features or feature crosses should you use to train city-specific relationships between car type and number of sales?

A. Three individual features binned latitude, binned longitude, and one-hot encoded car type
B. One feature obtained as an element-wise product between latitude, longitude, and car type
C. One feature obtained as an element-wise product between binned latitude, binned longitude, and one-hot encoded car type
D. Two feature crosses as a element-wise product the first between binned latitude and one-hot encoded car type, and the second between binned longitude and one-hot encoded car type

Answer: C

Explanation:
A feature cross is a synthetic feature that is obtained by combining two or more existing features, usually by taking their product or concatenation. A feature cross can help to capture the nonlinear and interaction effects between the original features, and improve the predictive performance of the model. A feature cross can be applied to different types of features, such as numeric, categorical, or geospatial features1.
For the use case of building an ML model to predict car sales in different cities around the world, the best option is to use one feature obtained as an element-wise product between binned latitude, binned longitude, and one-hot encoded car type. This option involves creating a feature cross that combines three individual features: binned latitude, binned longitude, and one-hot encoded car type. Binning is a technique that transforms a continuous numeric feature into a discrete categorical feature by dividing its range into equal intervals, or bins. One-hot encoding is a technique that

transforms a categorical feature into a binary vector, where each element corresponds to a possible category, and has a value of 1 if the feature belongs to that category, and 0 otherwise. By applying binning and one-hot encoding to the latitude, longitude, and car type features, the feature cross can capture the city-specific relationships between car type and number of sales, as each combination of bins and car types can represent a different city and its preference for a certain car type. For example, the feature cross can learn that a city with a latitude bin of [40, 50], a longitude bin of [-80, -70], and a car type of SUV has a higher number of sales than a city with a latitude bin of [-10, 0], a longitude bin of [10, 20], and a car type of sedan. Therefore, using one feature obtained as an

element-wise product between binned latitude, binned longitude, and one-hot encoded car type is the best option for this use case.

Reference:

[Feature Crosses | Machine Learning Crash Course]

Question: 24

You trained a text classification model. You have the following SignatureDefs:

```
signature def['serving default*]:
The given SavedModel SignatureDef contains the following input(s): inputs['text'] tensor info: dtype: DT
STRING shape: (-1, 2} name: serving_default_text:O
The given SavedModel SignatureDef contains the following output(s): outputs['Softmax'] tensor_info:
dtype: DT_FLOAT shape: (-1, 2) name: StatefulPartitionedCall:0
Method name is: tensorflow/serving/predict
```

You started a TensorFlow-serving component server and tried to send an HTTP request to get a prediction using:

```
headers — ("content-type": "application/json")
json_response = requests.post('http://Localhost:8501/vl/models/text_model:predict', data=data, headers-
headers)
```

What is the correct way to write the predict request?

A. data json.dumps({"signature_name": "serving_default'\ "instances": [fab', 'be1, 'cd']]})

B. data json dumps({"signature_name": "serving_default"! "instances": [['a', 'b', "c", 'd', 'e', 'f']]})

C. data json.dumps({"signature_name": "serving_default, "instances": [['a', 'b\ 'c'1, [d\ 'e\ T]]})

D. data json dumps({"signature_name": f,serving_default", "instances": [['a', 'b'], [c\ 'd'], ['e\ T]]})

Answer: D

Explanation:

A predict request is a way to send data to a trained model and get predictions in return. A predict request can be written in different formats, such as JSON, protobuf, or gRPC, depending on the service and the platform that are used to host and serve the model. A predict request usually contains the following information:

The signature name: This is the name of the signature that defines the inputs and outputs of the model. A signature is a way to specify the expected format, type, and shape of the data that the model can accept and produce. A signature can be specified when exporting or saving the model, or it can be automatically inferred by the service or the platform. A model can have multiple signatures, but only one can be used for each predict request.

The instances: This is the data that is sent to the model for prediction. The instances can be a single instance or a batch of instances, depending on the size and shape of the data. The instances should match the input specification of the signature, such as the number, name, and type of the input

tensors.

For the use case of training a text classification model, the correct way to write the predict request is D. data json.dumps({"signature_name": "serving_default", "instances": [['a', 'b'], ['c', 'd'], ['e', 'f']]}) This option involves writing the predict request in JSON format, which is a common and convenient format for sending and receiving data over the web. JSON stands for JavaScript Object Notation, and it is a way to represent data as a collection of name-value pairs or an ordered list of values. JSON can be easily converted to and from Python objects using the json module.

This option also involves using the signature name "serving_default", which is the default signature name that is assigned to the model when it is saved or exported without specifying a custom signature name. The serving_default signature defines the input and output tensors of the model based on the SignatureDef that is shown in the image. According to the SignatureDef, the model expects an input tensor called "text" that has a shape of (-1, 2) and a type of DT_STRING, and produces an output tensor called "softmax" that has a shape of (-1, 2) and a type of DT_FLOAT. The -1 in the shape indicates that the dimension can vary depending on the number of instances, and the 2 indicates that the dimension is fixed at 2. The DT_STRING and DT_FLOAT indicate that the data type is string and float, respectively.

This option also involves sending a batch of three instances to the model for prediction. Each instance is a list of two strings, such as ['a', 'b'], ['c', 'd'], or ['e', 'f']. These instances match the input specification of the signature, as they have a shape of (3, 2) and a type of string. The model will process these instances and produce a batch of three predictions, each with a softmax output that has a shape of (1, 2) and a type of float. The softmax output is a probability distribution over the two possible classes that the model can predict, such as positive or negative sentiment.

Therefore, writing the predict request as data json.dumps({"signature_name": "serving_default", "instances": [['a', 'b'], ['c', 'd'], ['e', 'f']]}) is the correct and valid way to send data to the text classification model and get predictions in return.

Reference:

[json — JSON encoder and decoder]

Question: 25

You work for a social media company. You need to detect whether posted images contain cars. Each training example is a member of exactly one class. You have trained an object detection neural network and deployed the model version to Al Platform Prediction for evaluation. Before deployment, you created an evaluation job and attached it to the Al Platform Prediction model version. You notice that the precision is lower than your business requirements allow. How should you adjust the model's final layer softmax threshold to increase precision?

A. Increase the recall
B. Decrease the recall.
C. Increase the number of false positives
D. Decrease the number of false negatives

Answer: B

Explanation:

Precision and recall are two common metrics for evaluating the performance of a classification model. Precision measures the proportion of positive predictions that are correct, while recall measures the proportion of positive examples that are correctly predicted. Precision and recall are inversely related, meaning that increasing one will decrease the other, and vice versa. The trade-off between precision and recall depends on the goal and the cost of the classification problem1.

For the use case of detecting whether posted images contain cars, precision is more important than recall, as the social media company wants to minimize the number of false positives, or images that are incorrectly labeled as containing

cars. A high precision means that the model is confident and accurate in its positive predictions, while a low recall means that the model may miss some positive examples, or images that actually contain cars. The cost of missing some positive examples is lower than the cost of making wrong positive predictions, as the latter may affect the user experience and the reputation of the social media company.

The softmax function is a function that transforms a vector of real numbers into a probability distribution over the possible classes. The softmax function is often used as the final layer of a neural network for multi-class classification problems, as it assigns a probability to each class, and the class with the highest probability is chosen as the prediction. The softmax function is defined as: softmax (x_i) exp (x_i) / sum_j exp (x_j)

where x_i is the input value for class i, and softmax (x_i) is the output probability for class i.

The softmax threshold is a parameter that determines the minimum probability that a class must have to be chosen as the prediction. For example, if the softmax threshold is 0.5, then the class with the highest probability must have at least 0.5 to be selected, otherwise the prediction is none. The softmax threshold can be used to adjust the trade-off between precision and recall, as a higher threshold will increase the precision and decrease the recall, while a lower threshold will decrease the precision and increase the recall2.

For the use case of detecting whether posted images contain cars, the best way to adjust the model's final layer softmax threshold to increase precision is to decrease the recall. This means that the softmax threshold should be increased, so that the model will only make positive predictions when it is highly confident, and avoid making false positives. By increasing the softmax threshold, the model will become more selective and accurate in its positive predictions, and improve the precision metric. Therefore, decreasing the recall is the best option for this use case.

Reference:

Precision and recall - Wikipedia

How to add a threshold in softmax scores - Stack Overflow

Question: 26

You developed an ML model with AI Platform, and you want to move it to production. You serve a few thousand queries per second and are experiencing latency issues. Incoming requests are served by a load balancer that distributes them across multiple Kubeflow CPU-only pods running on Google Kubernetes Engine (GKE). Your goal is to improve the serving latency without changing the underlying infrastructure. What should you do?

A. Significantly increase the max_batch_size TensorFlow Serving parameter
B. Switch to the tensorflow-model-server-universal version of TensorFlow Serving
C. Significantly increase the max_enqueued_batches TensorFlow Serving parameter
D. Recompile TensorFlow Serving using the source to support CPU-specific optimizations Instruct GKE to choose an appropriate baseline minimum CPU platform for serving nodes

Answer: D

Explanation:

TensorFlow Serving is a service that allows you to deploy and serve TensorFlow models in a scalable and efficient way. TensorFlow Serving supports various platforms and hardware, such as CPU, GPU, and TPU. However, the default TensorFlow Serving binaries are built with generic CPU instructions, which may not leverage the full potential of the CPU architecture. To improve the serving latency and performance, you can recompile TensorFlow Serving using the source code and enable CPU-specific optimizations, such as AVX, AVX2, and FMA1. These optimizations can speed up the computation and inference of the TensorFlow models, especially for deep neural networks.

Google Kubernetes Engine (GKE) is a service that allows you to run and manage containerized applications on Google Cloud using Kubernetes. GKE supports various types and sizes of nodes, which are the virtual machines that run the containers. GKE also supports different CPU platforms, which are the generations and models of the CPUs that power the nodes. GKE allows you to choose a baseline minimum CPU platform for your node pool, which is a group of nodes

with the same configuration. By choosing a baseline minimum CPU platform, you can ensure that your nodes have the CPU features and capabilities that match your workload requirements2.

For the use case of serving a few thousand queries per second and experiencing latency issues, the best option is to recompile TensorFlow Serving using the source to support CPU-specific optimizations, and instruct GKE to choose an appropriate baseline minimum CPU platform for serving nodes. This option can improve the serving latency and performance without changing the underlying infrastructure, as it only involves rebuilding the TensorFlow Serving binary and selecting the CPU platform for the GKE nodes. This option can also take advantage of the CPU-only pods that are running on GKE, as it can optimize the CPU utilization and efficiency. Therefore, recompiling TensorFlow Serving using the source to support CPU-specific optimizations and instructing GKE to choose an appropriate baseline minimum CPU platform for serving nodes is the best option for this use case.

Reference:

Building TensorFlow Serving from source

Specifying a minimum CPU platform for a node pool

Question: 27

You built and manage a production system that is responsible for predicting sales numbers. Model accuracy is crucial, because the production model is required to keep up with market changes. Since being deployed to production, the model hasn't changed; however the accuracy of the model has steadily deteriorated. What issue is most likely causing the steady decline in model accuracy?

A. Poor data quality
B. Lack of model retraining
C. Too few layers in the model for capturing information
D. Incorrect data split ratio during model training, evaluation, validation, and test

Answer: B

Explanation:

Model retraining is the process of updating an existing machine learning model with new data and parameters to improve its performance and accuracy. Model retraining is essential for maintaining the relevance and validity of the model, especially when the data or the environment changes over time. Model retraining can help to avoid or reduce the effects of model degradation, which is the phenomenon of the model's predictive performance decreasing as it is tested on new datasets within rapidly evolving environments1.

For the use case of predicting sales numbers, model accuracy is crucial, because the production model is required to keep up with market changes. Market changes can affect the demand, supply, price, and preference of the products, and thus influence the sales numbers. If the model is not retrained with new data that reflects the market changes, it may become outdated and inaccurate, and fail to capture the patterns and trends of the sales numbers. Therefore, the most likely issue that is causing the steady decline in model accuracy is the lack of model retraining.

The other options are not as likely as option B, because they are not directly related to the model's ability to adapt to market changes. Option A, poor data quality, may affect the model's accuracy, but it is not a specific cause of model degradation over time. Option C, too few layers in the model for capturing information, may affect the model's complexity and expressiveness, but it is not a specific cause of model degradation over time. Option D, incorrect data split ratio during model training, evaluation, validation, and test, may affect the model's generalization and validation, but it is not a specific cause of model degradation over time. Therefore, option B, lack of model retraining, is the best answer for this question.

Reference:

Beware Steep Decline: Understanding Model Degradation In Machine Learning Models

www.atmicnetworks.com

Question: 28

You are an ML engineer at a large grocery retailer with stores in multiple regions. You have been asked to create an inventory prediction model. Your models features include region, location, historical demand, and seasonal popularity. You want the algorithm to learn from new inventory data on a daily basis. Which algorithms should you use to build the model?

A. Classification
B. Reinforcement Learning
C. Recurrent Neural Networks (RNN)
D. Convolutional Neural Networks (CNN)

Answer: B

Explanation:
Reinforcement learning is a machine learning technique that enables an agent to learn from its own actions and feedback in an environment. Reinforcement learning does not require labeled data or explicit rules, but rather relies on trial and error and reward and punishment mechanisms to optimize the agent's behavior and achieve a goal. Reinforcement learning can be used to solve complex and dynamic problems that involve sequential decision making and adaptation to changing situations1.
For the use case of creating an inventory prediction model for a large grocery retailer with stores in multiple regions, reinforcement learning is a suitable algorithm to use. This is because the problem involves multiple factors that affect the inventory demand, such as region, location, historical demand, and seasonal popularity, and the inventory manager needs to make optimal decisions on

how much and when to order, store, and distribute the products. Reinforcement learning can help the inventory manager to learn from the new inventory data on a daily basis, and adjust the inventory policy accordingly. Reinforcement learning can also handle the uncertainty and variability of the inventory demand, and balance the trade-off between overstocking and understocking2. The other options are not as suitable as option B, because they are not designed to handle sequential decision making and adaptation to changing situations. Option A, classification, is a machine learning technique that assigns a label to an input based on predefined categories. Classification can be used to predict the inventory demand for a single product or a single period, but it cannot optimize the inventory policy over multiple products and periods. Option C, recurrent neural networks (RNN), are a type of neural network that can process sequential data, such as text, speech, or time series. RNN can be used to model the temporal patterns and dependencies of the inventory demand, but they cannot learn from feedback and rewards. Option D, convolutional neural networks (CNN), are a type of neural network that can process spatial data, such as images, videos, or graphs. CNN can be used to extract features and patterns from the inventory data, but they cannot optimize the inventory policy over multiple actions and states. Therefore, option B, reinforcement learning, is the best answer for this question.

Reference:
Reinforcement learning - Wikipedia
Reinforcement Learning for Inventory Optimization

Question: 29

You need to train a computer vision model that predicts the type of government ID present in a given image using a GPU-powered virtual machine on Compute Engine. You use the following parameters: • Optimizer: SGD

- Image shape 224x224

- Batch size 64
- Epochs 10
- Verbose 2

During training you encounter the following error: ResourceExhaustedError: out of Memory (oom) when allocating tensor. What should you do?

A. Change the optimizer B. Reduce the batch size C. Change the learning rate D. Reduce the image shape

Answer: B

Explanation:

A ResourceExhaustedError: out of memory (OOM) when allocating tensor is an error that occurs when the GPU runs out of memory while trying to allocate memory for a tensor. A tensor is a multidimensional array of numbers that represents the data or the parameters of a machine learning model. The size and shape of a tensor depend on various factors, such as the input data, the model architecture, the batch size, and the optimization algorithm1.

For the use case of training a computer vision model that predicts the type of government ID present in a given image using a GPU-powered virtual machine on Compute Engine, the best option to resolve the error is to reduce the batch size. The batch size is a parameter that determines how many input examples are processed at a time by the model. A larger batch size can improve the model's accuracy and stability, but it also requires more memory and computation. A smaller batch size can reduce the memory and computation requirements, but it may also affect the model's performance and convergence2.

By reducing the batch size, the GPU can allocate less memory for each tensor, and avoid running out of memory. Reducing the batch size can also speed up the training process, as the GPU can process more batches in parallel. However, reducing the batch size too much may also have some drawbacks, such as increasing the noise and variance of the gradient updates, and slowing down the convergence of the model. Therefore, the optimal batch size should be chosen based on the trade-off between memory, computation, and performance3.

The other options are not as effective as option B, because they are not directly related to the memory allocation of the GPU. Option A, changing the optimizer, may affect the speed and quality of the optimization process, but it may not reduce the memory usage of the model. Option C, changing the learning rate, may affect the convergence and stability of the model, but it may not reduce the memory usage of the model. Option D, reducing the image shape, may reduce the size of the input tensor, but it may also reduce the quality and resolution of the image, and affect the model's accuracy. Therefore, option B, reducing the batch size, is the best answer for this question. Reference:

ResourceExhaustedError: OOM when allocating tensor with shape - Stack Overflow How does batch size affect model performance and training time? - Stack Overflow How to choose an optimal batch size for training a neural network? - Stack Overflow

Question: 30

You have been asked to develop an input pipeline for an ML training model that processes images from disparate sources at a low latency. You discover that your input data does not fit in memory. How should you create a dataset following Google-recommended best practices?

A. Create a tf.data.Dataset.prefetch transformation

B. Convert the images to tf .Tensor Objects, and then run Dataset. from_tensor_slices{).

C. Convert the images to tf .Tensor Objects, and then run tf. data. Dataset. from_tensors ().

D. Convert the images Into TFRecords, store the images in Cloud Storage, and then use the tf. data API to read the

images for training

Answer: D

Explanation:

An input pipeline is a way to prepare and feed data to a machine learning model for training or inference. An input pipeline typically consists of several steps, such as reading, parsing, transforming, batching, and prefetching the data. An input pipeline can improve the performance and efficiency of the model, as it can handle large and complex datasets, optimize the data processing, and reduce the latency and memory usage1.

For the use case of developing an input pipeline for an ML training model that processes images from disparate sources at a low latency, the best option is to convert the images into TFRecords, store the images in Cloud Storage, and then use the tf.data API to read the images for training. This option

involves using the following components and techniques:

TFRecords: TFRecords is a binary file format that can store a sequence of data records, such as images, text, or audio. TFRecords can help to compress, serialize, and store the data efficiently, and reduce the data loading and parsing time. TFRecords can also support data sharding and interleaving, which can improve the data throughput and parallelism2.

Cloud Storage: Cloud Storage is a service that allows you to store and access data on Google Cloud. Cloud Storage can help to store and manage large and distributed datasets, such as images from different sources, and provide high availability, durability, and scalability. Cloud Storage can also integrate with other Google Cloud services, such as Compute Engine, AI Platform, and Dataflow3. tf.data API: tf.data API is a set of tools and methods that allow you to create and manipulate data pipelines in TensorFlow. tf.data API can help to read, transform, batch, and prefetch the data efficiently, and optimize the data processing for performance and memory. tf.data API can also support various data sources and formats, such as TFRecords, CSV, JSON, and images.

By using these components and techniques, the input pipeline can process large datasets of images from disparate sources that do not fit in memory, and provide low latency and high performance for the ML training model. Therefore, converting the images into TFRecords, storing the images in Cloud Storage, and using the tf.data API to read the images for training is the best option for this use case. Reference:

Build TensorFlow input pipelines | TensorFlow Core

TFRecord and tf.Example | TensorFlow Core

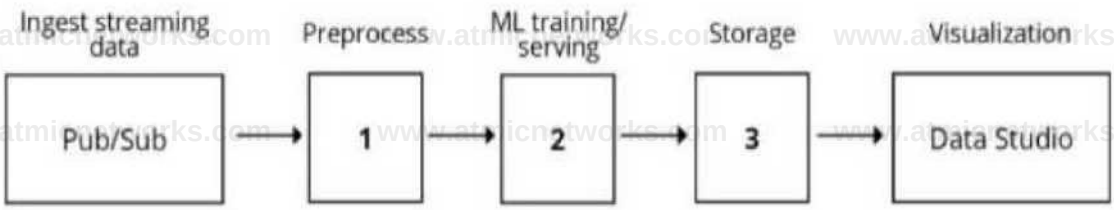
Cloud Storage documentation | Google Cloud

[tf.data: Build TensorFlow input pipelines | TensorFlow Core]

Question: 31

You are building an ML model to detect anomalies in real-time sensor dat

a. You will use Pub/Sub to handle incoming requests. You want to store the results for analytics and visualization. How should you configure the pipeline?



A. 1 Dataflow, 2 - AI Platform, 3 BigQuery

B. 1 DataProc, 2 AutoML, 3 Cloud Bigtable

C. 1 BigQuery, 2 AutoML, 3 Cloud Functions

D. 1 BigQuery, 2 AI Platform, 3 Cloud Storage

Answer: A

Explanation:
Dataflow is a fully managed service for executing Apache Beam pipelines that can process streaming or batch data1.
AI Platform is a unified platform that enables you to build and run machine learning applications across Google Cloud2.
BigQuery is a serverless, highly scalable, and cost-effective cloud data warehouse designed for business agility3.
These services are suitable for building an ML model to detect anomalies in real-time sensor data, as they can handle large-scale data ingestion, preprocessing, training, serving, storage, and visualization. The other options are not as suitable because:
DataProc is a service for running Apache Spark and Apache Hadoop clusters, which are not optimized for streaming data processing4.
AutoML is a suite of machine learning products that enables developers with limited machine learning expertise to train high-quality models specific to their business needs5. However, it does not support custom models or real-time predictions.
Cloud Bigtable is a scalable, fully managed NoSQL database service for large analytical and operational workloads. However, it is not designed for ad hoc queries or interactive analysis. Cloud Functions is a serverless execution environment for building and connecting cloud services. However, it is not suitable for storing or visualizing data.
Cloud Storage is a service for storing and accessing data on Google Cloud. However, it is not a data warehouse and does not support SQL queries or visualization tools.

Question: 32

You have a functioning end-to-end ML pipeline that involves tuning the hyperparameters of your ML model using AI Platform, and then using the best-tuned parameters for training. Hypertuning is taking longer than expected and is delaying the downstream processes. You want to speed up the tuning job without significantly compromising its effectiveness. Which actions should you take? Choose 2 answers

A. Decrease the number of parallel trials
B. Decrease the range of floating-point values
C. Set the early stopping parameter to TRUE
D. Change the search algorithm from Bayesian search to random search.
E. Decrease the maximum number of trials during subsequent training phases.

Answer: C, E

Explanation:
Hyperparameter tuning is the process of finding the optimal values for the parameters of a machine learning model that affect its performance. AI Platform provides a service for hyperparameter tuning that can run multiple trials in parallel and use different search algorithms to find the best combination of hyperparameters. However, hyperparameter tuning can be time-consuming and costly, especially if the search space is large and the model training is complex. Therefore, it is important to optimize the tuning job to reduce the time and resources required.
One way to speed up the tuning job is to set the early stopping parameter to TRUE. This means that the tuning service will automatically stop trials that are unlikely to perform well based on the intermediate results. This can save time and

resources by avoiding unnecessary computations for trials that are not promising. The early stopping parameter can be set in the trainingInput.hyperparameters field of the training job request1

Another way to speed up the tuning job is to decrease the maximum number of trials during subsequent training phases. This means that the tuning service will use fewer trials to refine the search space after the initial phase. This can reduce the time required for the tuning job to converge to the optimal solution. The maximum number of trials can be set in the trainingInput.hyperparameters.maxTrials field of the training job request1

The other options are not effective ways to speed up the tuning job. Decreasing the number of parallel trials will reduce the concurrency of the tuning job and increase the overall time required. Decreasing the range of floating-point values will reduce the diversity of the search space and may miss some optimal solutions. Changing the search algorithm from Bayesian search to random search will reduce the efficiency of the tuning job and may require more trials to find the best solution1 References: 1: Hyperparameter tuning overview

Question: 33

You have written unit tests for a Kubeflow Pipeline that require custom libraries. You want to automate the execution of unit tests with each new push to your development branch in Cloud Source Repositories. What should you do?

A. Write a script that sequentially performs the push to your development branch and executes the unit tests on Cloud Run

B. Using Cloud Build, set an automated trigger to execute the unit tests when changes are pushed to your development branch.

C. Set up a Cloud Logging sink to a Pub/Sub topic that captures interactions with Cloud Source Repositories Configure a Pub/Sub trigger for Cloud Run, and execute the unit tests on Cloud Run. D. Set up a Cloud Logging sink to a Pub/Sub topic that captures interactions with Cloud Source Repositories. Execute the unit tests using a Cloud Function that is triggered when messages are sent to the Pub/Sub topic

Answer: B

Explanation:

Cloud Build is a service that executes your builds on Google Cloud Platform infrastructure. Cloud Build can import source code from Cloud Source Repositories, Cloud Storage, GitHub, or Bitbucket, execute a build to your specifications, and produce artifacts such as Docker containers or Java archives1

Cloud Build allows you to set up automated triggers that start a build when changes are pushed to a source code repository. You can configure triggers to filter the changes based on the branch, tag, or file path2

To automate the execution of unit tests for a Kubeflow Pipeline that require custom libraries, you can use Cloud Build to set an automated trigger to execute the unit tests when changes are pushed to your development branch in Cloud Source Repositories. You can specify the steps of the build in a YAML or JSON file, such as installing the custom libraries, running the unit tests, and reporting the results. You can also use Cloud Build to build and deploy the Kubeflow Pipeline components if the unit tests pass3

The other options are not recommended or feasible. Writing a script that sequentially performs the push to your development branch and executes the unit tests on Cloud Run is not a good practice, as it does not leverage the benefits of Cloud Build and its integration with Cloud Source Repositories. Setting up a Cloud Logging sink to a Pub/Sub topic that captures interactions with Cloud Source

Repositories and using a Pub/Sub trigger for Cloud Run or Cloud Function to execute the unit tests is unnecessarily

complex and inefficient, as it adds extra steps and latency to the process. Cloud Run and Cloud Function are also not designed for executing unit tests, as they have limitations on the memory, CPU, and execution time45 Reference: 1: Cloud Build overview 2: Creating and managing build triggers 3: Building and deploying Kubeflow Pipelines using Cloud Build 4: Cloud Run documentation 5: Cloud Functions documentation

Question: 34

You have trained a deep neural network model on Google Cloud. The model has low loss on the training data, but is performing worse on the validation dat

a. You want the model to be resilient to overfitting. Which strategy should you use when retraining the model?

A. Apply a dropout parameter of 0 2, and decrease the learning rate by a factor of 10

B. Apply a L2 regularization parameter of 0.4, and decrease the learning rate by a factor of 10.

C. Run a hyperparameter tuning job on Al Platform to optimize for the L2 regularization and dropout parameters

D. Run a hyperparameter tuning job on Al Platform to optimize for the learning rate, and increase the number of neurons by a factor of 2.

Answer: C

Explanation:

Overfitting occurs when a model tries to fit the training data so closely that it does not generalize well to new data. Overfitting can be caused by having a model that is too complex for the data, such as having too many parameters or layers. Overfitting can lead to poor performance on the validation data, which reflects how the model will perform on unseen data1

To prevent overfitting, one strategy is to use regularization techniques that penalize the complexity of the model and encourage it to learn simpler patterns. Two common regularization techniques for deep neural networks are L2 regularization and dropout. L2 regularization adds a term to the loss function that is proportional to the squared magnitude of the model's weights. This term penalizes large weights and encourages the model to use smaller weights. Dropout randomly drops out some units in the network during training, which prevents co-adaptation of features and reduces the effective number of parameters. Both L2 regularization and dropout have hyperparameters that control the strength of the regularization effect23

Another strategy to prevent overfitting is to use hyperparameter tuning, which is the process of finding the optimal values for the parameters of the model that affect its performance. Hyperparameter tuning can help find the best combination of hyperparameters that minimize the validation loss and improve the generalization ability of the model. AI Platform provides a service for hyperparameter tuning that can run multiple trials in parallel and use different search algorithms to find the best solution.

Therefore, the best strategy to use when retraining the model is to run a hyperparameter tuning job on AI Platform to optimize for the L2 regularization and dropout parameters. This will allow the model to find the optimal balance between fitting the training data and generalizing to new data. The other options are not as effective, as they either use fixed values for the regularization parameters, which may not be optimal, or they do not address the issue of overfitting at all. References: 1: Generalization: Peril of Overfitting 2: Regularization for Deep Learning 3: Dropout: A Simple Way to Prevent Neural Networks from Overfitting : [Hyperparameter tuning overview]

www.atmicnetworks.com

Question: 35

You are training a Resnet model on AI Platform using TPUs to visually categorize types of defects in automobile engines. You capture the training profile using the Cloud TPU profiler plugin and observe that it is highly input-bound. You want to reduce the bottleneck and speed up your model training process. Which modifications should you make to the tf .data dataset?

Choose 2 answers

A. Use the interleave option for reading data
B. Reduce the value of the repeat parameter
C. Increase the buffer size for the shuffle option.
D. Set the prefetch option equal to the training batch size
E. Decrease the batch size argument in your transformation

Answer: A, D

Explanation:

The tf.data dataset is a TensorFlow API that provides a way to create and manipulate data pipelines for machine learning. The tf.data dataset allows you to apply various transformations to the data, such as reading, shuffling, batching, prefetching, and interleaving. These transformations can affect the performance and efficiency of the model training process1

One of the common performance issues in model training is input-bound, which means that the model is waiting for the input data to be ready and is not fully utilizing the computational resources. Input-bound can be caused by slow data loading, insufficient parallelism, or large data size. Inputbound can be detected by using the Cloud TPU profiler plugin, which is a tool that helps you analyze the performance of your model on Cloud TPUs. The Cloud TPU profiler plugin can show you the percentage of time that the TPU cores are idle, which indicates input-bound2

To reduce the input-bound bottleneck and speed up the model training process, you can make some modifications to the tf.data dataset. Two of the modifications that can help are:

Use the interleave option for reading data. The interleave option allows you to read data from multiple files in parallel and interleave their records. This can improve the data loading speed and reduce the idle time of the TPU cores. The interleave option can be applied by using

the tf.data.Dataset.interleave method, which takes a function that returns a dataset for each input element, and a number of parallel calls3

Set the prefetch option equal to the training batch size. The prefetch option allows you to prefetch the next batch of data while the current batch is being processed by the model. This can reduce the latency between batches and improve the throughput of the model training. The prefetch option can be applied by using the tf.data.Dataset.prefetch method, which takes a buffer size argument. The buffer size should be equal to the training batch size, which is the number of examples per batch4 The other options are not effective or counterproductive. Reducing the value of the repeat parameter will reduce the number of epochs, which is the number of times the model sees the entire dataset. This can affect the model's accuracy and convergence. Increasing the buffer size for the shuffle option will increase the randomness of the data, but also increase the memory usage and

the data loading time. Decreasing the batch size argument in your transformation will reduce the number of examples per batch, which can affect the model's stability and performance.

Reference: 1: tf.data: Build TensorFlow input pipelines 2: Cloud TPU Tools in TensorBoard 3: tf.data.Dataset.interleave 4: tf.data.Dataset.prefetch : [Better performance with the tf.data API]

www.atmicnetworks.com

Question: 36

You work for a public transportation company and need to build a model to estimate delay times for multiple transportation routes. Predictions are served directly to users in an app in real time. Because different seasons and population increases impact the data relevance, you will retrain the model every month. You want to follow Google-recommended best practices. How should you configure the end-to-end architecture of the predictive model?

A. Configure Kubeflow Pipelines to schedule your multi-step workflow from training to deploying your model.

B. Use a model trained and deployed on BigQuery ML and trigger retraining with the scheduled query feature in BigQuery

C. Write a Cloud Functions script that launches a training and deploying job on Ai Platform that is triggered by Cloud Scheduler

D. Use Cloud Composer to programmatically schedule a Dataflow job that executes the workflow from training to deploying your model

Answer: A

Explanation:

The end-to-end architecture of the predictive model for estimating delay times for multiple transportation routes should be configured using Kubeflow Pipelines. Kubeflow Pipelines is a platform for building and deploying scalable, portable, and reusable machine learning pipelines on Kubernetes. Kubeflow Pipelines allows you to orchestrate your multi-step workflow from data preparation, model training, model evaluation, model deployment, and model serving. Kubeflow Pipelines also provides a user interface for managing and tracking your pipeline runs, experiments, and artifacts1

Using Kubeflow Pipelines has several advantages for this use case:

Full automation: You can define your pipeline as a Python script that specifies the steps and dependencies of your workflow, and use the Kubeflow Pipelines SDK to compile and upload your pipeline to the Kubeflow Pipelines service. You can also use the Kubeflow Pipelines UI to create, run, and monitor your pipeline2

Scalability: You can leverage the power of Kubernetes to scale your pipeline components horizontally and vertically, and use distributed training frameworks such as TensorFlow or PyTorch to train your model on multiple nodes or GPUs3

Portability: You can package your pipeline components as Docker containers that can run on any Kubernetes cluster, and use the Kubeflow Pipelines SDK to export and import your pipeline packages across different environments4

Reusability: You can reuse your pipeline components across different pipelines, and share your components with other users through the Kubeflow Pipelines Component Store. You can also use pre-built components from the Kubeflow Pipelines library or other sources5

Schedulability: You can use the Kubeflow Pipelines UI or the Kubeflow Pipelines SDK to schedule recurring pipeline runs based on cron expressions or intervals. For example, you can schedule your pipeline to run every month to retrain your model on the latest data.

The other options are not as suitable for this use case. Using a model trained and deployed on BigQuery ML is not recommended, as BigQuery ML is mainly designed for simple and quick machine learning tasks on large-scale data, and does not support complex models or custom code. Writing a Cloud Functions script that launches a training and deploying job on AI Platform is not ideal, as Cloud Functions has limitations on the memory, CPU, and execution time, and does not provide a user interface for managing and tracking your pipeline. Using Cloud Composer to programmatically schedule a Dataflow job that executes the workflow from training to deploying your model is not optimal, as Dataflow is mainly designed for data processing and streaming analytics, and does not support model serving or monitoring.

Reference: 1: Kubeflow Pipelines overview 2: Build a pipeline 3: Scale your machine learning training and prediction

workloads 4: Export and import pipelines 5: Build components and pipelines : [Schedule recurring pipeline runs] : [BigQuery ML overview] : [Cloud Functions documentation] : [Dataflow documentation]

Question: 37

You are an ML engineer at a global shoe store. You manage the ML models for the company's website. You are asked to build a model that will recommend new products to the user based on their purchase behavior and similarity with other users. What should you do?

A. Build a classification model
B. Build a knowledge-based filtering model
C. Build a collaborative-based filtering model
D. Build a regression model using the features as predictors

Answer: C

Explanation:
A recommender system is a type of machine learning system that suggests relevant items to users based on their preferences and behavior. Recommender systems are widely used in e-commerce, media, and entertainment industries to enhance user experience and increase revenue1
There are different types of recommender systems that use different filtering methods to generate recommendations. The most common types are:
Content-based filtering: This method uses the features of the items and the users to find the similarity between them. For example, a content-based recommender system for movies may use the genre, director, cast, and ratings of the movies, and the preferences, demographics, and history of the users, to recommend movies that are similar to the ones the user liked before2 Collaborative filtering: This method uses the feedback and ratings of the users to find the similarity between them and the items. For example, a collaborative filtering recommender system for books may use the ratings of the users for different books, and recommend books that are liked by other users who have similar ratings to the target user3
Hybrid method: This method combines content-based and collaborative filtering methods to overcome the limitations of each method and improve the accuracy and diversity of the recommendations. For example, a hybrid recommender system for music may use both the features

of the songs and the artists, and the ratings and listening habits of the users, to recommend songs that match the user's taste and preferences4
Deep learning-based: This method uses deep neural networks to learn complex and non-linear patterns from the data and generate recommendations. Deep learning-based recommender systems can handle large-scale and high-dimensional data, and incorporate various types of information, such as text, images, audio, and video. For example, a deep learning-based recommender system for fashion may use the images and descriptions of the products, and the profiles and feedback of the users, to recommend products that suit the user's style and preferences.
For the use case of building a model that will recommend new products to the user based on their purchase behavior and similarity with other users, the best option is to build a collaborative-based filtering model. This is because collaborative filtering can leverage the implicit feedback and ratings of the users to find the items that are most likely to interest them. Collaborative filtering can also help discover new products that the user may not be aware of, and increase the diversity and serendipity of the recommendations3
The other options are not as suitable for this use case. Building a classification model or a regression model using the features as predictors is not a good idea, as these models are not designed for recommendation tasks, and may not capture the preferences and behavior of the users. Building a knowledge-based filtering model is not relevant, as this

method uses the explicit knowledge and requirements of the users to find the items that meet their criteria, and does not rely on the purchase behavior or similarity with other users.

Reference: 1: Recommender system 2: Content-based filtering 3: Collaborative filtering 4: Hybrid recommender system : [Deep learning for recommender systems] : [Knowledge-based recommender system]

Question: 38

You are training an LSTM-based model on AI Platform to summarize text using the following job submission script:

```
gcloud ai-platform jobs submit training $JOB_NAME \
        --package-path $TRAINER_PACKAGE_PATH \
        --module-name $MAIN_TRAINER_MODULE \
        --job-dir $JOB_DIR \
        --region $REGION \
        --scale-tier basic \
        -- \
        --epochs 20 \
        --batch size=32 \
        --learning rate=0.001 \
```

You want to ensure that training time is minimized without significantly compromising the accuracy of your model. What should you do?

A. Modify the 'epochs' parameter

B. Modify the 'scale-tier' parameter
C. Modify the batch size' parameter
D. Modify the 'learning rate' parameter

Answer: B

Explanation:
The training time of a machine learning model depends on several factors, such as the complexity of the model, the size of the data, the hardware resources, and the hyperparameters. To minimize the training time without significantly compromising the accuracy of the model, one should optimize these factors as much as possible.
One of the factors that can have a significant impact on the training time is the scale-tier parameter, which specifies the type and number of machines to use for the training job on AI Platform. The scaletier parameter can be one of the predefined values, such as BASIC, STANDARD_1, PREMIUM_1, or BASIC_GPU, or a custom value that allows you to configure the machine type, the number of workers, and the number of parameter servers1
To speed up the training of an LSTM-based model on AI Platform, one should modify the scale-tier parameter to use a higher tier or a custom configuration that provides more computational resources, such as more CPUs, GPUs, or TPUs. This can reduce the training time by increasing the parallelism and throughput of the model training. However, one should also consider the trade-off between the training time and the cost, as higher tiers or custom configurations may incur higher charges2
The other options are not as effective or may have adverse effects on the model accuracy. Modifying the epochs

parameter, which specifies the number of times the model sees the entire dataset, may reduce the training time, but also affect the model's convergence and performance. Modifying the batch size parameter, which specifies the number of examples per batch, may affect the model's stability and generalization ability, as well as the memory usage and the gradient update frequency. Modifying the learning rate parameter, which specifies the step size of the gradient descent optimization, may affect the model's convergence and performance, as well as the risk of overshooting or getting stuck in local minima3

Reference: 1: Using predefined machine types 2: Distributed training 3: Hyperparameter tuning overview

Question: 39

You are training a TensorFlow model on a structured data set with 100 billion records stored in several CSV files. You need to improve the input/output execution performance. What should you do?

A. Load the data into BigQuery and read the data from BigQuery.
B. Load the data into Cloud Bigtable, and read the data from Bigtable
C. Convert the CSV files into shards of TFRecords, and store the data in Cloud Storage
D. Convert the CSV files into shards of TFRecords, and store the data in the Hadoop Distributed File System (HDFS)

Answer: C

Explanation:

The input/output execution performance of a TensorFlow model depends on how efficiently the model can read and process the data from the data source. Reading and processing data from CSV files can be slow and inefficient, especially if the data is large and distributed. Therefore, to improve the input/output execution performance, one should use a more suitable data format and storage system.

One of the best options for improving the input/output execution performance is to convert the CSV files into shards of TFRecords, and store the data in Cloud Storage. TFRecord is a binary data format that can store a sequence of serialized TensorFlow examples. TFRecord has several advantages over CSV, such as:

Faster data loading: TFRecord can be read and processed faster than CSV, as it avoids the overhead of parsing and decoding the text data. TFRecord also supports compression and checksums, which can reduce the data size and ensure data integrity1

Better performance: TFRecord can improve the performance of the model, as it allows the model to access the data in a sequential and streaming manner, and leverage the tf.data API to build efficient data pipelines. TFRecord also supports sharding and interleaving, which can increase the parallelism and throughput of the data processing2

Easier integration: TFRecord can integrate seamlessly with TensorFlow, as it is the native data format for TensorFlow. TFRecord also supports various types of data, such as images, text, audio, and video, and can store the data schema and metadata along with the data3

Cloud Storage is a scalable and reliable object storage service that can store any amount of data. Cloud Storage has several advantages over other storage systems, such as:

High availability: Cloud Storage can provide high availability and durability for the data, as it replicates the data across multiple regions and zones, and supports versioning and lifecycle management. Cloud Storage also offers various storage classes, such as Standard, Nearline, Coldline, and Archive, to meet different performance and cost requirements4

Low latency: Cloud Storage can provide low latency and high bandwidth for the data, as it supports HTTP and HTTPS protocols, and integrates with other Google Cloud services, such as AI Platform, Dataflow, and BigQuery. Cloud Storage also supports resumable uploads and downloads, and parallel composite uploads, which can improve the data transfer speed and reliability5

Easy access: Cloud Storage can provide easy access and management for the data, as it supports various tools and libraries, such as gsutil, Cloud Console, and Cloud Storage Client Libraries. Cloud Storage also supports fine-grained access control and encryption, which can ensure the data security and privacy.

The other options are not as effective or feasible. Loading the data into BigQuery and reading the data from BigQuery is not recommended, as BigQuery is mainly designed for analytical queries on large-scale data, and does not support streaming or real-time data processing. Loading the data into Cloud Bigtable and reading the data from Bigtable is not ideal, as Cloud Bigtable is mainly designed for low-latency and high-throughput key-value operations on sparse and wide tables, and does not support complex data types or schemas. Converting the CSV files into shards of TFRecords and storing the data in the Hadoop Distributed File System (HDFS) is not optimal, as HDFS is not natively supported by TensorFlow, and requires additional configuration and dependencies, such as Hadoop, Spark, or Beam.

Reference: 1: TFRecord and tf.Example 2: Better performance with the tf.data API 3: TensorFlow Data Validation 4: Cloud Storage overview 5: Performance : [How-to guides]

Question: 40

You have deployed multiple versions of an image classification model on AI Platform. You want to monitor the performance of the model versions overtime. How should you perform this comparison?

A. Compare the loss performance for each model on a held-out dataset.
B. Compare the loss performance for each model on the validation data
C. Compare the receiver operating characteristic (ROC) curve for each model using the What-If Tool
D. Compare the mean average precision across the models using the Continuous Evaluation feature

Answer: D

Explanation:

The performance of an image classification model can be measured by various metrics, such as accuracy, precision, recall, F1-score, and mean average precision (mAP). These metrics can be calculated based on the confusion matrix, which compares the predicted labels and the true labels of the images1

One of the best ways to monitor the performance of multiple versions of an image classification model on AI Platform is to compare the mean average precision across the models using the Continuous Evaluation feature. Mean average precision is a metric that summarizes the precision and recall of a model across different confidence thresholds and classes. Mean average precision is especially useful for multi-class and multi-label image classification problems, where the model has to assign one or more labels to each image from a set of possible labels. Mean average precision can range from 0 to 1, where a higher value indicates a better performance2

Continuous Evaluation is a feature of AI Platform that allows you to automatically evaluate the performance of your deployed models using online prediction requests and responses. Continuous Evaluation can help you monitor the quality and consistency of your models over time, and detect any issues or anomalies that may affect the model performance. Continuous Evaluation can also provide various evaluation metrics and visualizations, such as accuracy, precision, recall, F1-score, ROC curve, and confusion matrix, for different types of models, such as classification, regression, and object detection3

To compare the mean average precision across the models using the Continuous Evaluation feature, you need to do the following steps:

Enable the online prediction logging for each model version that you want to evaluate. This will allow AI Platform to collect the prediction requests and responses from your models and store them in BigQuery4

Create an evaluation job for each model version that you want to evaluate. This will allow AI Platform to compare the predicted labels and the true labels of the images, and calculate the evaluation metrics, such as mean average precision. You need to specify the BigQuery table that contains the prediction logs, the data schema, the label column, and the evaluation interval.

View the evaluation results for each model version on the AI Platform Models page in the Google Cloud console. You

can see the mean average precision and other metrics for each model version over time, and compare them using charts and tables. You can also filter the results by different classes and confidence thresholds.
The other options are not as effective or feasible. Comparing the loss performance for each model on a held-out dataset or on the validation data is not a good idea, as the loss function may not reflect the actual performance of the model on the online prediction data, and may vary depending on the choice of the loss function and the optimization algorithm. Comparing the receiver operating
characteristic (ROC) curve for each model using the What-If Tool is not possible, as the What-If Tool does not support image data or multi-class classification problems.
Reference: 1: Confusion matrix 2: Mean average precision 3: Continuous Evaluation
overview 4: Configure online prediction logging : [Create an evaluation job] : [View evaluation results] : [What-If Tool overview]

Question: 41

Your team trained and tested a DNN regression model with good results. Six months after deployment, the model is performing poorly due to a change in the distribution of the input dat a. How should you address the input differences in production?

A. Create alerts to monitor for skew, and retrain the model.
B. Perform feature selection on the model, and retrain the model with fewer features
C. Retrain the model, and select an L2 regularization parameter with a hyperparameter tuning service
D. Perform feature selection on the model, and retrain the model on a monthly basis with fewer features

Answer: A

Explanation:
The performance of a DNN regression model can degrade over time due to a change in the distribution of the input data. This phenomenon is known as data drift or concept drift, and it can affect the accuracy and reliability of the model predictions. Data drift can be caused by various factors, such as seasonal changes, population shifts, market trends, or external events1 To address the input differences in production, one should create alerts to monitor for skew, and retrain the model. Skew is a measure of how much the input data in production differs from the input data used for training the model. Skew can be detected by comparing the statistics and distributions of the input features in the training and production data, such as mean, standard deviation, histogram, or quantiles. Alerts can be set up to notify the model developers or operators when the skew exceeds a certain threshold, indicating a significant change in the input data2
When an alert is triggered, the model should be retrained with the latest data that reflects the current distribution of the input features. Retraining the model can help the model adapt to the new data and improve its performance. Retraining the model can be done manually or automatically, depending on the frequency and severity of the data drift. Retraining the model can also involve updating the model architecture, hyperparameters, or optimization algorithm, if necessary3 The other options are not as effective or feasible. Performing feature selection on the model and retraining the model with fewer features is not a good idea, as it may reduce the expressiveness and complexity of the model, and ignore some important features that may affect the output. Retraining the model and selecting an L2 regularization parameter with a hyperparameter tuning service is not relevant, as L2 regularization is a technique to prevent overfitting, not data drift. Retraining the model on a monthly basis with fewer features is not optimal, as it may not capture the timely changes in the input data, and may compromise the model performance.
Reference: 1: Data drift detection for machine learning models 2: Skew and drift detection 3: Retraining machine learning models

www.atmicnetworks.com

Question: 42

You manage a team of data scientists who use a cloud-based backend system to submit training jobs. This system has become very difficult to administer, and you want to use a managed service instead. The data scientists you work with use many different frameworks, including Keras, PyTorch, theano. Scikit-team, and custom libraries. What should you do?

A. Use the AI Platform custom containers feature to receive training jobs using any framework
B. Configure Kubeflow to run on Google Kubernetes Engine and receive training jobs through TFJob
C. Create a library of VM images on Compute Engine; and publish these images on a centralized repository
D. Set up Slurm workload manager to receive jobs that can be scheduled to run on your cloud infrastructure.

Answer: A

Explanation:

A cloud-based backend system is a system that runs on a cloud platform and provides services or resources to other applications or users. A cloud-based backend system can be used to submit training jobs, which are tasks that involve training a machine learning model on a given dataset using a specific framework and configuration1

However, a cloud-based backend system can also have some drawbacks, such as:

High maintenance: A cloud-based backend system may require a lot of administration and management, such as provisioning, scaling, monitoring, and troubleshooting the cloud resources and services. This can be time-consuming and costly, and may distract from the core business objectives2 Low flexibility: A cloud-based backend system may not support all the frameworks and libraries that the data scientists need to use for their training jobs. This can limit the choices and capabilities of the data scientists, and affect the quality and performance of their models3

Poor integration: A cloud-based backend system may not integrate well with other cloud services or tools that the data scientists need to use for their machine learning workflows, such as data processing, model deployment, or model monitoring. This can create compatibility and interoperability issues, and reduce the efficiency and productivity of the data scientists.

Therefore, it may be better to use a managed service instead of a cloud-based backend system to submit training jobs.

A managed service is a service that is provided and operated by a third-party provider, and offers various benefits, such as:

Low maintenance: A managed service handles the administration and management of the cloud resources and services, and abstracts away the complexity and details of the underlying infrastructure. This can save time and money, and allow the data scientists to focus on their core tasks2

High flexibility: A managed service can support multiple frameworks and libraries that the data scientists need to use for their training jobs, and allow them to customize and configure their training environments and parameters. This can enhance the choices and capabilities of the data scientists, and improve the quality and performance of their models3

Easy integration: A managed service can integrate seamlessly with other cloud services or tools that the data scientists need to use for their machine learning workflows, and provide a unified and consistent interface and experience. This can solve the compatibility and interoperability issues, and increase the efficiency and productivity of the data scientists.

One of the best options for using a managed service to submit training jobs is to use the AI Platform custom containers feature to receive training jobs using any framework. AI Platform is a Google Cloud service that provides a platform for building, deploying, and managing machine learning models. AI Platform supports various machine learning frameworks, such as TensorFlow, PyTorch, scikit-learn, and XGBoost, and provides various features, such as hyperparameter tuning, distributed training, online prediction, and model monitoring.

The AI Platform custom containers feature allows the data scientists to use any framework or library that they want for

www.atmicnetworks.com

their training jobs, and package their training application and dependencies as a Docker container image. The data scientists can then submit their training jobs to AI Platform, and specify the container image and the training parameters. AI Platform will run the training jobs on the cloud infrastructure, and handle the scaling, logging, and monitoring of the training jobs. The data scientists can also use the AI Platform features to optimize, deploy, and manage their models.

The other options are not as suitable or feasible. Configuring Kubeflow to run on Google Kubernetes Engine and receive training jobs through TFJob is not ideal, as Kubeflow is mainly designed for TensorFlow-based training jobs, and does not support other frameworks or libraries. Creating a library of VM images on Compute Engine and publishing these images on a centralized repository is not optimal, as Compute Engine is a low-level service that requires a lot of administration and management, and does not provide the features and integrations of AI Platform. Setting up Slurm workload manager to receive jobs that can be scheduled to run on your cloud infrastructure is not relevant, as Slurm is a tool for managing and scheduling jobs on a cluster of nodes, and does not provide a managed service for training jobs.

Reference: 1: Cloud computing 2: Managed services 3: Machine learning frameworks : [Machine learning workflow] : [AI Platform overview] : [Custom containers for training]

Question: 43

You are developing a Kubeflow pipeline on Google Kubernetes Engine. The first step in the pipeline is to issue a query against BigQuery. You plan to use the results of that query as the input to the next step in your pipeline. You want to achieve this in the easiest way possible. What should you do?

A. Use the BigQuery console to execute your query and then save the query results Into a new BigQuery table.
B. Write a Python script that uses the BigQuery API to execute queries against BigQuery Execute this script as the first step in your Kubeflow pipeline
C. Use the Kubeflow Pipelines domain-specific language to create a custom component that uses the Python BigQuery client library to execute queries
D. Locate the Kubeflow Pipelines repository on GitHub Find the BigQuery Query Component, copy that component's URL, and use it to load the component into your pipeline. Use the component to execute queries against BigQuery

Answer: D

Explanation:

Kubeflow is an open source platform for developing, orchestrating, deploying, and running scalable and portable machine learning workflows on Kubernetes. Kubeflow Pipelines is a component of Kubeflow that allows you to build and manage end-to-end machine learning pipelines using a graphical user interface or a Python-based domain-specific language (DSL). Kubeflow Pipelines can

help you automate and orchestrate your machine learning workflows, and integrate with various Google Cloud services and tools1

One of the Google Cloud services that you can use with Kubeflow Pipelines is BigQuery, which is a serverless, scalable, and cost-effective data warehouse that allows you to run fast and complex queries on large-scale data. BigQuery can help you analyze and prepare your data for machine learning, and store and manage your machine learning models2

To execute a query against BigQuery as the first step in your Kubeflow pipeline, and use the results of that query as the input to the next step in your pipeline, the easiest way to do that is to use the BigQuery Query Component, which is a pre-built component that you can find in the Kubeflow Pipelines repository on GitHub. The BigQuery Query Component allows you to run a SQL query on BigQuery, and output the results as a table or a file. You can use the component's URL

to load the component into your pipeline, and specify the query and the output parameters. You can then use the output of the component as the input to the next step in your pipeline, such as a data processing or a model training step3

The other options are not as easy or feasible. Using the BigQuery console to execute your query and then save the query results into a new BigQuery table is not a good idea, as it does not integrate with your Kubeflow pipeline, and requires manual intervention and duplication of data. Writing a Python script that uses the BigQuery API to execute queries against BigQuery is not ideal, as it requires writing custom code and handling authentication and error handling. Using the Kubeflow Pipelines DSL to create a custom component that uses the Python BigQuery client library to execute queries is not optimal, as it requires creating and packaging a Docker container image for the component, and testing and debugging the component.

Reference: 1: Kubeflow Pipelines overview 2: BigQuery overview 3: BigQuery Query Component

Question: 44

You are developing ML models with AI Platform for image segmentation on CT scans. You frequently update your model architectures based on the newest available research papers, and have to rerun training on the same dataset to benchmark their performance. You want to minimize computation costs and manual intervention while having version control for your code. What should you do?

A. Use Cloud Functions to identify changes to your code in Cloud Storage and trigger a retraining job B. Use the gcloud command-line tool to submit training jobs on AI Platform when you update your code

C. Use Cloud Build linked with Cloud Source Repositories to trigger retraining when new code is pushed to the repository

D. Create an automated workflow in Cloud Composer that runs daily and looks for changes in code in Cloud Storage using a sensor.

Answer: C

Explanation:

Developing ML models with AI Platform for image segmentation on CT scans requires a lot of computation and experimentation, as image segmentation is a complex and challenging task that involves assigning a label to each pixel in an image. Image segmentation can be used for various medical applications, such as tumor detection, organ segmentation, or lesion localization1

To minimize the computation costs and manual intervention while having version control for the code, one should use Cloud Build linked with Cloud Source Repositories to trigger retraining when new code is pushed to the repository. Cloud Build is a service that executes your builds on Google Cloud Platform infrastructure. Cloud Build can import source code from Cloud Source Repositories, Cloud Storage, GitHub, or Bitbucket, execute a build to your specifications, and produce artifacts such as Docker containers or Java archives2

Cloud Build allows you to set up automated triggers that start a build when changes are pushed to a source code repository. You can configure triggers to filter the changes based on the branch, tag, or file path3

Cloud Source Repositories is a service that provides fully managed private Git repositories on Google Cloud Platform. Cloud Source Repositories allows you to store, manage, and track your code using the Git version control system. You can also use Cloud Source Repositories to connect to other Google Cloud services, such as Cloud Build, Cloud Functions, or Cloud Run4

To use Cloud Build linked with Cloud Source Repositories to trigger retraining when new code is pushed to the

repository, you need to do the following steps:

Create a Cloud Source Repository for your code, and push your code to the repository. You can use the Cloud SDK, Cloud Console, or Cloud Source Repositories API to create and manage your repository5

Create a Cloud Build trigger for your repository, and specify the build configuration and the trigger settings. You can use the Cloud SDK, Cloud Console, or Cloud Build API to create and manage your trigger.

Specify the steps of the build in a YAML or JSON file, such as installing the dependencies, running the tests, building the container image, and submitting the training job to AI Platform. You can also use the Cloud Build predefined or custom build steps to simplify your build configuration.

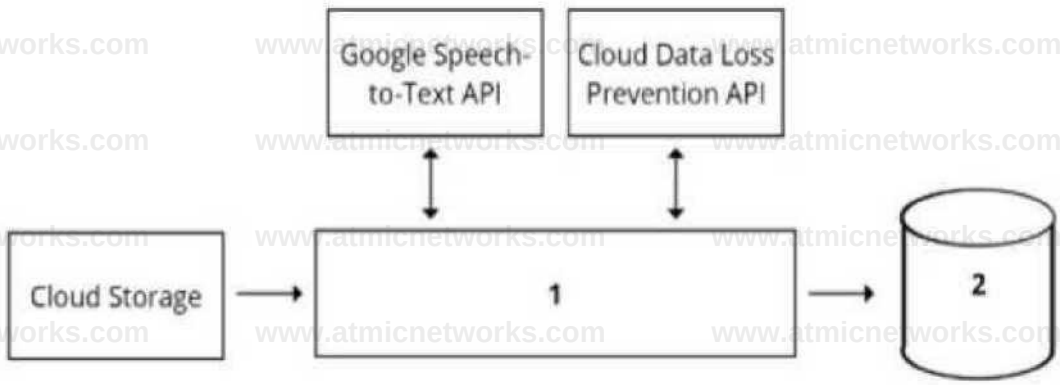
Push your new code to the repository, and the trigger will start the build automatically. You can monitor the status and logs of the build using the Cloud SDK, Cloud Console, or Cloud Build API. The other options are not as easy or feasible. Using Cloud Functions to identify changes to your code in Cloud Storage and trigger a retraining job is not ideal, as Cloud Functions has limitations on the memory, CPU, and execution time, and does not provide a user interface for managing and tracking your builds. Using the gcloud command-line tool to submit training jobs on AI Platform when you update your code is not optimal, as it requires manual intervention and does not leverage the benefits of Cloud Build and its integration with Cloud Source Repositories. Creating an automated workflow in Cloud Composer that runs daily and looks for changes in code in Cloud Storage using a sensor is not relevant, as Cloud Composer is mainly designed for orchestrating complex workflows across multiple systems, and does not provide a version control system for your code.

Reference: 1: Image segmentation 2: Cloud Build overview 3: Creating and managing build triggers 4: Cloud Source Repositories overview 5: Quickstart: Create a repository : [Quickstart: Create a build trigger] : [Configuring builds] : [Viewing build results]

Question: 45

Your organization's call center has asked you to develop a model that analyzes customer sentiments in each call. The call center receives over one million calls daily, and data is stored in Cloud Storage. The data collected must not leave the region in which the call originated, and no Personally Identifiable Information (PII) can be stored or analyzed. The data science team has a third-party tool

for visualization and access which requires a SQL ANSI-2011 compliant interface. You need to select components for data processing and for analytics. How should the data pipeline be designed?



A. 1 Dataflow, 2 BigQuery
B. 1 Pub/Sub, 2 Datastore
C. 1 Dataflow, 2 Cloud SQL
D. 1 Cloud Function, 2 Cloud SQL

Answer: A

Explanation:

A data pipeline is a set of steps or processes that move data from one or more sources to one or more destinations, usually for the purpose of analysis, transformation, or storage. A data pipeline can be designed using various components, such as data sources, data processing tools, data storage systems, and data analytics tools1

To design a data pipeline for analyzing customer sentiments in each call, one should consider the following requirements and constraints:

The call center receives over one million calls daily, and data is stored in Cloud Storage. This implies that the data is large, unstructured, and distributed, and requires a scalable and efficient data processing tool that can handle various types of data formats, such as audio, text, or image.

The data collected must not leave the region in which the call originated, and no Personally Identifiable Information (PII) can be stored or analyzed. This implies that the data is sensitive and subject to data privacy and compliance regulations, and requires a secure and reliable data storage system that can enforce data encryption, access control, and regional policies.

The data science team has a third-party tool for visualization and access which requires a SQL ANSI- 2011 compliant interface. This implies that the data analytics tool is external and independent of the data pipeline, and requires a standard and compatible data interface that can support SQL queries and operations.

One of the best options for selecting components for data processing and for analytics is to use Dataflow for data processing and BigQuery for analytics. Dataflow is a fully managed service for executing Apache Beam pipelines for data processing, such as batch or stream processing, extracttransform-load (ETL), or data integration. BigQuery is a serverless, scalable, and cost-effective data warehouse that allows you to run fast and complex queries on large-scale data23 Using Dataflow and BigQuery has several advantages for this use case:

Dataflow can process large and unstructured data from Cloud Storage in a parallel and distributed manner, and apply various transformations, such as converting audio to text, extracting sentiment scores, or anonymizing PII. Dataflow can also handle both batch and stream processing, which can enable real-time or near-real-time analysis of the call data.

BigQuery can store and analyze the processed data from Dataflow in a secure and reliable way, and enforce data encryption, access control, and regional policies. BigQuery can also support SQL ANSI- 2011 compliant interface, which can enable the data science team to use their third-party tool for visualization and access. BigQuery can also integrate with various Google Cloud services and tools, such as AI Platform, Data Studio, or Looker.

Dataflow and BigQuery can work seamlessly together, as they are both part of the Google Cloud ecosystem, and support various data formats, such as CSV, JSON, Avro, or Parquet. Dataflow and BigQuery can also leverage the benefits of Google Cloud infrastructure, such as scalability, performance, and cost-effectiveness.

The other options are not as suitable or feasible. Using Pub/Sub for data processing and Datastore for analytics is not ideal, as Pub/Sub is mainly designed for event-driven and asynchronous messaging, not data processing, and Datastore is mainly designed for low-latency and high-throughput key-value operations, not analytics. Using Cloud Function for data processing and Cloud SQL for analytics is not optimal, as Cloud Function has limitations on the memory, CPU, and execution time, and does not support complex data processing, and Cloud SQL is a relational database service that may not scale well for large-scale data. Using Cloud Composer for data processing and Cloud SQL for analytics is not relevant, as Cloud Composer is mainly designed for orchestrating complex workflows across multiple systems, not data processing, and Cloud SQL is a relational database service that may not scale well for large-scale data.

Reference: 1: Data pipeline 2: Dataflow overview 3: BigQuery overview : [Dataflow documentation] : [BigQuery documentation]

www.atmicnetworks.com

Question: 46

You work for an online retail company that is creating a visual search engine. You have set up an end- to-end ML pipeline on Google Cloud to classify whether an image contains your company's product. Expecting the release of new products in the near future, you configured a retraining functionality in the pipeline so that new data can be fed into your ML models. You also want to use Al Platform's continuous evaluation service to ensure that the models have high accuracy on your test data set. What should you do?

A. Keep the original test dataset unchanged even if newer products are incorporated into retraining B. Extend your test dataset with images of the newer products when they are introduced to retraining

C. Replace your test dataset with images of the newer products when they are introduced to retraining.

D. Update your test dataset with images of the newer products when your evaluation metrics drop below a pre-decided threshold.

Answer: B

Explanation:

The test dataset is used to evaluate the performance of the ML model on unseen data. It should reflect the distribution of the data that the model will encounter in production. Therefore, if the

retraining data includes new products, the test dataset should also be extended with images of those products to ensure that the model can generalize well to them. Keeping the original test dataset unchanged or replacing it entirely with images of the new products would not capture the diversity of the data that the model needs to handle. Updating the test dataset only when the evaluation metrics drop below a threshold would be reactive rather than proactive, and might result in poor user experience if the model fails to recognize the new products. Reference: Continuous evaluation documentation

Preparing and using test sets

Question: 47

You are responsible for building a unified analytics environment across a variety of on-premises data marts. Your company is experiencing data quality and security challenges when integrating data across the servers, caused by the use of a wide range of disconnected tools and temporary solutions. You need a fully managed, cloud-native data integration service that will lower the total cost of work and reduce repetitive work. Some members on your team prefer a codeless interface for building Extract, Transform, Load (ETL) process. Which service should you use?

A. Dataflow

B. Dataprep

C. Apache Flink

D. Cloud Data Fusion

Answer: D

Explanation:

Cloud Data Fusion is a fully managed, cloud-native data integration service that helps users efficiently build and manage ETL/ELT data pipelines. It provides a graphical interface to increase time efficiency and reduce complexity, and allows users to easily create and explore data pipelines using a code-free, point and click visual interface. Cloud Data Fusion

also supports a broad range of data sources and formats, including on-premises data marts, and ensures data quality and security by using built-in transformation capabilities and Cloud Data Loss Prevention. Cloud Data Fusion lowers the total cost of ownership by handling performance, scalability, availability, security, and compliance needs automatically. Reference:

Cloud Data Fusion documentation Cloud Data Fusion overview

Question: 48

You want to rebuild your ML pipeline for structured data on Google Cloud. You are using PySpark to conduct data transformations at scale, but your pipelines are taking over 12 hours to run. To speed up development and pipeline run time, you want to use a serverless tool and SQL syntax. You have already moved your raw data into Cloud Storage. How should you build the pipeline on Google Cloud while meeting the speed and processing requirements?

A. Use Data Fusion's GUI to build the transformation pipelines, and then write the data into BigQuery B. Convert your PySpark into SparkSQL queries to transform the data and then run your pipeline on Dataproc to write the data into BigQuery.

C. Ingest your data into Cloud SQL convert your PySpark commands into SQL queries to transform the data, and then use federated queries from BigQuery for machine learning

D. Ingest your data into BigQuery using BigQuery Load, convert your PySpark commands into BigQuery SQL queries to transform the data, and then write the transformations to a new table

Answer: D

Explanation:

BigQuery is a serverless, scalable, and cost-effective data warehouse that allows users to run SQL queries on large volumes of data. BigQuery Load is a tool that can ingest data from Cloud Storage into BigQuery tables. BigQuery SQL is a dialect of SQL that supports many of the same functions and operations as PySpark, such as window functions, aggregate functions, joins, and subqueries. By using BigQuery Load and BigQuery SQL, you can rebuild your ML pipeline for structured data on Google Cloud without having to manage any servers or clusters, and with faster performance and lower cost than using PySpark on Dataproc. You can also use BigQuery ML to create and evaluate ML models using SQL commands. Reference:

BigQuery documentation
BigQuery Load documentation
BigQuery SQL reference
BigQuery ML documentation

Question: 49

You are building a real-time prediction engine that streams files which may contain Personally Identifiable Information (PII) to Google Cloud. You want to use the Cloud Data Loss Prevention (DLP) API to scan the files. How should you ensure that the PII is not accessible by unauthorized individuals?

A. Stream all files to Google CloudT and then write the data to BigQuery Periodically conduct a bulk scan of the table using the DLP API.

B. Stream all files to Google Cloud, and write batches of the data to BigQuery While the data is being written to

BigQuery conduct a bulk scan of the data using the DLP API.

C. Create two buckets of data Sensitive and Non-sensitive Write all data to the Non-sensitive bucket Periodically conduct a bulk scan of that bucket using the DLP API, and move the sensitive data to the Sensitive bucket

D. Create three buckets of data: Quarantine, Sensitive, and Non-sensitive Write all data to the Quarantine bucket.

E. Periodically conduct a bulk scan of that bucket using the DLP API, and move the data to either the Sensitive or Non-Sensitive bucket

Answer: D

Explanation:
The Cloud DLP API is a service that allows users to inspect, classify, and de-identify sensitive data. It can be used to scan data in Cloud Storage, BigQuery, Cloud Datastore, and Cloud Pub/Sub. The best way to ensure that the PII is not accessible by unauthorized individuals is to use a quarantine bucket to store the data before scanning it with the DLP API. This way, the data is isolated from other applications and users until it is classified and moved to the appropriate bucket. The other options are not as secure or efficient, as they either expose the data to BigQuery before scanning, or scan the data after writing it to a non-sensitive bucket. Reference:

Cloud DLP documentation

Scanning and classifying Cloud Storage files

Question: 50

You are designing an ML recommendation model for shoppers on your company's ecommerce website. You will use Recommendations AI to build, test, and deploy your system. How should you develop recommendations that increase revenue while following best practices?

A. Use the "Other Products You May Like" recommendation type to increase the click-through rate B. Use the "Frequently Bought Together' recommendation type to increase the shopping cart size for each order.

C. Import your user events and then your product catalog to make sure you have the highest quality event stream

D. Because it will take time to collect and record product data, use placeholder values for the product catalog to test the viability of the model.

Answer: B

Explanation:
Recommendations AI is a service that allows users to build, test, and deploy personalized product recommendations for their ecommerce websites. It uses Google's deep learning models to learn from user behavior and product data, and generate high-quality recommendations that can increase revenue, click-through rate, and customer satisfaction. One of the best practices for using Recommendations AI is to choose the right recommendation type for the business objective. The "Frequently Bought Together" recommendation type shows products that are often purchased together with the current product, and encourages users to add more items to their shopping cart. This can increase the average order value and the revenue for each transaction. The other options are not as effective or feasible for this objective. The "Other Products You May Like" recommendation type shows products that are similar to the current product, and may increase the click-through rate, but not necessarily the shopping cart size. Importing the user events and then the product catalog is not a recommended order, as it may cause data inconsistency and missing recommendations. The product catalog should be imported first, and then the user events. Using placeholder values for the product catalog is not a viable option, as it will not produce meaningful recommendations or reflect the real performance of the model.
Reference: Recommendations AI documentation

Choosing a recommendation type

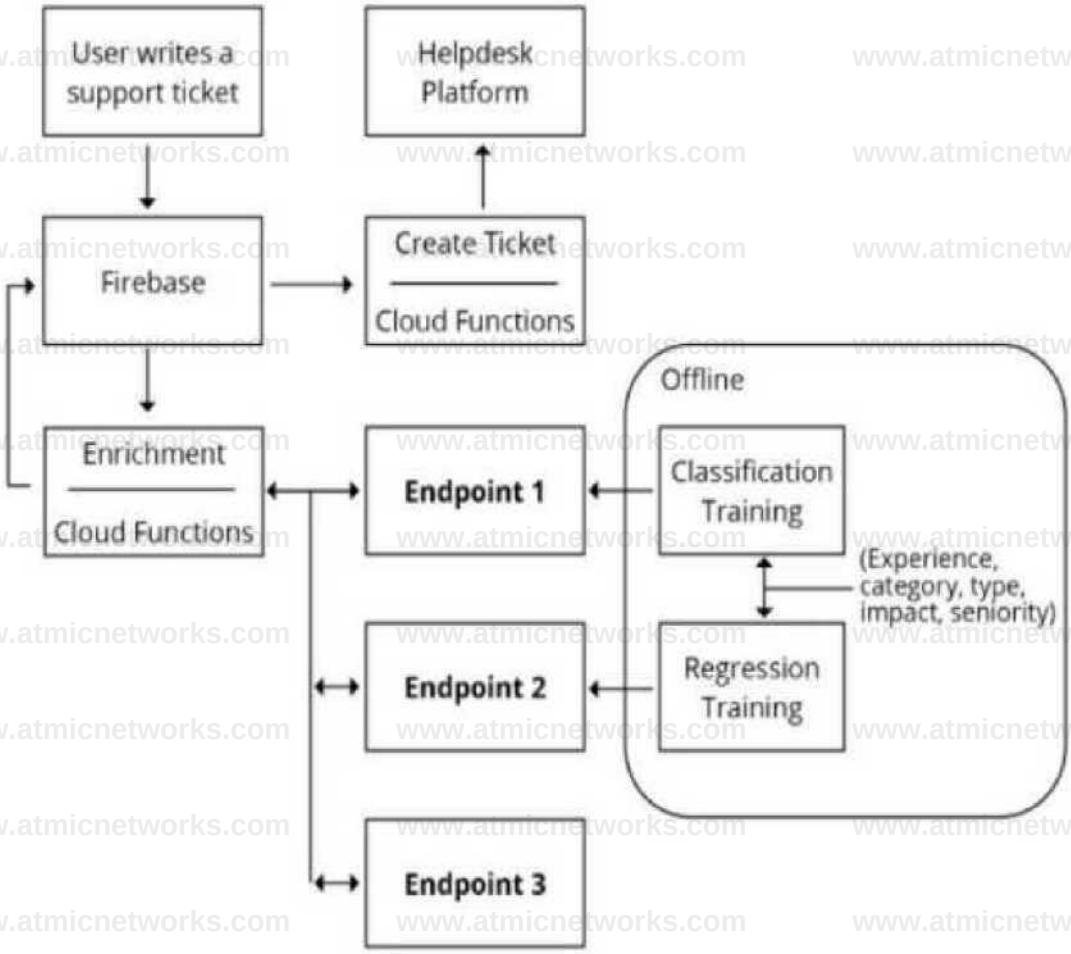
Importing data to Recommendations AI

www.atmicnetworks.com

Question: 51

You are designing an architecture with a serverless ML system to enrich customer support tickets with informative metadata before they are routed to a support agent. You need a set of models to predict ticket priority, predict ticket resolution time, and perform sentiment analysis to help agents make strategic decisions when they process support requests. Tickets are not expected to have any domain-specific terms or jargon.

The proposed architecture has the following flow:



Which endpoints should the Enrichment Cloud Functions call?

A. 1 Vertex Al. 2 Vertex Al. 3 AutoML Natural Language
B. 1 Vertex Al. 2 Vertex Al. 3 Cloud Natural Language API
C. 1 Vertex Al. 2 Vertex Al. 3 AutoML Vision
D. 1 Cloud Natural Language API. 2 Vertex Al, 3 Cloud Vision API

Answer: B

Explanation:

Vertex AI is a unified platform for building and deploying ML models on Google Cloud. It supports both custom and AutoML models, and provides various tools and services for ML development, such as Vertex Pipelines, Vertex Vizier, Vertex Explainable AI, and Vertex Feature Store. Vertex AI can be used to create models for predicting ticket priority and resolution time, as these are domain-specific tasks that require custom training data and evaluation metrics. Cloud

Natural Language API is a pretrained service that provides natural language understanding capabilities, such as sentiment analysis, entity analysis, syntax analysis, and content classification. Cloud Natural Language API can be used to perform sentiment analysis on the support tickets, as this is a general task that does not require domain-specific knowledge or jargon. The other options are not suitable for the given architecture. AutoML Natural Language and AutoML Vision are services that allow users to create custom natural language and vision models using their own data and labels. They are not needed for sentiment analysis, as Cloud Natural Language API already provides this functionality. Cloud Vision API is a pretrained service that provides image analysis capabilities, such as object detection, face detection, text detection, and image labeling. It is not relevant for the support tickets, as they are not expected to have any images. Reference:

Vertex AI documentation

Cloud Natural Language API documentation

Question: 52

You work with a data engineering team that has developed a pipeline to clean your dataset and save it in a Cloud Storage bucket. You have created an ML model and want to use the data to refresh your model as soon as new data is available. As part of your CI/CD workflow, you want to automatically run a Kubeflow Pipelines training job on Google Kubernetes Engine (GKE). How should you architect this workflow?

A. Configure your pipeline with Dataflow, which saves the files in Cloud Storage After the file is saved, start the training job on a GKE cluster

B. Use App Engine to create a lightweight python client that continuously polls Cloud Storage for new files As soon as a file arrives, initiate the training job

C. Configure a Cloud Storage trigger to send a message to a Pub/Sub topic when a new file is available in a storage bucket. Use a Pub/Sub-triggered Cloud Function to start the training job on a GKE cluster

D. Use Cloud Scheduler to schedule jobs at a regular interval. For the first step of the job. check the timestamp of objects in your Cloud Storage bucket If there are no new files since the last run, abort the job.

Answer: C

Explanation:

This option is the best way to architect the workflow, as it allows you to use event-driven and serverless components to automate the ML training process. Cloud Storage triggers are a feature that allows you to send notifications to a Pub/Sub topic when an object is created, deleted, or updated in a storage bucket. Pub/Sub is a service that allows you to publish and subscribe to messages on various topics. Pub/Sub-triggered Cloud Functions are a type of Cloud Functions that are invoked when a message is published to a specific Pub/Sub topic. Cloud Functions are a serverless platform that allows you to run code in response to events. By using these components,

you can create a workflow that starts the training job on a GKE cluster as soon as a new file is available in the Cloud Storage bucket, without having to manage any servers or poll for changes. The other options are not as efficient or scalable as this option. Dataflow is a service that allows you to create and run data processing pipelines, but it is not designed to trigger ML training jobs on GKE. App Engine is a service that allows you to build and deploy web applications, but it is not suitable for polling Cloud Storage for new files, as it may incur unnecessary costs and latency. Cloud Scheduler is a service that allows you to schedule jobs at regular intervals, but it is not ideal for triggering ML training jobs based on data availability, as it may miss some files or run unnecessary jobs. Reference: Cloud Storage triggers documentation Pub/Sub documentation

Pub/Sub-triggered Cloud Functions documentation Cloud Functions documentation

Kubeflow Pipelines documentation

www.atmicnetworks.com

Question: 53

You are developing models to classify customer support emails. You created models with TensorFlow Estimators using small datasets on your on-premises system, but you now need to train the models using large datasets to ensure high performance. You will port your models to Google Cloud and want to minimize code refactoring and infrastructure overhead for easier migration from on-prem to cloud. What should you do?

A. Use Vertex AI Platform for distributed training
B. Create a cluster on Dataproc for training
C. Create a Managed Instance Group with autoscaling
D. Use Kubeflow Pipelines to train on a Google Kubernetes Engine cluster.

Answer: A

Explanation:
Vertex AI Platform is a unified platform for building and deploying ML models on Google Cloud. It supports both custom and AutoML models, and provides various tools and services for ML development, such as Vertex Pipelines, Vertex Vizier, Vertex Explainable AI, and Vertex Feature Store. Vertex AI Platform allows users to train their TensorFlow models using distributed training, which can speed up the training process and handle large datasets. Vertex AI Platform also minimizes code refactoring and infrastructure overhead, as it is compatible with TensorFlow Estimators and handles the provisioning, configuration, and scaling of the training resources automatically. The other options are not as suitable for this scenario. Dataproc is a service that allows users to create and run data processing pipelines using Apache Spark and Hadoop, but it is not designed for TensorFlow model training. Managed Instance Groups are a feature that allows users to create and manage groups of identical compute instances, but they require more configuration and management than Vertex AI Platform. Kubeflow Pipelines are a tool that allows users to create and run ML workflows on Google Kubernetes Engine, but they involve more complexity and code changes than Vertex AI Platform. Reference:
Vertex AI Platform documentation
Distributed training with Vertex AI Platform

Question: 54

You work for a large technology company that wants to modernize their contact center. You have been asked to develop a solution to classify incoming calls by product so that requests can be more quickly routed to the correct support team. You have already transcribed the calls using the Speech- to-Text API. You want to minimize data preprocessing and development time. How should you build the model?

A. Use the Al Platform Training built-in algorithms to create a custom model
B. Use AutoML Natural Language to extract custom entities for classification
C. Use the Cloud Natural Language API to extract custom entities for classification
D. Build a custom model to identify the product keywords from the transcribed calls, and then run the keywords through a classification algorithm

Answer: B

Explanation:
AutoML Natural Language is a service that allows users to create custom natural language models using their own data and labels. It supports various natural language tasks, such as text classification, entity extraction, and sentiment analysis. AutoML Natural Language can be used to build a model to classify incoming calls by product, as it can extract custom entities from the transcribed calls and assign them to predefined categories. AutoML Natural Language also

minimizes data preprocessing and development time, as it handles the data preparation, model training, and evaluation automatically. The other options are not as suitable for this scenario. AI Platform Training built-in algorithms are a set of pre-defined algorithms that can be used to train ML models on AI Platform, but they do not support natural language processing tasks. Cloud Natural Language API is a pretrained service that provides natural language understanding capabilities, such as sentiment analysis, entity analysis, syntax analysis, and content classification. However, it does not support custom entities or categories, and may not recognize the product names from the calls. Building a custom model to identify the product keywords and then running them through a classification algorithm would require more data preprocessing and development time, as well as more coding and testing. Reference:

AutoML Natural Language documentation

AI Platform Training built-in algorithms documentation Cloud Natural Language API documentation

Question: 55

You are an ML engineer at a regulated insurance company. You are asked to develop an insurance approval model that accepts or rejects insurance applications from potential customers. What factors should you consider before building the model?

A. Redaction, reproducibility, and explainability

B. Traceability, reproducibility, and explainability

C. Federated learning, reproducibility, and explainability

D. Differential privacy federated learning, and explainability

Answer: B

Explanation:

Before building an insurance approval model, an ML engineer should consider the factors of traceability, reproducibility, and explainability, as these are important aspects of responsible AI and fairness in a regulated domain. Traceability is the ability to track the provenance and lineage of the data, models, and decisions throughout the ML lifecycle. It helps to ensure the quality, reliability, and accountability of the ML system, and to comply with the regulatory and ethical standards.

Reproducibility is the ability to recreate the same results and outcomes using the same data, models, and parameters. It helps to verify the validity, consistency, and robustness of the ML system, and to debug and improve the performance. Explainability is the ability to understand and interpret the logic, behavior, and outcomes of the ML system. It helps to increase the transparency, trust, and confidence of the ML system, and to identify and mitigate any potential biases, errors, or risks. The other options are not as relevant or comprehensive as this option. Redaction is the process of removing sensitive or confidential information from the data or documents, but it is not a factor that the ML engineer should consider before building the model, as it is more related to the data preparation and protection. Federated learning is a technique that allows training ML models on decentralized data without transferring the data to a central server, but it is not a factor that the ML engineer should consider before building the model, as it is more related to the model architecture and privacy preservation. Differential privacy is a method that adds noise to the data or the model outputs to protect the individual privacy of the data subjects, but it is not a factor that the ML engineer should consider before building the model, as it is more related to the model evaluation and deployment. Reference:

Responsible AI documentation

Traceability documentation

Reproducibility documentation Explainability documentation

www.atmicnetworks.com

Question: 56

You work for a large hotel chain and have been asked to assist the marketing team in gathering predictions for a targeted marketing strategy. You need to make predictions about user lifetime value (LTV) over the next 30 days so that marketing can be adjusted accordingly. The customer dataset is in BigQuery, and you are preparing the tabular data for training with AutoML Tables. This data has a time signal that is spread across multiple columns. How should you ensure that AutoML fits the best model to your data?

A. Manually combine all columns that contain a time signal into an array Allow AutoML to interpret this array appropriately
Choose an automatic data split across the training, validation, and testing sets

B. Submit the data for training without performing any manual transformations Allow AutoML to handle the appropriate transformations Choose an automatic data split across the training, validation, and testing sets

C. Submit the data for training without performing any manual transformations, and indicate an appropriate column as the Time column Allow AutoML to split your data based on the time signal provided, and reserve the more recent data for the validation and testing sets

D. Submit the data for training without performing any manual transformations Use the columns that have a time signal to manually split your data Ensure that the data in your validation set is from 30 days after the data in your training set and that the data in your testing set is from 30 days after your validation set

Answer: C

Explanation:
This answer is correct because it allows AutoML Tables to handle the time signal in the data and split the data accordingly. This ensures that the model is trained on the historical data and evaluated on the more recent data, which is consistent with the prediction task. AutoML Tables can automatically detect and handle temporal features in the data, such as date, time, and duration. By specifying the Time column, AutoML Tables can also perform time-series forecasting and use the time signal to generate additional features, such as seasonality and trend. Reference:
[AutoML Tables: Preparing your training data]
[AutoML Tables: Time-series forecasting]

Question: 57

Your team is building an application for a global bank that will be used by millions of customers. You built a forecasting model that predicts customers1 account balances 3 days in the future. Your team will use the results in a new feature that will notify users when their account balance is likely to drop below $25. How should you serve your predictions?

A.
1 . Create a Pub/Sub topic for each user
2 Deploy a Cloud Function that sends a notification when your model predicts that a user's account balance will drop below the $25 threshold.

B.
1. Create a Pub/Sub topic for each user
2. Deploy an application on the App Engine standard environment that sends a notification when your model predicts that

a user's account balance will drop below the $25 threshold C.

1. Build a notification system on Firebase
2. Register each user with a user ID on the Firebase Cloud Messaging server, which sends a notification when the average of all account balance predictions drops below the $25 threshold D. 1 Build a notification system on Firebase
3. Register each user with a user ID on the Firebase Cloud Messaging server, which sends a notification when your model predicts that a user's account balance will drop below the $25 threshold

Answer: D

Explanation:

This answer is correct because it uses Firebase, a platform that provides a scalable and reliable notification system for mobile and web applications. Firebase Cloud Messaging (FCM) allows you to send messages and notifications to users across different devices and platforms. By registering each user with a user ID on the FCM server, you can target specific users based on their account balance predictions and send them personalized notifications when their balance is likely to drop below the $25 threshold. This way, you can provide a useful and timely feature for your customers and increase their engagement and retention. Reference: [Firebase Cloud Messaging]

[Firebase Cloud Messaging: Send messages to specific devices]

Question: 58

You have trained a text classification model in TensorFlow using Al Platform. You want to use the trained model for batch predictions on text data stored in BigQuery while minimizing computational overhead. What should you do?

A. Export the model to BigQuery ML.
B. Deploy and version the model on Al Platform.
C. Use Dataflow with the SavedModel to read the data from BigQuery
D. Submit a batch prediction job on Al Platform that points to the model location in Cloud Storage.

Answer: D

Explanation:

This answer is correct because it allows you to use the trained TensorFlow model for batch predictions on text data stored in BigQuery without any additional processing or overhead. Al Platform provides a service for running batch prediction jobs that can take input data from BigQuery or Cloud Storage and write the output to BigQuery or Cloud Storage. You can use the SavedModel format to export your TensorFlow model to Cloud Storage and then submit a batch prediction job that points to the model location and the input data location. Al Platform will handle the scaling and distribution of the prediction requests and return the results in the specified output location. Reference:

[Al Platform: Batch prediction overview]

[Al Platform: Exporting a SavedModel for prediction]

Question: 59

Your organization wants to make its internal shuttle service route more efficient. The shuttles currently stop at all pick-up points across the city every 30 minutes between 7 am and 10 am. The development team has already built an application on Google Kubernetes Engine that requires users to confirm their presence and shuttle station one day in advance. What approach should you take?

A. 1. Build a tree-based regression model that predicts how many passengers will be picked up at each shuttle

station.

2. Dispatch an appropriately sized shuttle and provide the map with the required stops based on the prediction.

B. 1. Build a tree-based classification model that predicts whether the shuttle should pick up passengers at each shuttle station.

2. Dispatch an available shuttle and provide the map with the required stops based on the prediction

C. 1. Define the optimal route as the shortest route that passes by all shuttle stations with confirmed attendance at the given time under capacity constraints.

2 Dispatch an appropriately sized shuttle and indicate the required stops on the map

D. 1. Build a reinforcement learning model with tree-based classification models that predict the presence of passengers at shuttle stops as agents and a reward function around a distance-based metric

2. Dispatch an appropriately sized shuttle and provide the map with the required stops based on the simulated outcome.

Answer: A

Explanation:

This answer is correct because it uses a regression model to estimate the number of passengers at each shuttle station, which is a continuous variable. A tree-based regression model can handle both numerical and categorical features, such as the time of day, the location of the station, and the weather conditions. Based on the predicted number of passengers, the organization can dispatch a shuttle that has enough capacity and provide a map that shows the required stops. This way, the organization can optimize the shuttle service route and reduce the waiting time and fuel consumption. Reference: [Tree-based regression models]

Question: 60

You need to build classification workflows over several structured datasets currently stored in BigQuery. Because you will be performing the classification several times, you want to complete the following steps without writing code: exploratory data analysis, feature selection, model building, training, and hyperparameter tuning and serving. What should you do?

A. Configure AutoML Tables to perform the classification task

B. Run a BigQuery ML task to perform logistic regression for the classification

C. Use Al Platform Notebooks to run the classification model with pandas library

D. Use Al Platform to run the classification model job configured for hyperparameter tuning

Answer: A

Explanation:

AutoML Tables is a service that allows you to automatically build and deploy state-of-the-art machine learning models on structured data without writing code. You can use AutoML Tables to perform the following steps for the classification task:

Exploratory data analysis: AutoML Tables provides a graphical user interface (GUI) and a commandline interface (CLI) to explore your data, visualize statistics, and identify potential issues.

Feature selection: AutoML Tables automatically selects the most relevant features for your model based on the data schema and the target column. You can also manually exclude or include features, or create new features from existing ones using feature engineering.

Model building: AutoML Tables automatically builds and evaluates multiple machine learning models using different algorithms and architectures. You can also specify the optimization objective, the budget, and the evaluation metric for your model.

Training and hyperparameter tuning: AutoML Tables automatically trains and tunes your model using the best practices and techniques from Google's research and engineering teams. You can monitor the training progress and the performance of your model on the GUI or the CLI.

Serving: AutoML Tables automatically deploys your model to a fully managed, scalable, and secure environment. You can use the GUI or the CLI to request predictions from your model, either online (synchronously) or offline (asynchronously).

Reference:

[AutoML Tables documentation]

[AutoML Tables overview]

[AutoML Tables how-to guides]

Question: 61

You recently joined an enterprise-scale company that has thousands of datasets. You know that there are accurate descriptions for each table in BigQuery, and you are searching for the proper BigQuery table to use for a model you are building on AI Platform. How should you find the data that you need?

A. Use Data Catalog to search the BigQuery datasets by using keywords in the table description.

B. Tag each of your model and version resources on AI Platform with the name of the BigQuery table that was used for training.

C. Maintain a lookup table in BigQuery that maps the table descriptions to the table ID. Query the lookup table to find the correct table ID for the data that you need.

D. Execute a query in BigQuery to retrieve all the existing table names in your project using the INFORMATION_SCHEMA metadata tables that are native to BigQuery. Use the result o find the table that you need.

Answer: A

Explanation:

Data Catalog is a fully managed and scalable metadata management service that allows you to quickly discover, manage, and understand your data in Google Cloud. You can use Data Catalog to search the BigQuery datasets by using keywords in the table description, as well as other metadata attributes such as table name, column name, labels, tags, and more. Data Catalog also provides a rich browsing experience that lets you explore the schema, preview the data, and access the BigQuery console directly from the Data Catalog UI. Data Catalog helps you find the data that you need for your model building on AI Platform without writing any code or queries.

Reference:

[Data Catalog documentation]

[Data Catalog overview]

[Searching for data assets]

Question: 62

You are working on a classification problem with time series data and achieved an area under the receiver operating characteristic curve (AUC ROC) value of 99% for training data after just a few experiments. You haven't explored using any sophisticated algorithms or spent any time on hyperparameter tuning. What should your next step be to identify and fix the problem?

A. Address the model overfitting by using a less complex algorithm.
B. Address data leakage by applying nested cross-validation during model training.
C. Address data leakage by removing features highly correlated with the target value.
D. Address the model overfitting by tuning the hyperparameters to reduce the AUC ROC value.

Answer: B

Explanation:
Data leakage is a problem where information from outside the training dataset is used to create the model, resulting in an overly optimistic or invalid estimate of the model performance. Data leakage can occur in time series data when the temporal order of the data is not preserved during data preparation or model evaluation. For example, if the data is shuffled before splitting into train and test sets, or if future data is used to impute missing values in past data, then data leakage can occur. One way to address data leakage in time series data is to apply nested cross-validation during model training. Nested cross-validation is a technique that allows you to perform both model selection and model evaluation in a robust way, while preserving the temporal order of the data. Nested crossvalidation involves two levels of cross-validation: an inner loop for model selection and an outer loop for model evaluation. The inner loop splits the training data into k folds, trains and tunes the model on k-1 folds, and validates the model on the remaining fold. The inner loop repeats this process for each fold and selects the best model based on the validation performance. The outer loop splits the data into n folds, trains the best model from the inner loop on n-1 folds, and tests the model on the remaining fold. The outer loop repeats this process for each fold and evaluates the model performance based on the test results.
Nested cross-validation can help to avoid data leakage in time series data by ensuring that the model is trained and tested on non-overlapping data, and that the data used for validation is never seen by the model during training. Nested cross-validation can also provide a more reliable estimate of the model performance than a single train-test split or a simple cross-validation, as it reduces the variance and bias of the estimate.
Reference:
Data Leakage in Machine Learning
How to Avoid Data Leakage When Performing Data Preparation
Classification on a single time series - prevent leakage between train and test

Question: 63

You work for an online travel agency that also sells advertising placements on its website to other companies.

You have been asked to predict the most relevant web banner that a user should see next. Security is important to your company. The model latency requirements are 300ms@p99, the inventory is thousands of web banners, and your exploratory analysis has shown that navigation context is a good predictor. You want to Implement the simplest solution. How should you configure the prediction pipeline?

A. Embed the client on the website, and then deploy the model on AI Platform Prediction.
B. Embed the client on the website, deploy the gateway on App Engine, and then deploy the model on AI Platform Prediction.
C. Embed the client on the website, deploy the gateway on App Engine, deploy the database on Cloud Bigtable for writing and for reading the user's navigation context, and then deploy the model on AI Platform Prediction.
D. Embed the client on the website, deploy the gateway on App Engine, deploy the database on Memorystore for writing and for reading the user's navigation context, and then deploy the model on Google Kubernetes Engine.

Answer: A

Explanation:

In this scenario, the goal is to predict the most relevant web banner that a user should see next on an online travel agency's website. The model needs to have low latency requirements of 300ms@p99, and there are thousands of web banners to choose from. The exploratory analysis has shown that the navigation context is a good predictor. Security is also important to the company. Given these requirements, the best configuration for the prediction pipeline would be to embed the client on the website and deploy the model on AI Platform Prediction. Option A is the correct answer.

Option A: Embed the client on the website, and then deploy the model on AI Platform Prediction. This option is the simplest solution that meets the requirements. The client can collect the user's navigation context and send it to the model deployed on AI Platform Prediction for prediction. AI Platform Prediction can handle large-scale prediction requests and has low latency requirements. This option does not require any additional infrastructure or services, making it the simplest solution. Option B: Embed the client on the website, deploy the gateway on App Engine, and then deploy the model on AI Platform Prediction. This option adds an additional layer of infrastructure by deploying the gateway on App Engine. While App Engine can handle large-scale requests, it adds complexity to the pipeline and may not be necessary for this use case.

Option C: Embed the client on the website, deploy the gateway on App Engine, deploy the database on Cloud Bigtable for writing and for reading the user's navigation context, and then deploy the model on AI Platform Prediction. This option adds even more complexity to the pipeline by deploying the database on Cloud Bigtable. While Cloud Bigtable can provide fast and scalable access to the user's navigation context, it may not be needed for this use case. Moreover, Cloud Bigtable may introduce additional latency and cost to the pipeline.

Option D: Embed the client on the website, deploy the gateway on App Engine, deploy the database on Memorystore for writing and for reading the user's navigation context, and then deploy the model on Google Kubernetes Engine. This option is the most complex and costly solution that does not meet the requirements. Deploying the model on Google Kubernetes Engine requires more management and configuration than AI Platform Prediction. Moreover, Google Kubernetes Engine may not be able to meet the low latency requirements of 300ms@p99. Deploying the database on Memorystore also adds unnecessary overhead and cost to the pipeline.

Reference:

AI Platform Prediction documentation
App Engine documentation
Cloud Bigtable documentation [Memorystore documentation] [Google Kubernetes Engine documentation]

Question: 64

Your team is building a convolutional neural network (CNN)-based architecture from scratch. The preliminary experiments running on your on-premises CPU-only infrastructure were encouraging, but have slow convergence. You have been asked to speed up model training to reduce time-to- market. You want to experiment with virtual machines (VMs) on Google Cloud to leverage more powerful hardware. Your code does not include any manual device placement and has not been wrapped in Estimator model-level abstraction. Which environment should you train your model on?

A. AVM on Compute Engine and 1 TPU with all dependencies installed manually.
B. AVM on Compute Engine and 8 GPUs with all dependencies installed manually.
C. A Deep Learning VM with an n1-standard-2 machine and 1 GPU with all libraries pre-installed.
D. A Deep Learning VM with more powerful CPU e2-highcpu-16 machines with all libraries preinstalled.

Answer: C

Explanation:

In this scenario, the goal is to speed up model training for a CNN-based architecture on Google Cloud. The code does not include any manual device placement and has not been wrapped in Estimator model-level abstraction. Given these constraints, the best environment to train the model on would be a Deep Learning VM with an n1-standard-2 machine and 1 GPU with all libraries preinstalled. Option C is the correct answer.

Option C: A Deep Learning VM with an n1-standard-2 machine and 1 GPU with all libraries preinstalled. This option is the most suitable for the scenario because it provides a ready-to-use environment for deep learning on Google Cloud. A Deep Learning VM is a specialized VM image that is pre-installed with popular deep learning frameworks such as TensorFlow, PyTorch, Keras, and more. A Deep Learning VM also comes with NVIDIA GPU drivers and CUDA libraries that enable GPU acceleration for model training. A Deep Learning VM can be easily configured and launched from the Google Cloud Console or the Cloud SDK. An n1-standard-2 machine is a general-purpose machine type that provides 2 vCPUs and 7.5 GB of memory. This machine type can be sufficient for running a CNN-based architecture. A GPU is a specialized hardware accelerator that can speed up the computation of matrix operations and convolutions, which are common in CNN-based architectures. By using a Deep Learning VM with an n1-standard-2 machine and 1 GPU, the model training can be significantly faster than on an on-premises CPU-only infrastructure.

Option A: A VM on Compute Engine and 1 TPU with all dependencies installed manually. This option is not suitable for the scenario because it requires manual installation of dependencies and device placement. A TPU is a custom-designed ASIC that can provide high performance and efficiency for TensorFlow models. However, to use a TPU, the code needs to include manual device placement and be wrapped in Estimator model-level abstraction. Moreover, to use a TPU, the dependencies such as TensorFlow, Cloud TPU Client, and Cloud Storage need to be installed manually on the VM. This option can be complex and time-consuming to set up and may not be compatible with the existing code.

Option B: A VM on Compute Engine and 8 GPUs with all dependencies installed manually. This option is not suitable for the scenario because it requires manual installation of dependencies and may not be cost-effective. While using 8 GPUs can provide high parallelism and speed for model training, it also increases the cost and complexity of the environment. Moreover, to use GPUs, the dependencies such as NVIDIA GPU drivers, CUDA libraries, and deep learning frameworks need to be installed manually on the VM. This option can be tedious and error-prone to set up and may not be necessary for the scenario.

Option D: A Deep Learning VM with more powerful CPU e2-highcpu-16 machines with all libraries pre-installed. This option is not suitable for the scenario because it does not leverage GPU acceleration for model training. While using more powerful CPU machines can provide more compute resources and memory for model training, it may not be as fast and efficient as using GPU machines. CPU machines are not optimized for matrix operations and convolutions, which are common in CNN-based architectures. Moreover, using more powerful CPU machines can also increase the cost of the environment. This option can be suboptimal and wasteful for the scenario. Reference:

Deep Learning VM Image documentation
Compute Engine documentation
Cloud TPU documentation
Machine types documentation
GPUs on Compute Engine documentation

Question: 65

You work on a growing team of more than 50 data scientists who all use AI Platform. You are designing a strategy to organize your jobs, models, and versions in a clean and scalable way. Which strategy should you choose?

A. Set up restrictive IAM permissions on the AI Platform notebooks so that only a single user or group can access a given instance.

B. Separate each data scientist's work into a different project to ensure that the jobs, models, and versions created by each data scientist are accessible only to that user.

C. Use labels to organize resources into descriptive categories. Apply a label to each created resource so that users can filter the results by label when viewing or monitoring the resources.

D. Set up a BigQuery sink for Cloud Logging logs that is appropriately filtered to capture information about AI Platform resource usage. In BigQuery, create a SQL view that maps users to the resources they are using

Answer: C

Explanation:

Labels are key-value pairs that you can attach to AI Platform resources such as jobs, models, and versions. Labels can help you organize your resources into descriptive categories that reflect your business needs. For example, you can use labels to indicate the owner, purpose, environment, or status of a resource. You can also use labels to filter the results when you list or monitor your resources on the Google Cloud Console or the Cloud SDK. Using labels can help you manage your resources in a clean and scalable way, without requiring separate projects or restrictive permissions. Reference:

Using labels to organize AI Platform resources

Creating and managing labels

Question: 66

You work for a credit card company and have been asked to create a custom fraud detection model based on historical data using AutoML Tables. You need to prioritize detection of fraudulent transactions while minimizing false positives. Which optimization objective should you use when training the model?

A. An optimization objective that minimizes Log loss

B. An optimization objective that maximizes the Precision at a Recall value of 0.50

C. An optimization objective that maximizes the area under the precision-recall curve (AUC PR) value D. An optimization objective that maximizes the area under the receiver operating characteristic curve (AUC ROC) value

Answer: C

Explanation:

In this scenario, the goal is to create a custom fraud detection model using AutoML Tables. Fraud detection is a type of binary classification problem, where the model needs to predict whether a transaction is fraudulent or not. The optimization objective is a metric that defines how the model is trained and evaluated. AutoML Tables allows you to choose from different optimization objectives for binary classification problems, such as Log loss, Precision at a Recall value, AUC PR, and AUC ROC. To choose the best optimization objective for fraud detection, we need to consider the characteristics of the problem and the data. Fraud detection is a problem where the positive class (fraudulent transactions) is very rare compared to the negative class (legitimate transactions). This means that the data is highly imbalanced, and the model needs to be sensitive to the minority class. Moreover, fraud detection is a problem where the cost of false negatives (missing a fraudulent transaction) is much higher than the cost of false positives (flagging a legitimate transaction as fraudulent). This means that the model needs to have high recall (the ability to detect all fraudulent transactions) while maintaining high precision (the ability to avoid false alarms).

Given these considerations, the best optimization objective for fraud detection is the one that maximizes the area

under the precision-recall curve (AUC PR) value. The AUC PR value is a metric that measures the trade-off between precision and recall for different probability thresholds. A higher AUC PR value means that the model can achieve high precision and high recall at the same time. The AUC PR value is also more suitable for imbalanced data than the AUC ROC value, which measures the trade-off between the true positive rate and the false positive rate. The AUC ROC value can be misleading for imbalanced data, as it can give a high score even if the model has low recall or low precision. Therefore, option C is the correct answer. Option A is not suitable, as Log loss is a metric that measures the difference between the predicted probabilities and the actual labels, and does not account for the trade-off between precision and recall. Option B is not suitable, as Precision at a

Recall value is a metric that measures the precision at a fixed recall level, and does not account for the trade-off between precision and recall at different thresholds. Option D is not suitable, as AUC ROC is a metric that can be misleading for imbalanced data, as explained above.

Reference:

AutoML Tables documentation

Optimization objectives for binary classification

Precision-Recall Curves: How to Easily Evaluate Machine Learning Models in No Time ROC Curves and Area Under the Curve Explained (video)

Question: 67

Your company manages a video sharing website where users can watch and upload videos. You need to create an ML model to predict which newly uploaded videos will be the most popular so that those videos can be prioritized on your company's website. Which result should you use to determine whether the model is successful?

A. The model predicts videos as popular if the user who uploads them has over 10,000 likes.
B. The model predicts 97.5% of the most popular clickbait videos measured by number of clicks.
C. The model predicts 95% of the most popular videos measured by watch time within 30 days of being uploaded.
D. The Pearson correlation coefficient between the log-transformed number of views after 7 days and 30 days after publication is equal to 0.

Answer: C

Explanation:

In this scenario, the goal is to create an ML model to predict which newly uploaded videos will be the most popular on a video sharing website. The result that should be used to determine whether the model is successful is the one that best aligns with the business objective and the evaluation metric. Option C is the correct answer because it defines the most popular videos as the ones that have the highest watch time within 30 days of being uploaded, and it sets a high accuracy threshold of 95% for the model prediction.

Option C: The model predicts 95% of the most popular videos measured by watch time within 30 days of being uploaded. This option is the best result for the scenario because it reflects the business objective and the evaluation metric. The business objective is to prioritize the videos that will attract and retain the most viewers on the website. The watch time is a good indicator of the viewer engagement and satisfaction, as it measures how long the viewers watch the videos. The 30-day window is a reasonable time frame to capture the popularity trend of the videos, as it accounts for the initial interest and the viral potential of the videos. The 95% accuracy threshold is a high standard for

the model prediction, as it means that the model can correctly identify 95 out of 100 of the most popular videos based on the watch time metric.

Option A: The model predicts videos as popular if the user who uploads them has over 10,000 likes. This option is not a good result for the scenario because it does not reflect the business objective or the evaluation metric. The business objective is to prioritize the videos that will be the most popular on the website, not the users who upload them. The number of likes that a user has is not a good indicator of the popularity of their videos, as it does not measure the viewer engagement or satisfaction with the videos. Moreover, this option does not specify a time frame or an accuracy threshold for the model prediction, making it vague and unreliable.

Option B: The model predicts 97.5% of the most popular clickbait videos measured by number of clicks. This option is not a good result for the scenario because it does not reflect the business objective or the evaluation metric. The business objective is to prioritize the videos that will be the most popular on the website, not the videos that have the most misleading or sensational titles or thumbnails. The number of clicks that a video has is not a good indicator of the popularity of the video, as it does not measure the viewer engagement or satisfaction with the video content. Moreover, this option only focuses on the clickbait videos, which may not represent the majority or the diversity of the videos on the website.

Option D: The Pearson correlation coefficient between the log-transformed number of views after 7 days and 30 days after publication is equal to 0. This option is not a good result for the scenario because it does not reflect the business objective or the evaluation metric. The business objective is to prioritize the videos that will be the most popular on the website, not the videos that have the most consistent or inconsistent number of views over time. The Pearson correlation coefficient is a metric that measures the linear relationship between two variables, not the popularity of the videos. A correlation coefficient of 0 means that there is no linear relationship between the log-transformed number of views after 7 days and 30 days, which does not indicate whether the videos are popular or not. Moreover, this option does not specify a threshold or a target value for the correlation coefficient, making it meaningless and irrelevant.

Question: 68

You are working on a Neural Network-based project. The dataset provided to you has columns with different ranges. While preparing the data for model training, you discover that gradient optimization is having difficulty moving weights to a good solution. What should you do?

A. Use feature construction to combine the strongest features.
B. Use the representation transformation (normalization) technique.
C. Improve the data cleaning step by removing features with missing values.
D. Change the partitioning step to reduce the dimension of the test set and have a larger training set.

Answer: B

Explanation:

Representation transformation (normalization) is a technique that transforms the features to be on a similar scale, such as between 0 and 1, or with mean 0 and standard deviation 1. This technique can improve the performance and training stability of the neural network model, as it can prevent the gradient optimization from being dominated by features with larger scales, and help the model converge faster and better. There are different types of normalization techniques, such as min-max scaling, z-score scaling, log scaling, etc. You can learn more about normalization techniques from the following web search results:

Normalization | Machine Learning | Google for Developers

NORMALIZATION TECHNIQUES IN TRAINING DNNS: METHODOLOGY, ANALYSIS AND ...

Visualizing Different Normalization Techniques | by Dibya ... - Medium Data Normalization Techniques: Easy to Advanced (& the Best)

Question: 69

You work for a bank and are building a random forest model for fraud detection. You have a dataset that includes transactions, of which 1% are identified as fraudulent. Which data transformation strategy would likely improve the performance of your classifier?

A. Write your data in TFRecords.
B. Z-normalize all the numeric features.
C. Oversample the fraudulent transaction 10 times.
D. Use one-hot encoding on all categorical features.

Answer: C

Explanation:
Oversampling is a technique for dealing with imbalanced datasets, where the majority class dominates the minority class. It balances the distribution of classes by increasing the number of samples in the minority class. Oversampling can improve the performance of a classifier by reducing the bias towards the majority class and increasing the sensitivity to the minority class.
In this case, the dataset includes transactions, of which 1% are identified as fraudulent. This means that the fraudulent transactions are the minority class and the non-fraudulent transactions are the majority class. A random forest model trained on this dataset might have a low recall for the fraudulent transactions, meaning that it might miss many of them and fail to detect fraud. This could have a high cost for the bank and its customers.
One way to overcome this problem is to oversample the fraudulent transactions 10 times, meaning that each fraudulent transaction is duplicated 10 times in the training dataset. This would increase the proportion of fraudulent transactions from 1% to about 10%, making the dataset more balanced. This would also make the random forest model more aware of the patterns and features that distinguish fraudulent transactions from non-fraudulent ones, and thus improve its accuracy and recall for the minority class.
For more information about oversampling and other techniques for imbalanced data, see the following references:
Random Oversampling and Undersampling for Imbalanced Classification
Exploring Oversampling Techniques for Imbalanced Datasets

Question: 70

You are developing an ML model intended to classify whether X-Ray images indicate bone fracture risk. You have trained on Api Resnet architecture on Vertex AI using a TPU as an accelerator, however

you are unsatisfied with the trainning time and use memory usage. You want to quickly iterate your training code but make minimal changes to the code. You also want to minimize impact on the models accuracy. What should you do?

A. Configure your model to use bfloat16 instead float32
B. Reduce the global batch size from 1024 to 256
C. Reduce the number of layers in the model architecture
D. Reduce the dimensions of the images used un the model

Answer: A

Explanation:
Using bfloat16 instead of float32 can reduce the memory usage and training time of the model, while having minimal impact on the accuracy. Bfloat16 is a 16-bit floating-point format that preserves the range of 32-bit floating-point numbers, but reduces the precision from 24 bits to 8 bits. This means that bfloat16 can store the same magnitude of numbers as float32, but with less detail. Bfloat16 is supported by TPUs and some GPUs, and can be used as a drop-in replacement for float32 in most cases. Bfloat16 can also improve the numerical stability of the model, as it reduces the risk of overflow and underflow errors.
Reducing the global batch size, the number of layers, or the dimensions of the images can also reduce the memory usage and training time of the model, but they can also affect the model's accuracy and performance. Reducing the global batch size can make the model less stable and converge slower, as it reduces the amount of information available for each gradient update. Reducing the number of layers can make the model less expressive and powerful, as it reduces the depth and complexity of the network. Reducing the dimensions of the images can make the model less accurate and robust, as it reduces the resolution and quality of the input data. Reference: Bfloat16: The secret to high performance on Cloud TPUs
Bfloat16 floating-point format
How does Batch Size impact your model learning

Question: 71

Your task is classify if a company logo is present on an image. You found out that 96% of a data does not include a logo. You are dealing with data imbalance problem. Which metric do you use to evaluate to model?

A. F1 Score
B. RMSE
C. F Score with higher precision weighting than recall
D. F Score with higher recall weighted than precision

Answer: A

Explanation:
The F1 score is a metric that combines both precision and recall, and is suitable for evaluating imbalanced classification problems. Precision measures the fraction of true positives among the predicted positives, and recall measures the fraction of true positives among the actual positives. The F1 score is the harmonic mean of precision and recall, and it ranges from 0 to 1, with higher values indicating better performance. The F1 score is a good metric for imbalanced data because it balances both the false positives and the false negatives, and does not favor the majority class over the minority class.
The other options are not good metrics for imbalanced data. RMSE (root mean squared error) is a metric for regression problems, not classification problems. It measures the average squared difference between the predicted and the

actual values, and is not suitable for binary outcomes. F score with higher precision weighting than recall, or F0.5 score, is a metric that gives more importance to precision than recall. This means that it penalizes false positives more than false negatives, which is not desirable for imbalanced data where the minority class is more important. F score with higher recall weighting than precision, or F2 score, is a metric that gives more importance to recall than precision. This means that it penalizes false negatives more than false positives, which might be suitable for some imbalanced data problems, but not for the logo detection problem. In this problem, both false positives and false negatives are equally important, as we want to accurately identify the presence or absence of a logo in an image. Therefore, the F1 score is a better metric than the F2 score. Reference:

Tour of Evaluation Metrics for Imbalanced Classification Metrics for imbalanced data (simply explained)

Question: 72

You need to train a regression model based on a dataset containing 50,000 records that is stored in BigQuery. The data includes a total of 20 categorical and numerical features with a target variable that can include negative values. You need to minimize effort and training time while maximizing model performance. What approach should you take to train this regression model?

A. Create a custom TensorFlow DNN model.
B. Use BQML XGBoost regression to train the model
C. Use AutoML Tables to train the model without early stopping.
D. Use AutoML Tables to train the model with RMSLE as the optimization objective

Answer: D

Explanation:
AutoML Tables is a service that allows you to automatically build, analyze, and deploy machine learning models on tabular data. It is suitable for large-scale regression and classification problems, and it supports various optimization objectives, data splitting methods, and hyperparameter tuning algorithms. AutoML Tables can handle both categorical and numerical features, and it can also handle missing values and outliers. AutoML Tables is a good choice for this problem because it minimizes the effort and training time required to train a regression model, while maximizing the model performance.

RMSLE stands for Root Mean Squared Logarithmic Error, and it is a metric that measures the average difference between the logarithm of the predicted values and the logarithm of the actual values. RMSLE is useful for regression problems where the target variable can include negative values, and where large differences between small values are more important than large differences between large values. For example, RMSLE penalizes underestimating a value of 10 by 2 more than overestimating a value of 1000 by 20. RMSLE is a good optimization objective for this problem because it can handle negative values in the target variable, and it can reduce the impact of outliers and large errors. For more information about AutoML Tables and RMSLE, see the following references: AutoML Tables: end-to-end workflows on AI Platform Pipelines
Predict workload failures before they happen with AutoML Tables How to Calculate RMSE in R

Question: 73

Your data science team has requested a system that supports scheduled model retraining, Docker containers, and a service that supports autoscaling and monitoring for online prediction requests. Which platform components should

you choose for this system?

A. Vertex AI Pipelines and App Engine
B. Vertex AI Pipelines and Al Platform Prediction
C. Cloud Composer, BigQuery ML , and Al Platform Prediction
D. Cloud Composer, Al Platform Training with custom containers, and App Engine

Answer: B

Explanation:
Vertex AI Pipelines and AI Platform Prediction are the platform components that best suit the requirements of the data science team. Vertex AI Pipelines is a service that allows you to orchestrate and automate your machine learning workflows using pipelines. Pipelines are portable and scalable ML workflows that are based on containers. You can use Vertex AI Pipelines to schedule model retraining, use custom containers, and integrate with other Google Cloud services. AI Platform Prediction is a service that allows you to host your trained models and serve online predictions. You can use AI Platform Prediction to deploy models trained on Vertex AI or elsewhere, and benefit from features such as autoscaling, monitoring, logging, and explainability. Reference:
Vertex AI Pipelines AI Platform Prediction

Question: 74

While monitoring your model training's GPU utilization, you discover that you have a native synchronous implementation. The training data is split into multiple files. You want to reduce the execution time of your input pipeline. What should you do?

A. Increase the CPU load

B. Add caching to the pipeline
C. Increase the network bandwidth
D. Add parallel interleave to the pipeline

Answer: D

Explanation:
Parallel interleave is a technique that can improve the performance of the input pipeline by reading and processing data from multiple files in parallel. This can reduce the idle time of the GPU and speed up the training process. Parallel interleave can be implemented using the tf.data.experimental.parallel_interleave () function in TensorFlow, which takes a map function that returns a dataset for each input element, and a cycle length that determines how many input elements are processed concurrently. Parallel interleave can also handle different file sizes and processing times by using a block length argument that controls how many consecutive elements are produced from each input element before switching to another input element. For more information about parallel interleave and how to use it, see the following references:
How to use parallel_interleave in TensorFlow Better performance with the tf.data API

Question: 75

Your data science team is training a PyTorch model for image classification based on a pre-trained RestNet model. You

need to perform hyperparameter tuning to optimize for several parameters. What should you do?

A. Convert the model to a Keras model, and run a Keras Tuner job.
B. Run a hyperparameter tuning job on AI Platform using custom containers.
C. Create a Kuberflow Pipelines instance, and run a hyperparameter tuning job on Katib.
D. Convert the model to a TensorFlow model, and run a hyperparameter tuning job on AI Platform.

Answer: B

Explanation:
AI Platform supports hyperparameter tuning for PyTorch models using custom containers. This allows you to use any Python dependencies and libraries that are not included in the pre-built AI Platform Training runtime versions. You can also use a pre-trained model such as ResNet as a base for your custom model. To run a hyperparameter tuning job on AI Platform using custom containers, you need to do the following steps:
Create a Dockerfile that defines the container image for your training application. The Dockerfile should install PyTorch and any other dependencies, copy your training code and configuration files, and set the entrypoint for the container.
Build the container image and push it to Container Registry or another accessible registry.
Create a YAML file that defines the configuration for your hyperparameter tuning job. The YAML file should specify the container image URI, the training input and output paths, the hyperparameters to tune, the metric to optimize, and the tuning algorithm and budget.
Submit the hyperparameter tuning job to AI Platform using the gcloud command-line tool or the AI Platform Training API.
Reference:
Hyperparameter tuning overview
Using custom containers
PyTorch on AI Platform Training

Question: 76

You have a large corpus of written support cases that can be classified into 3 separate categories: Technical Support, Billing Support, or Other Issues. You need to quickly build, test, and deploy a service that will automatically classify future written requests into one of the categories. How should you configure the pipeline?

A. Use the Cloud Natural Language API to obtain metadata to classify the incoming cases.
B. Use AutoML Natural Language to build and test a classifier. Deploy the model as a REST API.
C. Use BigQuery ML to build and test a logistic regression model to classify incoming requests. Use BigQuery ML to perform inference.
D. Create a TensorFlow model using Google's BERT pre-trained model. Build and test a classifier, and deploy the model using Vertex AI.

Answer: B

Explanation:
AutoML Natural Language is a service that allows you to quickly build, test and deploy natural language processing (NLP) models without needing to have expertise in NLP or machine learning. You can use it to train a classifier on your corpus of written support cases, and then use the AutoML API to perform classification on new requests. Once the model is

trained, it can be deployed as a REST API. This allows the classifier to be integrated into your pipeline and be easily consumed by other systems.

Question: 77

You need to quickly build and train a model to predict the sentiment of customer reviews with custom categories without writing code. You do not have enough data to train a model from scratch. The resulting model should have high predictive performance. Which service should you use?

A. AutoML Natural Language
B. Cloud Natural Language API
C. AI Hub pre-made Jupyter Notebooks
D. AI Platform Training built-in algorithms

Answer: A

Explanation:
AutoML Natural Language is a service that allows you to build and train custom natural language models without writing code. You can use AutoML Natural Language to perform sentiment analysis
with custom categories, such as positive, negative, or neutral. You can also use pre-trained models or transfer learning to leverage existing knowledge and reduce the amount of data required to train a model from scratch. AutoML Natural Language provides a user-friendly interface and a powerful AutoML engine that optimizes your model for high predictive performance.
Cloud Natural Language API is a service that provides pre-trained models for common natural language tasks, such as sentiment analysis, entity analysis, and syntax analysis. However, it does not allow you to customize the categories or use your own data for training.
AI Hub pre-made Jupyter Notebooks are interactive documents that contain code, text, and visualizations for various machine learning scenarios. However, they require some coding skills and data preparation to use them effectively.
AI Platform Training built-in algorithms are pre-configured machine learning algorithms that you can use to train models on AI Platform. However, they do not support sentiment analysis as a natural language task.
Reference:
AutoML Natural Language documentation
Cloud Natural Language API documentation
AI Hub documentation
AI Platform Training documentation

Question: 78

You need to build an ML model for a social media application to predict whether a user's submitted profile photo meets the requirements. The application will inform the user if the picture meets the requirements. How should you build a model to ensure that the application does not falsely accept a non-compliant picture?

A. Use AutoML to optimize the model's recall in order to minimize false negatives.
B. Use AutoML to optimize the model's F1 score in order to balance the accuracy of false positives and false negatives.
C. Use Vertex AI Workbench user-managed notebooks to build a custom model that has three times as many examples

of pictures that meet the profile photo requirements.

D. Use Vertex AI Workbench user-managed notebooks to build a custom model that has three times as many examples of pictures that do not meet the profile photo requirements.

Answer: A

Explanation:

Recall is the ratio of true positives to the sum of true positives and false negatives. It measures how well the model can identify all the relevant cases. In this scenario, the relevant cases are the pictures that do not meet the profile photo requirements. Therefore, minimizing false negatives means minimizing the cases where the model incorrectly predicts that a non-compliant picture meets the requirements. By using AutoML to optimize the model's recall, the model will be more likely to reject a non-compliant picture and inform the user accordingly. Reference:

[AutoML Vision] is a service that allows you to train custom ML models for image classification and object detection tasks. You can use AutoML to optimize your model for different metrics, such as recall, precision, or F1 score.

[Recall] is one of the evaluation metrics for ML models. It is defined as TP / (TP + FN), where TP is the number of true positives and FN is the number of false negatives. Recall measures how well the model can identify all the relevant cases. A high recall means that the model has a low rate of false negatives.

Question: 79

You lead a data science team at a large international corporation. Most of the models your team trains are large-scale models using high-level TensorFlow APIs on AI Platform with GPUs. Your team usually takes a few weeks or months to iterate on a new version of a model. You were recently asked to review your team's spending. How should you reduce your Google Cloud compute costs without impacting the model's performance?

A. Use AI Platform to run distributed training jobs with checkpoints.

B. Use AI Platform to run distributed training jobs without checkpoints.

C. Migrate to training with Kuberflow on Google Kubernetes Engine, and use preemptible VMs with checkpoints.

D. Migrate to training with Kuberflow on Google Kubernetes Engine, and use preemptible VMs without checkpoints.

Answer: C

Explanation:

Option A is incorrect because using AI Platform to run distributed training jobs with checkpoints does not reduce the compute costs, but rather increases them by using more resources and storing the checkpoints.

Option B is incorrect because using AI Platform to run distributed training jobs without checkpoints may reduce the compute costs, but it also risks losing the progress of the training if the job fails or is interrupted.

Option C is correct because migrating to training with Kubeflow on Google Kubernetes Engine, and using preemptible VMs with checkpoints can reduce the compute costs significantly by using cheaper and more scalable resources, while also preserving the state of the training with checkpoints.

Option D is incorrect because using preemptible VMs without checkpoints may reduce the compute costs, but it also risks losing the training progress if the VMs are preempted.

Reference:

Kubeflow on Google Cloud

Using preemptible VMs and GPUs

Saving and loading models

Question: 80

You have deployed a model on Vertex AI for real-time inference. During an online prediction request, you get an "Out of Memory" error. What should you do?

A. Use batch prediction mode instead of online mode.
B. Send the request again with a smaller batch of instances.
C. Use base64 to encode your data before using it for prediction.
D. Apply for a quota increase for the number of prediction requests.

Answer: B

Explanation:

Option A is incorrect because using batch prediction mode instead of online mode does not solve the "Out of Memory" error, but rather changes the latency and throughput of the prediction service. Batch prediction mode is suitable for large-scale, asynchronous, and non-urgent predictions, while online prediction mode is suitable for low-latency, synchronous, and real-time predictions1. Option B is correct because sending the request again with a smaller batch of instances can reduce the memory consumption of the prediction service and avoid the "Out of Memory" error. The batch size is the number of instances that are processed together in one request. A smaller batch size means less data to load into memory at once2.

Option C is incorrect because using base64 to encode your data before using it for prediction does not reduce the memory consumption of the prediction service, but rather increases it. Base64 encoding is a way of representing binary data as ASCII characters, which increases the size of the data by about 33%3. Base64 encoding is only required for certain data types, such as images and audio, that cannot be represented as JSON or CSV4.

Option D is incorrect because applying for a quota increase for the number of prediction requests does not solve the "Out of Memory" error, but rather increases the number of requests that can be sent to the prediction service per day. Quotas are limits on the usage of Google Cloud resources, such as CPU, memory, disk, and network5. Quotas do not affect the performance of the prediction service, but rather the availability and cost of the service.

Reference:

Choosing between online and batch prediction

Online prediction input data

Base64 encoding

Preparing data for prediction Quotas and limits

Question: 81

You work at a subscription-based company. You have trained an ensemble of trees and neural networks to predict customer churn, which is the likelihood that customers will not renew their yearly subscription. The average prediction is a 15% churn rate, but for a particular customer the model predicts that they are 70% likely to churn. The customer has a product usage history of 30%, is located in New York City, and became a customer in 1997. You need to explain the difference between the actual prediction, a 70% churn rate, and the average prediction. You want to use Vertex Explainable AI. What should you do?

A. Train local surrogate models to explain individual predictions.
B. Configure sampled Shapley explanations on Vertex Explainable AI.

C. Configure integrated gradients explanations on Vertex Explainable AI.

D. Measure the effect of each feature as the weight of the feature multiplied by the feature value.

Answer: B

Explanation:

Option A is incorrect because training local surrogate models to explain individual predictions is not a feature of Vertex Explainable AI, but rather a general technique for interpreting black-box models. Local surrogate models are simpler models that approximate the behavior of the original model around a specific input1.

Option B is correct because configuring sampled Shapley explanations on Vertex Explainable AI is a way to explain the difference between the actual prediction and the average prediction for a given input. Sampled Shapley explanations are based on the Shapley value, which is a game-theoretic concept that measures how much each feature contributes to the prediction2. Vertex Explainable AI supports sampled Shapley explanations for tabular data, such as customer churn3.

Option C is incorrect because configuring integrated gradients explanations on Vertex Explainable AI is not suitable for explaining the difference between the actual prediction and the average prediction for a given input. Integrated gradients explanations are based on the idea of computing the gradients of the prediction with respect to the input features along a path from a baseline input to the actual input4. Vertex Explainable AI supports integrated gradients explanations for image and text data, but not for tabular data3.

Option D is incorrect because measuring the effect of each feature as the weight of the feature multiplied by the feature value is not a valid way to explain the difference between the actual prediction and the average prediction for a given input. This method assumes that the model is linear and additive, which is not the case for an ensemble of trees and neural networks. Moreover, this method does not account for the interactions between features or the non-linearity of the model5. Reference:

Local surrogate models

Shapley value

Vertex Explainable AI overview

Integrated gradients

Feature importance

Question: 82

You need to execute a batch prediction on 100 million records in a BigQuery table with a custom TensorFlow DNN regressor model, and then store the predicted results in a BigQuery table. You want to minimize the effort required to build this inference pipeline. What should you do?

A. Import the TensorFlow model with BigQuery ML, and run the ml.predict function.

B. Use the TensorFlow BigQuery reader to load the data, and use the BigQuery API to write the results to BigQuery.

C. Create a Dataflow pipeline to convert the data in BigQuery to TFRecords. Run a batch inference on Vertex AI Prediction, and write the results to BigQuery.

D. Load the TensorFlow SavedModel in a Dataflow pipeline. Use the BigQuery I/O connector with a custom function to perform the inference within the pipeline, and write the results to BigQuery.

Answer: A

Explanation:

Option A is correct because importing the TensorFlow model with BigQuery ML, and running the ml.predict function is the easiest way to execute a batch prediction on a large BigQuery table with a custom TensorFlow model, and store the predicted results in another BigQuery table. BigQuery ML allows you to import TensorFlow models that are stored in Cloud Storage, and use them for prediction with SQL queries1. The ml.predict function returns a table with the

predicted values, which can be saved to another BigQuery table2.

Option B is incorrect because using the TensorFlow BigQuery reader to load the data, and using the BigQuery API to write the results to BigQuery requires more effort to build the inference pipeline than option A. The TensorFlow BigQuery reader is a way to read data from BigQuery into TensorFlow datasets, which can be used for training or prediction3. However, this option also requires writing code to load the TensorFlow model, run the prediction, and use the BigQuery API to write the results back to BigQuery4.

Option C is incorrect because creating a Dataflow pipeline to convert the data in BigQuery to TFRecords, running a batch inference on Vertex AI Prediction, and writing the results to BigQuery requires more effort to build the inference pipeline than option A. Dataflow is a service for creating and running data processing pipelines, such as ETL (extract, transform, load) or batch processing5. Vertex AI Prediction is a service for deploying and serving ML models for online or batch prediction. However, this option also requires writing code to create the Dataflow pipeline, convert the data to TFRecords, run the batch inference, and write the results to BigQuery.

Option D is incorrect because loading the TensorFlow SavedModel in a Dataflow pipeline, using the BigQuery I/O connector with a custom function to perform the inference within the pipeline, and writing the results to BigQuery requires more effort to build the inference pipeline than option A. The BigQuery I/O connector is a way to read and write data from BigQuery within a Dataflow pipeline. However, this option also requires writing code to load the TensorFlow SavedModel, create the custom function for inference, and write the results to BigQuery.

Reference:

Importing models into BigQuery ML

Using imported models for prediction

TensorFlow BigQuery reader

BigQuery API

Dataflow overview

[Vertex AI Prediction overview]

[Batch prediction with Dataflow]

[BigQuery I/O connector]

[Using TensorFlow models in Dataflow]

Question: 83

You are creating a deep neural network classification model using a dataset with categorical input values. Certain columns have a cardinality greater than 10,000 unique values. How should you encode these categorical values as input into the model?

A. Convert each categorical value into an integer value.

B. Convert the categorical string data to one-hot hash buckets.

C. Map the categorical variables into a vector of boolean values.

D. Convert each categorical value into a run-length encoded string.

Answer: B

Explanation:

Option A is incorrect because converting each categorical value into an integer value is not a good way to encode categorical values with high cardinality. This method implies an ordinal relationship between the categories, which may not be true. For example, assigning the values 1, 2, and 3 to the categories “red”, “green”, and “blue” does not make sense, as there is no inherent order among these colors1.

Option B is correct because converting the categorical string data to one-hot hash buckets is a suitable way to encode categorical values with high cardinality. This method uses a hash function to map each category to a fixed-length vector of binary values, where only one element is 1 and the rest are 0. This method preserves the sparsity and independence

of the categories, and reduces the dimensionality of the input space2.

Option C is incorrect because mapping the categorical variables into a vector of boolean values is not a valid way to encode categorical values with high cardinality. This method implies that each category can be represented by a combination of true/false values, which may not be possible for a large number of categories. For example, if there are 10,000 categories, then there are 2^10,000 possible combinations of boolean values, which is impractical to store and process3.

Option D is incorrect because converting each categorical value into a run-length encoded string is not a useful way to encode categorical values with high cardinality. This method compresses a string by replacing consecutive repeated characters with the character and the number of repetitions. For example, "AAAABBBCC" becomes "A4B3C2". This method does not reduce the dimensionality of the input space, and does not preserve the semantic meaning of the categories4.

Reference:

Encoding categorical features

One-hot hash buckets

Boolean vector

Run-length encoding

Question: 84

You need to train a natural language model to perform text classification on product descriptions that contain millions of examples and 100,000 unique words. You want to preprocess the words individually so that they can be fed into a recurrent neural network. What should you do?

A. Create a hot-encoding of words, and feed the encodings into your model.
B. Identify word embeddings from a pre-trained model, and use the embeddings in your model.
C. Sort the words by frequency of occurrence, and use the frequencies as the encodings in your model.
D. Assign a numerical value to each word from 1 to 100,000 and feed the values as inputs in your model.

Answer: B

Explanation:

Option A is incorrect because creating a one-hot encoding of words, and feeding the encodings into your model is not an efficient way to preprocess the words individually for a natural language model. One-hot encoding is a method of representing categorical variables as binary vectors, where each element corresponds to a category and only one element is 1 and the rest are 01. However, this method is not suitable for high-dimensional and sparse data, such as words in a large vocabulary, because it requires a lot of memory and computation, and does not capture the semantic similarity or relationship between words2.

Option B is correct because identifying word embeddings from a pre-trained model, and using the embeddings in your model is a good way to preprocess the words individually for a natural language model. Word embeddings are low-dimensional and dense vectors that represent the meaning and usage of words in a continuous space3. Word embeddings can be learned from a large corpus of text using neural networks, such as word2vec, GloVe, or BERT4. Using pre-trained word embeddings can save time and resources, and improve the performance of the natural language model, especially when the training data is limited or noisy5.

Option C is incorrect because sorting the words by frequency of occurrence, and using the frequencies as the encodings in your model is not a meaningful way to preprocess the words individually for a natural language model. This method implies that the frequency of a word is a good indicator of its importance or relevance, which may not be true. For example, the word "the" is very frequent but not very informative, while the word "unicorn" is rare but more

distinctive. Moreover, this method does not capture the semantic similarity or relationship between words, and may introduce noise or bias into the model.

Option D is incorrect because assigning a numerical value to each word from 1 to 100,000 and feeding the values as inputs in your model is not a valid way to preprocess the words individually for a natural language model. This method implies an ordinal relationship between the words, which may not be true. For example, assigning the values 1, 2, and 3 to the words "apple", "banana", and "orange" does not make sense, as there is no inherent order among these fruits. Moreover, this method does not capture the semantic similarity or relationship between words, and may confuse the model with irrelevant or misleading information.

Reference:

One-hot encoding

Word embeddings

Word embedding

Pre-trained word embeddings

Using pre-trained word embeddings in a Keras model [Term frequency]

[Term frequency-inverse document frequency]

[Ordinal variable]

[Encoding categorical features]

Question: 85

Your data science team has requested a system that supports scheduled model retraining, Docker containers, and a service that supports autoscaling and monitoring for online prediction requests. Which platform components should you choose for this system?

A. Vertex AI Pipelines and App Engine
B. Vertex AI Pipelines, Vertex AI Prediction, and Vertex AI Model Monitoring
C. Cloud Composer, BigQuery ML, and Vertex AI Prediction
D. Cloud Composer, Vertex AI Training with custom containers, and App Engine

Answer: B

Explanation:

Option A is incorrect because Vertex AI Pipelines and App Engine do not meet all the requirements of the system. Vertex AI Pipelines is a service that allows you to create, run, and manage ML workflows using TensorFlow Extended (TFX) components or custom components1. App Engine is a service that allows you to build and deploy scalable web applications using standard or flexible environments2. However, App Engine does not support Docker containers in the standard environment, and does not provide a dedicated service for online prediction and monitoring of ML models3.

Option B is correct because Vertex AI Pipelines, Vertex AI Prediction, and Vertex AI Model Monitoring meet all the requirements of the system. Vertex AI Prediction is a service that allows you to deploy and serve ML models for online or batch prediction, with support for autoscaling and custom containers4. Vertex AI Model Monitoring is a service that allows you to monitor the performance and fairness of your deployed models, and get alerts for any issues or anomalies5.

Option C is incorrect because Cloud Composer, BigQuery ML, and Vertex AI Prediction do not meet all the requirements of the system. Cloud Composer is a service that allows you to create, schedule, and manage workflows using Apache Airflow. BigQuery ML is a service that allows you to create and use ML models within BigQuery using SQL queries. However, BigQuery ML does not support custom containers, and Vertex AI Prediction does not support scheduled model retraining or model monitoring.

Option D is incorrect because Cloud Composer, Vertex AI Training with custom containers, and App Engine do not meet

all the requirements of the system. Vertex AI Training is a service that allows you to train ML models using built-in algorithms or custom containers. However, Vertex AI Training does not support online prediction or model monitoring, and App Engine does not support Docker containers in the standard environment or online prediction and monitoring of ML models3. Reference:

Vertex AI Pipelines overview
App Engine overview
Choosing an App Engine environment
Vertex AI Prediction overview
Vertex AI Model Monitoring overview
[Cloud Composer overview]
[BigQuery ML overview]
[BigQuery ML limitations] [Vertex AI Training overview]

Question: 86

You are profiling the performance of your TensorFlow model training time and notice a performance issue caused by inefficiencies in the input data pipeline for a single 5 terabyte CSV file dataset on Cloud Storage. You need to optimize the input pipeline performance. Which action should you try first to increase the efficiency of your pipeline?

A. Preprocess the input CSV file into a TFRecord file.
B. Randomly select a 10 gigabyte subset of the data to train your model.
C. Split into multiple CSV files and use a parallel interleave transformation.
D. Set the reshuffle_each_iteration parameter to true in the tf.data.Dataset.shuffle method.

Answer: A

Explanation:
According to the web search results, the TFRecord format is a recommended way to store large amounts of data efficiently and improve the performance of the data input pipeline123. The TFRecord format is a binary format that can be compressed and serialized, which reduces the I/O overhead and the memory footprint of the data1. The tf.data API provides tools to create and read TFRecord files easily1.
The other options are not as effective as option A. Option B would reduce the amount of data available for training and might affect the model accuracy. Option C would still require reading from a single CSV file at a time, which might not utilize the full bandwidth of the remote storage. Option D would only affect the order of the data elements, not the speed of reading them.

Question: 87

You need to design an architecture that serves asynchronous predictions to determine whether a particular mission-critical machine part will fail. Your system collects data from multiple sensors from the machine. You want to build a model that will predict a failure in the next N minutes, given the average of each sensor's data from the past 12 hours. How should you design the architecture?

A. 1. HTTP requests are sent by the sensors to your ML model, which is deployed as a microservice and exposes a REST API for prediction
2. Your application queries a Vertex AI endpoint where you deployed your model.
3. Responses are received by the caller application as soon as the model produces the prediction.

B. 1. Events are sent by the sensors to Pub/Sub, consumed in real time, and processed by a Dataflow stream processing pipeline.

2. The pipeline invokes the model for prediction and sends the predictions to another Pub/Sub topic.

3. Pub/Sub messages containing predictions are then consumed by a downstream system for monitoring.

C. 1. Export your data to Cloud Storage using Dataflow.

2. Submit a Vertex AI batch prediction job that uses your trained model in Cloud Storage to perform scoring on the preprocessed data.

3. Export the batch prediction job outputs from Cloud Storage and import them into Cloud SQL.

D. 1. Export the data to Cloud Storage using the BigQuery command-line tool

2. Submit a Vertex AI batch prediction job that uses your trained model in Cloud Storage to perform scoring on the preprocessed data.

3. Export the batch prediction job outputs from Cloud Storage and import them into BigQuery.

Answer: B

Explanation:

Reasoning: The question asks for a design that serves asynchronous predictions to determine whether a machine part will fail. This means that the predictions do not need to be returned immediately to the sensors, but can be processed in batches and sent to a downstream system for monitoring. Option B is the only one that uses a streaming data pipeline with Pub/Sub and Dataflow, which can handle real-time data ingestion, processing, and prediction. Option B also invokes the model for prediction, which is required by the question. The other options either use synchronous predictions (option A), batch predictions (options C and D), or do not invoke the model for prediction (option D).

Reference: You can learn more about the differences between synchronous, asynchronous, and batch predictions in Vertex AI from this document. You can also find examples of how to use Pub/Sub and Dataflow for streaming data pipelines from this tutorial and this codelab.

Question: 88

Your company manages an application that aggregates news articles from many different online sources and sends them to users. You need to build a recommendation model that will suggest articles to readers that are similar to the articles they are currently reading. Which approach should you use?

A. Create a collaborative filtering system that recommends articles to a user based on the user's past behavior.

B. Encode all articles into vectors using word2vec, and build a model that returns articles based on vector similarity.

C. Build a logistic regression model for each user that predicts whether an article should be recommended to a user.

D. Manually label a few hundred articles, and then train an SVM classifier based on the manually classified articles that categorizes additional articles into their respective categories.

Answer: B

Explanation:

Option A is incorrect because creating a collaborative filtering system that recommends articles to a user based on the user's past behavior is not the best approach to suggest articles that are similar to the articles they are currently reading. Collaborative filtering is a method of recommendation that uses the ratings or preferences of other users to predict the preferences of a target user1. However, this method does not consider the content or features of the articles, and may not be able to find articles that are similar in terms of topic, style, or sentiment.

Option B is correct because encoding all articles into vectors using word2vec, and building a model that returns articles

based on vector similarity is a suitable approach to suggest articles that are similar to the articles they are currently reading. Word2vec is a technique that learns lowdimensional and dense representations of words from a large corpus of text, such that words that are semantically similar have similar vectors2. By applying word2vec to the articles, we can obtain vector representations of the articles that capture their meaning and usage. Then, we can use a similarity measure, such as cosine similarity, to find articles that have similar vectors to the current article3.

Option C is incorrect because building a logistic regression model for each user that predicts whether an article should be recommended to a user is not a feasible approach to suggest articles that are similar to the articles they are currently reading. Logistic regression is a supervised learning method that models the probability of a binary outcome (such as recommend or not) based on some input features (such as user profile or article content)4. However, this method requires a large amount of labeled data for each user, which may not be available or scalable. Moreover, this method does not directly measure the similarity between articles, but rather the likelihood of a user's preference.

Option D is incorrect because manually labeling a few hundred articles, and then training an SVM classifier based on the manually classified articles that categorizes additional articles into their respective categories is not an effective approach to suggest articles that are similar to the articles they are currently reading. SVM (support vector machine) is a supervised learning method that finds a hyperplane that separates the data into different classes (such as news categories) with the maximum margin5. However, this method also requires a large amount of labeled data, which may be costly and time-consuming to obtain. Moreover, this method does not account for the finegrained similarity between articles within the same category, or the cross-category similarity between articles from different categories.

Reference:

Collaborative filtering

Word2vec

Cosine similarity

Logistic regression SVM

Question: 89

You work for a large social network service provider whose users post articles and discuss news. Millions of comments are posted online each day, and more than 200 human moderators constantly review comments and flag those that are inappropriate. Your team is building an ML model to help human moderators check content on the platform. The model scores each comment and flags suspicious comments to be reviewed by a human. Which metric(s) should you use to monitor the model's performance?

A. Number of messages flagged by the model per minute

B. Number of messages flagged by the model per minute confirmed as being inappropriate by humans.

C. Precision and recall estimates based on a random sample of 0.1% of raw messages each minute sent to a human for review

D. Precision and recall estimates based on a sample of messages flagged by the model as potentially inappropriate each minute

Answer: D

Explanation:

Precision measures the fraction of messages flagged by the model that are actually inappropriate, while recall measures the fraction of inappropriate messages that are flagged by the model. These metrics are useful for evaluating how well the model can identify and filter out inappropriate comments.

Option A is not a good metric because it does not account for the accuracy of the model. The model

might flag many messages that are not inappropriate, or miss many messages that are inappropriate. Option B is better than option A, but it still does not account for the recall of the model. The model might flag only a few messages that are highly likely to be inappropriate, but miss many other messages that are less obvious but still inappropriate.

Option C is not a good metric because it does not focus on the messages that are flagged by the model. The random sample of 0.1% of raw messages might contain very few inappropriate messages, making the precision and recall estimates unreliable.

Question: 90

You are a lead ML engineer at a retail company. You want to track and manage ML metadata in a centralized way so that your team can have reproducible experiments by generating artifacts. Which management solution should you recommend to your team?

A. Store your tf.logging data in BigQuery.
B. Manage all relational entities in the Hive Metastore.
C. Store all ML metadata in Google Cloud's operations suite.
D. Manage your ML workflows with Vertex ML Metadata.

Answer: D

Explanation:

Vertex ML Metadata is a service that lets you track and manage the metadata produced by your ML workflows in a centralized way. It helps you have reproducible experiments by generating artifacts that represent the data, parameters, and metrics used or produced by your ML system. You can also analyze the lineage and performance of your ML artifacts using Vertex ML Metadata.

Some of the benefits of using Vertex ML Metadata are:

It captures your ML system's metadata as a graph, where artifacts and executions are nodes, and events are edges that link them as inputs or outputs.

It allows you to create contexts to group sets of artifacts and executions together, such as experiments, runs, or projects.

It supports querying and filtering the metadata using the Vertex AI SDK for Python or REST commands.

It integrates with other Vertex AI services, such as Vertex AI Pipelines and Vertex AI Experiments, to automatically log metadata and artifacts.

The other options are not suitable for tracking and managing ML metadata in a centralized way. Option A: Storing your tf.logging data in BigQuery is not enough to capture the full metadata of your ML system, such as the artifacts and their lineage. BigQuery is a data warehouse service that is mainly used for analytics and reporting, not for metadata management.

Option B: Managing all relational entities in the Hive Metastore is not a good solution for ML metadata, as it is designed for storing metadata of Hive tables and partitions, not for ML artifacts and executions. Hive Metastore is a component of the Apache Hive project, which is a data warehouse system for querying and analyzing large datasets stored in Hadoop.

Option C: Storing all ML metadata in Google Cloud's operations suite is not a feasible option, as it is a set of tools for monitoring, logging, tracing, and debugging your applications and infrastructure, not for ML metadata. Google Cloud's operations suite does not provide the features and integrations that Vertex ML Metadata offers for ML workflows.

www.atmicnetworks.com

Question: 91

You have been given a dataset with sales predictions based on your company's marketing activities. The data is structured and stored in BigQuery, and has been carefully managed by a team of data analysts. You need to prepare a report providing insights into the predictive capabilities of the dat a. You were asked to run several ML models with different levels of sophistication, including simple models and multilayered neural networks. You only have a few hours to gather the results of your experiments. Which Google Cloud tools should you use to complete this task in the most efficient and self-serviced way?

A. Use BigQuery ML to run several regression models, and analyze their performance.

B. Read the data from BigQuery using Dataproc, and run several models using SparkML.

C. Use Vertex AI Workbench user-managed notebooks with scikit-learn code for a variety of ML algorithms and performance metrics.

D. Train a custom TensorFlow model with Vertex AI, reading the data from BigQuery featuring a variety of ML algorithms.

Answer: A

Explanation:

Option A is correct because using BigQuery ML to run several regression models, and analyze their performance is the most efficient and self-serviced way to complete the task. BigQuery ML is a service that allows you to create and use ML models within BigQuery using SQL queries1. You can use BigQuery ML to run different types of regression models, such as linear regression, logistic regression, or DNN regression2. You can also use BigQuery ML to analyze the performance of your models, such as the mean squared error, the accuracy, or the ROC curve3. BigQuery ML is fast, scalable, and easy to use, as it does not require any data movement, coding, or additional tools4. Option B is incorrect because reading the data from BigQuery using Dataproc, and running several models using SparkML is not the most efficient and self-serviced way to complete the task. Dataproc is a service that allows you to create and manage clusters of virtual machines that run Apache Spark and other open-source tools5. SparkML is a library that provides ML algorithms and utilities for Spark. However, this option requires more effort and resources than option A, as it involves moving the data from BigQuery to Dataproc, creating and configuring the clusters, writing and running the SparkML code, and analyzing the results.

Option C is incorrect because using Vertex AI Workbench user-managed notebooks with scikit-learn code for a variety of ML algorithms and performance metrics is not the most efficient and selfserviced way to complete the task. Vertex AI Workbench is a service that allows you to create and use notebooks for ML development and experimentation. Scikit-learn is a library that provides ML algorithms and utilities for Python. However, this option also requires more effort and resources than option A, as it involves creating and managing the notebooks, writing and running the scikit-learn code, and analyzing the results.

Option D is incorrect because training a custom TensorFlow model with Vertex AI, reading the data from BigQuery featuring a variety of ML algorithms is not the most efficient and self-serviced way to complete the task. TensorFlow is a framework that allows you to create and train ML models using Python or other languages. Vertex AI is a service that allows you to train and deploy ML models using built-in algorithms or custom containers. However, this option also requires more effort and resources than option A, as it involves writing and running the TensorFlow code, creating and managing the training jobs, and analyzing the results. Reference:

BigQuery ML overview

Creating a model in BigQuery ML

Evaluating a model in BigQuery ML

BigQuery ML benefits

Dataproc overview

[SparkML overview]
[Vertex AI Workbench overview]
[Scikit-learn overview]
[TensorFlow overview]
[Vertex AI overview]

Question: 92

You are an ML engineer at a bank. You have developed a binary classification model using AutoML Tables to predict whether a customer will make loan payments on time. The output is used to approve or reject loan requests. One customer's loan request has been rejected by your model, and the bank's risks department is asking you to provide the reasons that contributed to the model's decision. What should you do?

A. Use local feature importance from the predictions.
B. Use the correlation with target values in the data summary page.
C. Use the feature importance percentages in the model evaluation page.
D. Vary features independently to identify the threshold per feature that changes the classification.

Answer: A

Explanation:
Option A is correct because using local feature importance from the predictions is the best way to provide the reasons that contributed to the model's decision for a specific customer's loan request. Local feature importance is a measure of how much each feature affects the prediction for a given instance, relative to the average prediction for the dataset1. AutoML Tables provides local feature importance values for each prediction, which can be accessed using the Vertex AI SDK for Python or the Cloud Console2. By using local feature importance, you can explain why the model rejected the loan request based on the customer's data.
Option B is incorrect because using the correlation with target values in the data summary page is not a good way to provide the reasons that contributed to the model's decision for a specific customer's loan request. The correlation with target values is a measure of how much each feature is linearly related to the target variable for the entire dataset, not for a single instance3. The data summary page in AutoML Tables shows the correlation with target values for each feature, as well as other statistics such as mean, standard deviation, and histogram4. However, these statistics are not useful for explaining the model's decision for a specific customer, as they do not account for the interactions between features or the non-linearity of the model.
Option C is incorrect because using the feature importance percentages in the model evaluation page is not a good way to provide the reasons that contributed to the model's decision for a specific customer's loan request. The feature importance percentages are a measure of how much each feature affects the overall accuracy of the model for the entire dataset, not for a single instance5. The model evaluation page in AutoML Tables shows the feature importance percentages for each feature, as well as other metrics such as precision, recall, and confusion matrix. However, these metrics are not useful for explaining the model's decision for a specific customer, as they do not reflect the individual contribution of each feature for a given prediction.
Option D is incorrect because varying features independently to identify the threshold per feature that changes the classification is not a feasible way to provide the reasons that contributed to the model's decision for a specific customer's loan request. This method involves changing the value of one feature at a time, while keeping the other features constant, and observing how the prediction changes. However, this method is not practical, as it requires making multiple prediction requests, and may not capture the interactions between features or the non-linearity of the model. Reference:
Local feature importance

Getting local feature importance values
Correlation with target values
Data summary page
Feature importance percentages
[Model evaluation page]
[Varying features independently]

Question: 93

You work for a magazine distributor and need to build a model that predicts which customers will renew their subscriptions for the upcoming year. Using your company's historical data as your training set, you created a TensorFlow model and deployed it to AI Platform. You need to determine which customer attribute has the most predictive power for each prediction served by the model. What should you do?

A. Use AI Platform notebooks to perform a Lasso regression analysis on your model, which will eliminate features that do not provide a strong signal.
B. Stream prediction results to BigQuery. Use BigQuery's CORR(X1, X2) function to calculate the Pearson correlation coefficient between each feature and the target variable.
C. Use the AI Explanations feature on AI Platform. Submit each prediction request with the 'explain' keyword to retrieve feature attributions using the sampled Shapley method.
D. Use the What-If tool in Google Cloud to determine how your model will perform when individual features are excluded. Rank the feature importance in order of those that caused the most significant performance drop when removed from the model.

Answer: C

Explanation:

Option A is incorrect because using AI Platform notebooks to perform a Lasso regression analysis on your model, which will eliminate features that do not provide a strong signal, is not a suitable way to determine which customer attribute has the most predictive power for each prediction served by the model. Lasso regression is a method of feature selection that applies a penalty to the coefficients of the linear model, and shrinks them to zero for irrelevant features1. However, this method assumes that the model is linear and additive, which may not be the case for a TensorFlow model. Moreover,
this method does not provide feature attributions for each prediction, but rather for the entire dataset.
Option B is incorrect because streaming prediction results to BigQuery, and using BigQuery's CORR(X1, X2) function to calculate the Pearson correlation coefficient between each feature and the target variable, is not a valid way to determine which customer attribute has the most predictive power for each prediction served by the model. The Pearson correlation coefficient is a measure of the linear relationship between two variables, ranging from -1 to 12. However, this method does not account for the interactions between features or the non-linearity of the model. Moreover, this method does not provide feature attributions for each prediction, but rather for the entire dataset.
Option C is correct because using the AI Explanations feature on AI Platform, and submitting each prediction request with the 'explain' keyword to retrieve feature attributions using the sampled Shapley method, is the best way to determine which customer attribute has the most predictive power for each prediction served by the model. AI Explanations is a service that allows you to get feature attributions for your deployed models on AI Platform3. Feature attributions are values that indicate how much each feature contributed to the prediction for a given instance4. The sampled Shapley method is a technique that uses the Shapley value, a game-theoretic concept, to measure the contribution of each feature to the prediction5. By using AI Explanations, you can get feature attributions for each prediction request, and identify the most important features for each customer. Option D is incorrect because using the

What-If tool in Google Cloud to determine how your model will perform when individual features are excluded, and ranking the feature importance in order of those that caused the most significant performance drop when removed from the model, is not a practical way to determine which customer attribute has the most predictive power for each prediction served by the model. The What-If tool is a tool that allows you to visualize and analyze your ML models and datasets. However, this method requires manually editing or removing features for each instance, and observing the change in the prediction. This method is not scalable or efficient, and may not capture the interactions between features or the non-linearity of the model. Reference:

Lasso regression

Pearson correlation coefficient

AI Explanations overview Feature attributions Sampled Shapley method [What-If tool overview]

Question: 94

You are working on a binary classification ML algorithm that detects whether an image of a classified scanned document contains a company's logo. In the dataset, 96% of examples don't have the logo, so the dataset is very skewed. Which metrics would give you the most confidence in your model?

A. F-score where recall is weighed more than precision

B. RMSE

C. F1 score

D. F-score where precision is weighed more than recall

Answer: A

Explanation:

Option A is correct because using F-score where recall is weighed more than precision is a suitable metric for binary classification with imbalanced data. F-score is a harmonic mean of precision and recall, which are two metrics that measure the accuracy and completeness of the positive class1. Precision is the fraction of true positives among all predicted positives, while recall is the fraction of true positives among all actual positives1. When the data is imbalanced, the positive class is the minority class, which is usually the class of interest. For example, in this case, the positive class is the images that contain the company's logo, which are rare but important to detect. By weighing recall more than precision, we can emphasize the importance of finding all the positive examples, even if some false positives are included2.

Option B is incorrect because using RMSE (root mean squared error) is not a valid metric for binary classification with imbalanced data. RMSE is a metric that measures the average magnitude of the errors between the predicted and actual values3. RMSE is suitable for regression problems, where the target variable is continuous, not for classification problems, where the target variable is discrete4.

Option C is incorrect because using F1 score is not the best metric for binary classification with imbalanced data. F1 score is a special case of F-score where precision and recall are equally weighted1. F1 score is suitable for balanced data, where the positive and negative classes are equally important and frequent5. However, for imbalanced data, the positive class is more important and less frequent than the negative class, so F1 score may not reflect the performance of the model well2.

Option D is incorrect because using F-score where precision is weighed more than recall is not a good metric for binary classification with imbalanced data. By weighing precision more than recall, we can emphasize the importance of minimizing the false positives, even if some true positives are missed2. However, for imbalanced data, the true positives are more important and less frequent than the false positives, so this metric may not reflect the performance of the model well2. Reference:

Precision, recall, and F-measure

F-score for imbalanced data

RMSE

Regression vs classification F1 score

[Imbalanced classification]

[Binary classification]

Question: 95

You work on the data science team for a multinational beverage company. You need to develop an ML model to predict the company's profitability for a new line of naturally flavored bottled waters in different locations. You are provided with historical data that includes product types, product sales volumes, expenses, and profits for all regions. What should you use as the input and output for your model?

A. Use latitude, longitude, and product type as features. Use profit as model output.

B. Use latitude, longitude, and product type as features. Use revenue and expenses as model outputs.

C. Use product type and the feature cross of latitude with longitude, followed by binning, as features. Use profit as model output.

D. Use product type and the feature cross of latitude with longitude, followed by binning, as features. Use revenue and expenses as model outputs.

Answer: C

Explanation:

Option A is incorrect because using latitude, longitude, and product type as features, and using profit as model output is not the best way to develop an ML model to predict the company's profitability for a new line of naturally flavored bottled waters in different locations. This option does not capture the interaction between latitude and longitude, which may affect the profitability of the product. For example, the same product may have different profitability in different regions, depending on the climate, culture, or preferences of the customers. Moreover, this option does not account for the granularity of the location data, which may be too fine or too coarse for the model. For example, using the exact coordinates of a city may not be meaningful, as the profitability may vary within the city, or using the country name may not be informative, as the profitability may vary across the country.

Option B is incorrect because using latitude, longitude, and product type as features, and using revenue and expenses as model outputs is not a suitable way to develop an ML model to predict the company's profitability for a new line of naturally flavored bottled waters in different locations. This option has the same drawbacks as option A, as it does not capture the interaction between latitude and longitude, or account for the granularity of the location data. Moreover, this option does not directly predict the profitability of the product, which is the target variable of interest. Instead, it predicts the revenue and expenses of the product, which are intermediate variables that depend on other factors, such as the price, the cost, or the demand of the product. To obtain the profitability, we would need to subtract the expenses from the revenue, which may introduce errors or uncertainties in the prediction.

Option C is correct because using product type and the feature cross of latitude with longitude, followed by binning, as features, and using profit as model output is a good way to develop an ML model to predict the company's profitability for a new line of naturally flavored bottled waters in different locations. This option captures the interaction between latitude and longitude, which may affect the profitability of the product, by creating a feature cross of these two features. A feature cross is a synthetic feature that combines the values of two or more features into a single feature1. This option also accounts for the granularity of the location data, by binning the feature cross into discrete buckets. Binning is a technique that groups continuous values into intervals, which can reduce the noise and complexity of the data2. Moreover, this option directly predicts the profitability of the product, which is the target variable of interest, by using it as the model output.

Option D is incorrect because using product type and the feature cross of latitude with longitude, followed by binning, as features, and using revenue and expenses as model outputs is not a valid way to develop an ML model to predict the company's profitability for a new line of naturally flavored bottled waters in different locations. This option has the same advantages as option C, as it captures the interaction between latitude and longitude, and accounts for the granularity of the location data, by creating a feature cross and binning it. However, this option does not directly predict the profitability of the product, which is the target variable of interest, but rather predicts the revenue and expenses of the product, which are intermediate variables that depend on other factors, as explained in option B.

Reference:

Feature cross

Binning

[Profitability]
[Revenue and expenses]
[Latitude and longitude]
[Product type]

Question: 96

You work as an ML engineer at a social media company, and you are developing a visual filter for users' profile photos. This requires you to train an ML model to detect bounding boxes around human faces. You want to use this filter in your company's iOS-based mobile phone application. You want to minimize code development and want the model to be optimized for inference on mobile phones. What should you do?

A. Train a model using AutoML Vision and use the "export for Core ML" option.
B. Train a model using AutoML Vision and use the "export for Coral" option.
C. Train a model using AutoML Vision and use the "export for TensorFlow.js" option.
D. Train a custom TensorFlow model and convert it to TensorFlow Lite (TFLite).

Answer: A

Explanation:

AutoML Vision is a Google Cloud service that allows you to train custom ML models for image classification, object detection, and segmentation without writing any code. You can use AutoML Vision to upload your training data, label it, and train a model using a graphical user interface. You can also evaluate the model's performance and export it for deployment. One of the export options is Core ML, which is a framework that lets you integrate ML models into iOS applications. Core ML optimizes the model for on-device performance, power efficiency, and minimal memory footprint. By using AutoML Vision and Core ML, you can minimize code development and have a model that is optimized for inference on mobile phones. Reference:

AutoML Vision documentation Core ML documentation

Question: 97

You have been asked to build a model using a dataset that is stored in a medium-sized (~10 GB) BigQuery table. You need to quickly determine whether this data is suitable for model development. You want to create a one-time report that includes both informative visualizations of data distributions and more sophisticated statistical analyses to share with other ML engineers on your team. You require maximum flexibility to create your report. What should you do?

A. Use Vertex AI Workbench user-managed notebooks to generate the report.

B. Use the Google Data Studio to create the report.

C. Use the output from TensorFlow Data Validation on Dataflow to generate the report.

D. Use Dataprep to create the report.

Answer: A

Explanation:

Option A is correct because using Vertex AI Workbench user-managed notebooks to generate the report is the best way to quickly determine whether the data is suitable for model development, and to create a one-time report that includes both informative visualizations of data distributions and more sophisticated statistical analyses to share with other ML engineers on your team. Vertex AI Workbench is a service that allows you to create and use notebooks for ML development and experimentation. You can use Vertex AI Workbench to connect to your BigQuery table, query and analyze the data using SQL or Python, and create interactive charts and plots using libraries such as pandas, matplotlib, or seaborn. You can also use Vertex AI Workbench to perform more advanced data analysis, such as outlier detection, feature engineering, or hypothesis testing, using libraries such as TensorFlow Data Validation, TensorFlow Transform, or SciPy. You can export your notebook as a PDF or HTML file, and share it with your team. Vertex AI Workbench provides maximum flexibility to create your report, as you can use any code or library that you want, and customize the report as you wish.

Option B is incorrect because using Google Data Studio to create the report is not the most flexible way to quickly determine whether the data is suitable for model development, and to create a onetime report that includes both informative visualizations of data distributions and more sophisticated statistical analyses to share with other ML engineers on your team. Google Data Studio is a service that allows you to create and share interactive dashboards and reports using data from various sources, such as BigQuery, Google Sheets, or Google Analytics. You can use Google Data Studio to connect to your BigQuery table, explore and visualize the data using charts, tables, or maps, and apply filters, calculations, or aggregations to the data. However, Google Data Studio does not support more sophisticated statistical analyses, such as outlier detection, feature engineering, or hypothesis testing, which may be useful for model development. Moreover, Google Data Studio is more suitable for creating recurring reports that need to be updated frequently, rather than one-time reports that are static.

Option C is incorrect because using the output from TensorFlow Data Validation on Dataflow to generate the report is not the most efficient way to quickly determine whether the data is suitable for model development, and to create a one-time report that includes both informative visualizations of data distributions and more sophisticated statistical analyses to share with other ML engineers on your team. TensorFlow Data Validation is a library that allows you to explore, validate, and monitor the quality of your data for ML. You can use TensorFlow Data Validation to compute descriptive statistics, detect anomalies, infer schemas, and generate data visualizations for your data. Dataflow is a service that allows you to create and run scalable data processing pipelines using Apache Beam. You can use Dataflow to run TensorFlow Data Validation on large datasets, such as those stored in BigQuery. However, this option is not very efficient, as it involves moving the data from BigQuery to Dataflow, creating and running the pipeline, and exporting the results. Moreover, this option does not provide maximum flexibility to create your report, as you are limited by the functionalities of TensorFlow Data Validation, and you may not be able to customize the report as you wish.

Option D is incorrect because using Dataprep to create the report is not the most flexible way to quickly determine whether the data is suitable for model development, and to create a one-time report that includes both informative visualizations of data distributions and more sophisticated statistical analyses to share with other ML engineers on your team. Dataprep is a service that allows you to explore, clean, and transform your data for analysis or ML. You can use Dataprep to connect to

your BigQuery table, inspect and profile the data using histograms, charts, or summary statistics, and apply transformations, such as filtering, joining, splitting, or aggregating, to the data. However, Dataprep does not support more sophisticated statistical analyses, such as outlier detection, feature engineering, or hypothesis testing, which may be useful for model development. Moreover, Dataprep is more suitable for creating data preparation workflows that need to be executed repeatedly, rather than one-time reports that are static.

Reference:

Vertex AI Workbench documentation
Google Data Studio documentation
TensorFlow Data Validation documentation
Dataflow documentation
Dataprep documentation
[BigQuery documentation]
[pandas documentation]
[matplotlib documentation]
[seaborn documentation]
[TensorFlow Transform documentation]
[SciPy documentation]
[Apache Beam documentation]

Question: 98

You work on an operations team at an international company that manages a large fleet of onpremises servers located in few data centers around the world. Your team collects monitoring data from the servers, including CPU/memory consumption. When an incident occurs on a server, your team is responsible for fixing it. Incident data has not been properly labeled yet. Your management team wants you to build a predictive maintenance solution that uses monitoring data from the VMs to detect potential failures and then alerts the service desk team. What should you do first?

A. Train a time-series model to predict the machines' performance values. Configure an alert if a machine's actual performance values significantly differ from the predicted performance values. B. Implement a simple heuristic (e.g., based on z-score) to label the machines' historical performance data. Train a model to predict anomalies based on this labeled dataset.
C. Develop a simple heuristic (e.g., based on z-score) to label the machines' historical performance data. Test this heuristic in a production environment.
D. Hire a team of qualified analysts to review and label the machines' historical performance data. Train a model based on this manually labeled dataset.

Answer: B

Explanation:
Option A is incorrect because training a time-series model to predict the machines' performance values, and configuring an alert if a machine's actual performance values significantly differ from the predicted performance values, is not the best way to build a predictive maintenance solution that uses monitoring data from the VMs to detect potential failures and then alerts the service desk team. This option assumes that the performance values follow a predictable pattern, which may not be the case for complex systems. Moreover, this option does not use any historical incident data, which may contain useful information for identifying failures. Furthermore, this option does not involve any model evaluation or validation, which are essential steps for ensuring the quality and reliability of the model.
Option B is correct because implementing a simple heuristic (e.g., based on z-score) to label the machines' historical performance data, and training a model to predict anomalies based on this labeled dataset, is a reasonable way to build a predictive maintenance solution that uses monitoring data from the VMs to detect potential failures and then alerts the service desk team. This option uses a simple and fast method to label the historical performance data, which is necessary for supervised learning. A z-score is a measure of how many standard deviations a value is away from the mean of a distribution1. By using a z-score, we can label the performance values that are unusually high or low as anomalies, which may indicate failures. Then, we can train a model to learn the patterns of normal and anomalous

performance values, and use it to predict anomalies on new data. We can also evaluate and validate the model using metrics such as precision, recall, or F1-score, and compare it with other models or methods.

Option C is incorrect because developing a simple heuristic (e.g., based on z-score) to label the machines' historical performance data, and testing this heuristic in a production environment, is not a safe way to build a predictive maintenance solution that uses monitoring data from the VMs to detect potential failures and then alerts the service desk team. This option does not involve any model training or evaluation, which are essential steps for ensuring the quality and reliability of the solution. Moreover, this option does not test the heuristic on a separate dataset, such as a validation or test set, before deploying it to production, which may lead to errors or failures in the production environment.

Option D is incorrect because hiring a team of qualified analysts to review and label the machines' historical performance data, and training a model based on this manually labeled dataset, is not a feasible way to build a predictive maintenance solution that uses monitoring data from the VMs to detect potential failures and then alerts the service desk team. This option may produce high-quality labels, but it is also costly, time-consuming, and prone to human errors or biases. Moreover, this option may not scale well with large or complex datasets, which may require more analysts or more time to label.

Reference:

Z-score

[Predictive maintenance]

[Anomaly detection] [Time-series analysis] [Model evaluation]

Question: 99

You are developing an ML model that uses sliced frames from video feed and creates bounding boxes around specific objects. You want to automate the following steps in your training pipeline: ingestion and preprocessing of data in Cloud Storage, followed by training and hyperparameter tuning of the object model using Vertex AI jobs, and finally deploying the model to an endpoint. You want to orchestrate the entire pipeline with minimal cluster management. What approach should you use?

A. Use Kubeflow Pipelines on Google Kubernetes Engine.
B. Use Vertex AI Pipelines with TensorFlow Extended (TFX) SDK.
C. Use Vertex AI Pipelines with Kubeflow Pipelines SDK.
D. Use Cloud Composer for the orchestration.

Answer: B

Explanation:

Option A is incorrect because using Kubeflow Pipelines on Google Kubernetes Engine is not the most convenient way to orchestrate the entire pipeline with minimal cluster management. Kubeflow Pipelines is an open-source platform that allows you to build, run, and manage ML pipelines using containers1. Google Kubernetes Engine is a service that allows you to create and manage clusters of virtual machines that run Kubernetes, an open-source system for orchestrating containerized applications2. However, this option requires more effort and resources than option B, as it involves creating and configuring the clusters, installing and maintaining Kubeflow Pipelines, and writing and running the pipeline code.

Option B is correct because using Vertex AI Pipelines with TensorFlow Extended (TFX) SDK is the best way to orchestrate the entire pipeline with minimal cluster management. Vertex AI Pipelines is a service that allows you to create and run scalable and portable ML pipelines on Google Cloud3. TensorFlow Extended (TFX) is a framework that provides a set of components and libraries for building production-ready ML pipelines using TensorFlow4. You can use Vertex AI Pipelines with TFX SDK to ingest and preprocess the data in Cloud Storage, train and tune the object model using Vertex AI jobs,

and deploy the model to an endpoint, using predefined or custom components. Vertex AI Pipelines handles the underlying infrastructure and orchestration for you, so you don't need to worry about cluster management or scalability.

Option C is incorrect because using Vertex AI Pipelines with Kubeflow Pipelines SDK is not the most suitable way to orchestrate the entire pipeline with minimal cluster management. Kubeflow Pipelines SDK is a library that allows you to build and run ML pipelines using Kubeflow Pipelines5. You can use Vertex AI Pipelines with Kubeflow Pipelines SDK to create and run ML pipelines on Google Cloud, using containers. However, this option is less convenient and consistent than option B, as it requires you to use different APIs and tools for different steps of the pipeline, such as Vertex AI SDK for training and deployment, and Kubeflow Pipelines SDK for ingestion and preprocessing. Moreover, this option does not leverage the benefits of TFX, such as the standard components, the metadata store, or the ML Metadata library.

Option D is incorrect because using Cloud Composer for the orchestration is not the most efficient way to orchestrate the entire pipeline with minimal cluster management. Cloud Composer is a service that allows you to create and run workflows using Apache Airflow, an open-source platform for orchestrating complex tasks. You can use Cloud Composer to orchestrate the entire pipeline, by creating and managing DAGs (directed acyclic graphs) that define the dependencies and order of the tasks. However, this option is more complex and costly than option B, as it involves creating and configuring the environments, installing and maintaining Airflow, and writing and running the DAGs.

Reference:

Kubeflow Pipelines documentation

Google Kubernetes Engine documentation

Vertex AI Pipelines documentation

TensorFlow Extended documentation

Kubeflow Pipelines SDK documentation [Cloud Composer documentation] [Vertex AI documentation]

[Cloud Storage documentation]

[TensorFlow documentation]

Question: 100

You are training an object detection machine learning model on a dataset that consists of three million X-ray images, each roughly 2 GB in size. You are using Vertex AI Training to run a custom training application on a Compute Engine instance with 32-cores, 128 GB of RAM, and 1 NVIDIA P100 GPU. You notice that model training is taking a very long time. You want to decrease training time without sacrificing model performance. What should you do?

A. Increase the instance memory to 512 GB and increase the batch size.

B. Replace the NVIDIA P100 GPU with a v3-32 TPU in the training job.

C. Enable early stopping in your Vertex AI Training job.

D. Use the tf.distribute.Strategy API and run a distributed training job.

Answer: D

Explanation:

Question: 101

You are a data scientist at an industrial equipment manufacturing company. You are developing a regression model to estimate the power consumption in the company's manufacturing plants based on sensor data collected from all of the plants. The sensors collect tens of millions of records every day. You need to schedule daily training runs for your model that use all the data collected up to the current date. You want your model to scale smoothly and require minimal

development work. What should you do?

A. Develop a custom TensorFlow regression model, and optimize it using Vertex AI Training.
B. Develop a regression model using BigQuery ML.
C. Develop a custom scikit-learn regression model, and optimize it using Vertex AI Training
D. Develop a custom PyTorch regression model, and optimize it using Vertex AI Training

Answer: B

Explanation:
BigQuery ML is a powerful tool that allows you to build and deploy machine learning models directly within BigQuery, Google's fully-managed, serverless data warehouse. It allows you to create regression models using SQL, which is a familiar and easy-to-use language for many data scientists. It also allows you to scale smoothly and require minimal development work since you don't have to worry about cluster management and it's fully-managed by Google. BigQuery ML also allows you to run your training on the same data where it's stored, this will minimize data movement, and thus minimize cost and time.

Reference:
BigQuery ML
BigQuery ML for regression
BigQuery ML for scalability

Question: 102

You built a custom ML model using scikit-learn. Training time is taking longer than expected. You decide to migrate your model to Vertex AI Training, and you want to improve the model's training time. What should you try out first?

A. Migrate your model to TensorFlow, and train it using Vertex AI Training.
B. Train your model in a distributed mode using multiple Compute Engine VMs.
C. Train your model with DLVM images on Vertex AI, and ensure that your code utilizes NumPy and SciPy internal methods whenever possible.
D. Train your model using Vertex AI Training with GPUs.

Answer: D

Explanation:
Option A is incorrect because migrating your model to TensorFlow, and training it using Vertex AI Training, is not the easiest way to improve the model's training time. TensorFlow is a framework that allows you to create and train ML models using Python or other languages. Vertex AI Training is a service that allows you to train and optimize ML models using built-in algorithms or custom containers. However, this option requires significant code changes, as TensorFlow and scikit-learn have different APIs and functionalities. Moreover, this option does not leverage the parallelism or the scalability of the cloud, as it only uses a single instance.
Option B is incorrect because training your model in a distributed mode using multiple Compute Engine VMs, is not the most convenient way to improve the model's training time. Compute Engine is a service that allows you to create and manage virtual machines that run on Google Cloud. You can use Compute Engine to run your scikit-learn model in a

distributed mode, by using libraries such as Dask or Joblib. However, this option requires more effort and resources than option D, as it involves creating and configuring the VMs, installing and maintaining the libraries, and writing and running the distributed code.

Option C is incorrect because training your model with DLVM images on Vertex AI, and ensuring that your code utilizes NumPy and SciPy internal methods whenever possible, is not the most effective way to improve the model's training time. DLVM (Deep Learning Virtual Machine) images are preconfigured VM images that include popular ML frameworks and tools, such as TensorFlow, PyTorch, or scikit-learn1. You can use DLVM images on Vertex AI to train your scikit-learn model, by using a custom container. NumPy and SciPy are libraries that provide numerical and scientific computing functionalities for Python. You can use NumPy and SciPy internal methods to optimize your scikit-learn code, as they are faster and more efficient than pure Python code2. However, this option does not leverage the parallelism or the scalability of the cloud, as it only uses a single instance. Moreover, this option may not have a significant impact on the training time, as scikit-learn already relies on NumPy and SciPy for most of its operations3.

Option D is correct because training your model using Vertex AI Training with GPUs, is the best way to improve the model's training time. A GPU (Graphics Processing Unit) is a hardware accelerator that can perform parallel computations faster than a CPU (Central Processing Unit)4. Vertex AI

Training is a service that allows you to train and optimize ML models using built-in algorithms or custom containers. You can use Vertex AI Training with GPUs to train your scikit-learn model, by using a custom container and specifying the accelerator type and count5. By using Vertex AI Training with GPUs, you can leverage the parallelism and the scalability of the cloud, and speed up the training process significantly, without changing your code.

Reference:

DLVM images

NumPy and SciPy

scikit-learn dependencies

GPU overview

Vertex AI Training with GPUs

[scikit-learn overview]

[TensorFlow overview]

[Compute Engine overview]

[Dask overview]

[Joblib overview]

[Vertex AI Training overview]

Question: 103

You are an ML engineer at a travel company. You have been researching customers' travel behavior for many years, and you have deployed models that predict customers' vacation patterns. You have observed that customers' vacation destinations vary based on seasonality and holidays; however, these seasonal variations are similar across years. You want to quickly and easily store and compare the model versions and performance statistics across years. What should you do?

A. Store the performance statistics in Cloud SQL. Query that database to compare the performance statistics across the model versions.

B. Create versions of your models for each season per year in Vertex AI. Compare the performance statistics across the models in the Evaluate tab of the Vertex AI UI.

C. Store the performance statistics of each pipeline run in Kubeflow under an experiment for each season per year. Compare the results across the experiments in the Kubeflow UI.

D. Store the performance statistics of each version of your models using seasons and years as events in Vertex ML

Metadata. Compare the results across the slices.

Answer: D

Explanation:
Option A is incorrect because Cloud SQL is a relational database service that is not designed for storing and comparing model performance statistics. It would require writing complex SQL queries to perform the comparison, and it would not provide any visualization or analysis tools.
Option B is incorrect because Vertex AI does not support creating versions of models for each season per year. Vertex AI models are versioned based on the training data and hyperparameters, not on external factors such as seasonality or holidays. Moreover, the Evaluate tab of the Vertex AI UI only shows the performance metrics of a single model version, not across multiple versions.
Option C is incorrect because Kubeflow is a different platform than Vertex AI, and it does not integrate well with Vertex AI Pipelines. Kubeflow experiments are used to group pipeline runs that
share a common goal or objective, not to compare performance statistics across different seasons or years. Kubeflow UI does not provide any tools to compare the results across the experiments, and it would require switching between different platforms to access the data.
Option D is correct because Vertex ML Metadata is a service that allows storing and tracking metadata associated with machine learning workflows, such as models, datasets, metrics, and events. Events are user-defined labels that can be used to group or slice the metadata for analysis. By using seasons and years as events, you can easily store and compare the performance statistics of each version of your models across different time periods. Vertex ML Metadata also provides tools to visualize and analyze the metadata, such as the ML Metadata Explorer and the What-If Tool.

Question: 104

You are an ML engineer at a manufacturing company. You need to build a model that identifies defects in products based on images of the product taken at the end of the assembly line. You want your model to preprocess the images with lower computation to quickly extract features of defects in products. Which approach should you use to build the model?

A. Reinforcement learning
B. Recommender system
C. Recurrent Neural Networks (RNN)
D. Convolutional Neural Networks (CNN)

Answer: D

Explanation:
Option A is incorrect because reinforcement learning is not a suitable approach to build a model that identifies defects in products based on images of the product taken at the end of the assembly line. Reinforcement learning is a type of machine learning that learns from its own actions and rewards, rather than from labeled data or explicit feedback1. Reinforcement learning is more suitable for problems that involve sequential decision making, such as games, robotics, or control systems1. However, defect detection is a problem that involves image classification or segmentation, which requires supervised learning, not reinforcement learning.
Option B is incorrect because a recommender system is not a relevant approach to build a model that identifies defects in products based on images of the product taken at the end of the assembly line. A recommender system is a system that suggests items or actions to users based on their preferences, behavior, or context2. A recommender system is more suitable for problems that involve personalization, such as e-commerce, entertainment, or social media2. However,

defect detection is a problem that involves image classification or segmentation, which requires supervised learning, not recommender system.

Option C is incorrect because recurrent neural networks (RNN) are not the most efficient approach to build a model that identifies defects in products based on images of the product taken at the end of the assembly line. RNNs are a type of neural networks that can process sequential data, such as text, speech, or video, by maintaining a hidden state that captures the temporal dependencies3. RNNs are more suitable for problems that involve natural language processing, speech recognition, or video analysis3. However, defect detection is a problem that involves image classification or segmentation, which does not require temporal dependencies, but rather spatial dependencies. Moreover, RNNs are computationally expensive and prone to vanishing or exploding gradients4.

Option D is correct because convolutional neural networks (CNN) are the best approach to build a model that identifies defects in products based on images of the product taken at the end of the assembly line. CNNs are a type of neural networks that can process image data, by applying convolutional filters that extract local features and reduce the dimensionality of the data5. CNNs are more suitable for problems that involve image classification, object detection, or segmentation5. CNNs can preprocess the images with lower computation to quickly extract features of defects in products, by using techniques such as pooling, dropout, or batch normalization6.

Reference:

Reinforcement learning
Recommender system
Recurrent neural network
Vanishing and exploding gradients
Convolutional neural network
CNN techniques
[Defect detection]
[Image classification]
[Image segmentation]

Question: 105

You have successfully deployed to production a large and complex TensorFlow model trained on tabular dat a. You want to predict the lifetime value (LTV) field for each subscription stored in the BigQuery table named subscription. subscriptionPurchase in the project named my-fortune500-company-project. You have organized all your training code, from preprocessing data from the BigQuery table up to deploying the validated model to the Vertex AI endpoint, into a TensorFlow Extended (TFX) pipeline. You want to prevent prediction drift, i.e., a situation when a feature data distribution in production changes significantly over time. What should you do?

A. Implement continuous retraining of the model daily using Vertex AI Pipelines.
B. Add a model monitoring job where 10% of incoming predictions are sampled 24 hours.
C. Add a model monitoring job where 90% of incoming predictions are sampled 24 hours.
D. Add a model monitoring job where 10% of incoming predictions are sampled every hour.

Answer: B

Explanation:

Option A is incorrect because implementing continuous retraining of the model daily using Vertex AI Pipelines is not the most efficient way to prevent prediction drift. Vertex AI Pipelines is a service that allows you to create and run scalable and portable ML pipelines on Google Cloud1. You can use Vertex AI Pipelines to retrain your model daily using the latest data from the BigQuery table. However, this option may be unnecessary or wasteful, as the data distribution may not change significantly every day, and retraining the model may consume a lot of resources and time.

Moreover, this option does not monitor the model performance or detect the prediction drift, which are essential steps for ensuring the quality and reliability of the model.

Option B is correct because adding a model monitoring job where 10% of incoming predictions are sampled 24 hours is the best way to prevent prediction drift. Model monitoring is a service that allows you to track the performance and health of your deployed models over time2. You can use model monitoring to sample a fraction of the incoming predictions and compare them with the ground truth labels, which can be obtained from the BigQuery table or other sources. You can also use model monitoring to compute various metrics, such as accuracy, precision, recall, or F1-score, and set thresholds or alerts for them. By using model monitoring, you can detect and diagnose the prediction drift, and decide when to retrain or update your model. Sampling 10% of the incoming predictions every 24 hours is a reasonable choice, as it balances the trade-off between the accuracy and the cost of the monitoring job.

Option C is incorrect because adding a model monitoring job where 90% of incoming predictions are sampled 24 hours is not a optimal way to prevent prediction drift. This option has the same advantages as option B, as it uses model monitoring to track the performance and health of the deployed model. However, this option is not cost-effective, as it samples a very large fraction of the incoming predictions, which may incur a lot of storage and processing costs. Moreover, this option may not improve the accuracy of the monitoring job significantly, as sampling 10% of the incoming predictions may already provide a representative sample of the data distribution.

Option D is incorrect because adding a model monitoring job where 10% of incoming predictions are sampled every hour is not a necessary way to prevent prediction drift. This option also has the same advantages as option B, as it uses model monitoring to track the performance and health of the deployed model. However, this option may be excessive, as it samples the incoming predictions too frequently, which may not reflect the actual changes in the data distribution. Moreover, this option may incur more storage and processing costs than option B, as it generates more samples and metrics.

Reference:

Vertex AI Pipelines documentation
Model monitoring documentation

[Prediction drift]

[TensorFlow Extended documentation]

[BigQuery documentation]

[Vertex AI documentation]

Question: 106

You recently developed a deep learning model using Keras, and now you are experimenting with different training strategies. First, you trained the model using a single GPU, but the training process was too slow. Next, you distributed the training across 4 GPUs using tf.distribute.MirroredStrategy (with no other changes), but you did not observe a decrease in training time. What should you do?

A. Distribute the dataset with tf.distribute.Strategy.experimental_distribute_dataset
B. Create a custom training loop.
C. Use a TPU with tf.distribute.TPUStrategy.
D. Increase the batch size.

Answer: D

Explanation:

Option A is incorrect because distributing the dataset with tf.distribute.Strategy.experimental_distribute_dataset is not the most effective way to decrease the training time. This

method allows you to distribute your dataset across multiple devices or machines, by creating a tf.data.Dataset instance that can be iterated over in parallel1. However, this option may not improve the training time significantly, as it does not change the amount of data or computation that each device or machine has to process. Moreover, this option may introduce additional overhead or complexity, as it requires you to handle the data sharding, replication, and synchronization across the devices or machines1.

Option B is incorrect because creating a custom training loop is not the easiest way to decrease the training time. A custom training loop is a way to implement your own logic for training your model, by using low-level TensorFlow APIs, such as tf.GradientTape, tf.Variable, or tf.function2. A custom training loop may give you more flexibility and control over the training process, but it also requires more effort and expertise, as you have to write and debug the code for each step of the training loop, such as computing the gradients, applying the optimizer, or updating the metrics2. Moreover, a custom training loop may not improve the training time significantly, as it does not change the amount of data or computation that each device or machine has to process.

Option C is incorrect because using a TPU with tf.distribute.TPUStrategy is not a valid way to decrease the training time. A TPU (Tensor Processing Unit) is a custom hardware accelerator designed for high- performance ML workloads3. A tf.distribute.TPUStrategy is a distribution strategy that allows you to distribute your training across multiple TPUs, by creating a tf.distribute.TPUStrategy instance that can be used with high-level TensorFlow APIs, such as Keras4. However, this option is not feasible, as Vertex AI Training does not support TPUs as accelerators for custom training jobs5. Moreover, this option may require significant code changes, as TPUs have different requirements and limitations than GPUs.

Option D is correct because increasing the batch size is the best way to decrease the training time. The batch size is a hyperparameter that determines how many samples of data are processed in each iteration of the training loop. Increasing the batch size may reduce the training time, as it reduces the number of iterations needed to train the model, and it allows each device or machine to process more data in parallel. Increasing the batch size is also easy to implement, as it only requires changing a single hyperparameter. However, increasing the batch size may also affect the convergence and the accuracy of the model, so it is important to find the optimal batch size that balances the trade-off between the training time and the model performance.

Reference:

tf.distribute.Strategy.experimental_distribute_dataset

Custom training loop

TPU overview

tf.distribute.TPUStrategy

Vertex AI Training accelerators

[TPU programming model]

[Batch size and learning rate]

[Keras overview]

[tf.distribute.MirroredStrategy]

[Vertex AI Training overview]

[TensorFlow overview]

Question: 107

You work for a gaming company that has millions of customers around the world. All games offer a chat feature that allows players to communicate with each other in real time. Messages can be typed in more than 20 languages and are translated in real time using the Cloud Translation API. You have been asked to build an ML system to moderate the chat in real time while assuring that the performance is uniform across the various languages and without changing the serving infrastructure.

You trained your first model using an in-house word2vec model for embedding the chat messages translated by the Cloud Translation API. However, the model has significant differences in performance across the different languages.

How should you improve it?

A. Add a regularization term such as the Min-Diff algorithm to the loss function.
B. Train a classifier using the chat messages in their original language.
C. Replace the in-house word2vec with GPT-3 or T5.
D. Remove moderation for languages for which the false positive rate is too high.

Answer: B

Explanation:
The problem with the current approach is that it relies on the Cloud Translation API to translate the chat messages into a common language before embedding them with the in-house word2vec model. This introduces two sources of error: the translation quality and the word2vec quality. The translation quality may vary across different languages, depending on the availability of data and the complexity of the grammar and vocabulary. The word2vec quality may also vary depending on the size and diversity of the corpus used to train it. These errors may affect the performance of the classifier that moderates the chat messages, resulting in significant differences across the languages. A better approach would be to train a classifier using the chat messages in their original language, without relying on the Cloud Translation API or the in-house word2vec model. This way, the classifier can learn the nuances and subtleties of each language, and avoid the errors introduced by the translation and embedding processes. This would also reduce the latency and cost of the moderation system, as it would not need to invoke the Cloud Translation API for every message. To train a classifier using the chat messages in their original language, one could use a multilingual pre-trained model such as mBERT or XLM-R, which can handle multiple languages and domains. Alternatively, one could train a separate classifier for each language, using a monolingual pre-trained model such as BERT or a custom model tailored to the specific language and task.

Reference:
Professional ML Engineer Exam Guide
Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate Google Cloud launches machine learning engineer certification
[mBERT: Bidirectional Encoder Representations from Transformers] [XLM-R: Unsupervised Cross-lingual Representation Learning at Scale] [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding]

Question: 108

You work for a gaming company that develops massively multiplayer online (MMO) games. You built a TensorFlow model that predicts whether players will make in-app purchases of more than $10 in the next two weeks. The model's predictions will be used to adapt each user's game experience. User data is stored in BigQuery. How should you serve your model while optimizing cost, user experience, and ease of management?

A. Import the model into BigQuery ML. Make predictions using batch reading data from BigQuery, and push the data to Cloud SQL
B. Deploy the model to Vertex AI Prediction. Make predictions using batch reading data from Cloud Bigtable, and push the data to Cloud SQL.
C. Embed the model in the mobile application. Make predictions after every in-app purchase event is published in Pub/Sub, and push the data to Cloud SQL.
D. Embed the model in the streaming Dataflow pipeline. Make predictions after every in-app purchase event is published in Pub/Sub, and push the data to Cloud SQL.

Answer: B

Explanation:

The best option to serve the model while optimizing cost, user experience, and ease of management is to deploy the model to Vertex AI Prediction, which is a managed service that can scale up or down according to the demand and provide low latency and high availability. Vertex AI Prediction can also handle TensorFlow models natively, without requiring any additional steps or conversions. By using batch prediction, the model can process large volumes of data efficiently and periodically, without affecting the user experience. The data can be read from Cloud Bigtable, which is a scalable and performant NoSQL database that can store user data in a flexible schema. The predictions can then be pushed to Cloud SQL, which is a fully managed relational database that can store the predictions in a structured format and enable easy querying and analysis. This option also simplifies the management of the model and the data, as it leverages the existing Google Cloud services and does not require any additional infrastructure or code.

The other options are not optimal for the following reasons:

A . Importing the model into BigQuery ML is not a good option, as it requires converting the TensorFlow model into a format that BigQuery ML can understand, which can introduce errors and reduce the performance. Moreover, BigQuery ML is not designed for serving real-time predictions, but rather for training and evaluating models using SQL queries. Reading and writing data from BigQuery and Cloud SQL can also incur additional costs and latency, as they are both relational databases that require schema definition and data transformation.

C . Embedding the model in the mobile application is not a good option, as it increases the size and complexity of the application, and requires updating the application every time the model changes. Moreover, it exposes the model to the users, which can pose security and privacy risks, as well as potential misuse or abuse. Additionally, it does not leverage the benefits of the cloud, such as scalability, reliability, and performance.

D . Embedding the model in the streaming Dataflow pipeline is not a good option, as it requires building and maintaining a custom pipeline that can handle the model inference and data processing. This can increase the development and operational costs and complexity, as well as the potential for errors and failures. Moreover, it does not take advantage of the batch prediction feature of Vertex AI Prediction, which can optimize the resource utilization and cost efficiency.

Reference:

Professional ML Engineer Exam Guide

Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate Google Cloud launches machine learning engineer certification

Vertex AI Prediction documentation

Cloud Bigtable documentation

Cloud SQL documentation

Question: 109

You are experimenting with a built-in distributed XGBoost model in Vertex AI Workbench usermanaged notebooks. You use BigQuery to split your data into training and validation sets using the following queries:

```
CREATE OR REPLACE TABLE 'myproject.mydataset.training' AS
(SELECT * FROM 'myproject.mydataset.mytable' WHERE RAND() < 0.8);
CREATE OR REPLACE TABLE 'myproject.mydataset.validation' AS
(SELECT * FROM 'myproject.mydataset.mytable' WHERE RAND() < 0.2);
```

After training the model, you achieve an area under the receiver operating characteristic curve (AUC ROC) value of 0.8, but after deploying the model to production, you notice that your model performance has dropped to an AUC ROC value of 0.65. What problem is most likely occurring?

A. There is training-serving skew in your production environment.

B. There is not a sufficient amount of training data.

C. The tables that you created to hold your training and validation records share some records, and you may not be

using all the data in your initial table.

D. The RAND() function generated a number that is less than 0.2 in both instances, so every record in the validation table will also be in the training table.

Answer: C

Explanation:

The most likely problem is that the tables that you created to hold your training and validation records share some records, and you may not be using all the data in your initial table. This is because the RAND() function generates a random number between 0 and 1 for each row, and the probability of a row being in both the training and validation tables is 0.2 * 0.8 0.16, which is not negligible. This means that some of the records that you use to validate your model are also used to train your model, which can lead to overfitting and poor generalization. Moreover, the probability of a row being in neither the training nor the validation table is 0.2 * 0.2 0.04, which means that you are wasting some of the data in your initial table and reducing the size of your datasets. A better way to split your data into training and validation sets is to use a hash function on a unique identifier column, such as the following queries:

CREATE OR REPLACE TABLE 'myproject.mydataset.training' AS (SELECT * FROM 'myproject.mydataset.mytable' WHERE MOD(FARM_FINGERPRINT(id), 10) < 8); CREATE OR REPLACE TABLE 'myproject.mydataset.validation' AS (SELECT * FROM 'myproject.mydataset.mytable' WHERE MOD(FARM_FINGERPRINT(id), 10) > 8);

This way, you can ensure that each row has a fixed 80% chance of being in the training table and a 20% chance of being in the validation table, without any overlap or omission.

Reference:

Professional ML Engineer Exam Guide

Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate Google Cloud launches machine learning engineer certification

BigQuery ML: Splitting data for training and testing

BigQuery: FARM_FINGERPRINT function

Question: 110

You need to analyze user activity data from your company's mobile applications. Your team will use BigQuery for data analysis, transformation, and experimentation with ML algorithms. You need to ensure real-time ingestion of the user activity data into BigQuery. What should you do?

A. Configure Pub/Sub to stream the data into BigQuery.

B. Run an Apache Spark streaming job on Dataproc to ingest the data into BigQuery.

C. Run a Dataflow streaming job to ingest the data into BigQuery.

D. Configure Pub/Sub and a Dataflow streaming job to ingest the data into BigQuery,

Answer: C

Explanation:

The best option to ensure real-time ingestion of the user activity data into BigQuery is to run a Dataflow streaming job to ingest the data into BigQuery. Dataflow is a fully managed service that can handle both batch and stream processing of data, and can integrate seamlessly with BigQuery and other Google Cloud services. Dataflow can also use Apache Beam as the programming model, which provides a unified and portable API for developing data pipelines. By using Dataflow, you can avoid the complexity and overhead of managing your own infrastructure, and focus on the logic and transformation of your data. Dataflow can also handle various types of data, such as structured, unstructured, or binary data, and can apply windowing, aggregation, and other operations on the data streams.

The other options are not optimal for the following reasons:

A . Configuring Pub/Sub to stream the data into BigQuery is not a good option, as Pub/Sub is a messaging service that

can publish and subscribe to data streams, but cannot perform any transformation or processing on the data. Pub/Sub can be used as a source or a sink for Dataflow, but not as a standalone solution for ingesting data into BigQuery.

B . Running an Apache Spark streaming job on Dataproc to ingest the data into BigQuery is not a good option, as it requires setting up and managing your own cluster of virtual machines, which can increase the cost and complexity of your solution. Moreover, Apache Spark is not natively integrated with BigQuery, and requires using connectors or intermediate storage to write data to BigQuery, which can introduce latency and inefficiency.

D . Configuring Pub/Sub and a Dataflow streaming job to ingest the data into BigQuery is not a bad option, but it is not necessary, as Dataflow can directly read data from the mobile applications without using Pub/Sub as an intermediary. Using Pub/Sub can add an extra layer of abstraction and reliability, but it can also increase the cost and complexity of your solution, and introduce some delay in the data ingestion.

Reference:

Professional ML Engineer Exam Guide

Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate Google Cloud launches machine learning engineer certification Dataflow documentation

BigQuery documentation

Question: 111

You work for a gaming company that manages a popular online multiplayer game where teams with 6 players play against each other in 5-minute battles. There are many new players every day. You need to build a model that automatically assigns available players to teams in real time. User research indicates that the game is more enjoyable when battles have players with similar skill levels. Which business metrics should you track to measure your model's performance? (Choose One Correct Answer)

A. Average time players wait before being assigned to a team

B. Precision and recall of assigning players to teams based on their predicted versus actual ability

C. User engagement as measured by the number of battles played daily per user

D. Rate of return as measured by additional revenue generated minus the cost of developing a new model

Answer: C

Explanation:

The best business metric to track to measure the model's performance is user engagement as measured by the number of battles played daily per user. This metric reflects the main goal of the model, which is to enhance the user experience and satisfaction by creating balanced and fair battles. If the model is successful, it should increase the user retention and loyalty, as well as the word-of-mouth and referrals. This metric is also easy to measure and interpret, as it can be directly obtained from the user activity data.

The other options are not optimal for the following reasons:

A . Average time players wait before being assigned to a team is not a good metric, as it does not capture the quality or outcome of the battles. It only measures the efficiency of the model, which is not the primary objective. Moreover, this metric can be influenced by external factors, such as the availability and demand of players, the network latency, and the server capacity.

B . Precision and recall of assigning players to teams based on their predicted versus actual ability is not a good metric, as it is difficult to measure and interpret. It requires having a reliable and consistent way of estimating the player's ability, which can be subjective and dynamic. It also requires having a ground truth label for each assignment, which can be costly and impractical to obtain. Moreover, this metric does not reflect the user feedback or satisfaction, which is the ultimate goal of the model.

D . Rate of return as measured by additional revenue generated minus the cost of developing a new model is not a good

metric, as it is not directly related to the model's performance. It measures the profitability of the model, which is a secondary objective. Moreover, this metric can be affected by many other factors, such as the market conditions, the pricing strategy, the marketing campaigns, and the competition.

Reference:

Professional ML Engineer Exam Guide

Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate Google Cloud launches machine learning engineer certification

How to measure user engagement

How to choose the right metrics for your machine learning model

Question: 112

You are building an ML model to predict trends in the stock market based on a wide range of factors. While exploring the data, you notice that some features have a large range. You want to ensure that the features with the largest magnitude don't overfit the model. What should you do?

A. Standardize the data by transforming it with a logarithmic function.
B. Apply a principal component analysis (PCA) to minimize the effect of any particular feature.
C. Use a binning strategy to replace the magnitude of each feature with the appropriate bin number.
D. Normalize the data by scaling it to have values between 0 and 1.

Answer: D

Explanation:

The best option to ensure that the features with the largest magnitude don't overfit the model is to normalize the data by scaling it to have values between 0 and 1. This is also known as min-max scaling or feature scaling, and it can reduce the variance and skewness of the data, as well as improve the numerical stability and convergence of the model. Normalizing the data can also make the model less sensitive to the scale of the features, and more focused on the relative importance of each feature. Normalizing the data can be done using various methods, such as dividing each value by the maximum value, subtracting the minimum value and dividing by the range, or using the sklearn.preprocessing.MinMaxScaler function in Python.

The other options are not optimal for the following reasons:

A . Standardizing the data by transforming it with a logarithmic function is not a good option, as it can distort the distribution and relationship of the data, and introduce bias and errors. Moreover, the logarithmic function is not defined for negative or zero values, which can limit its applicability and cause problems for the model.

B . Applying a principal component analysis (PCA) to minimize the effect of any particular feature is not a good option, as it can reduce the interpretability and explainability of the data and the model. PCA is a dimensionality reduction technique that transforms the data into a new set of orthogonal features that capture the most variance in the data. However, these new features are not directly related to the original features, and can lose some information and meaning in the process. Moreover, PCA can be computationally expensive and complex, and may not be necessary for the problem at hand.

C . Using a binning strategy to replace the magnitude of each feature with the appropriate bin number is not a good option, as it can lose the granularity and precision of the data, and introduce noise and outliers. Binning is a discretization technique that groups the continuous values of a feature into a finite number of bins or categories. However, this can reduce the variability and diversity of the data, and create artificial boundaries and gaps that may not reflect the true nature of the data. Moreover, binning can be arbitrary and subjective, and depend on the choice of the bin size and number.

Reference:

Professional ML Engineer Exam Guide

Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate Google Cloud launches machine learning engineer certification

Feature Scaling for Machine Learning: Understanding the Difference Between Normalization vs. Standardization

sklearn.preprocessing.MinMaxScaler documentation

Principal Component Analysis Explained Visually

Binning Data in Python

Question: 113

You work for a biotech startup that is experimenting with deep learning ML models based on properties of biological organisms. Your team frequently works on early-stage experiments with new architectures of ML models, and writes custom TensorFlow ops in C++. You train your models on large datasets and large batch sizes. Your typical batch size has 1024 examples, and each example is about 1 MB in size. The average size of a network with all weights and embeddings is 20 GB. What hardware should you choose for your models?

A. A cluster with 2 n1-highcpu-64 machines, each with 8 NVIDIA Tesla V100 GPUs (128 GB GPU memory in total), and a n1-highcpu-64 machine with 64 vCPUs and 58 GB RAM

B. A cluster with 2 a2-megagpu-16g machines, each with 16 NVIDIA Tesla A100 GPUs (640 GB GPU memory in total), 96 vCPUs, and 1.4 TB RAM

C. A cluster with an n1-highcpu-64 machine with a v2-8 TPU and 64 GB RAM

D. A cluster with 4 n1-highcpu-96 machines, each with 96 vCPUs and 86 GB RAM

Answer: B

Explanation:

The best hardware to choose for your models is a cluster with 2 a2-megagpu-16g machines, each with 16 NVIDIA Tesla A100 GPUs (640 GB GPU memory in total), 96 vCPUs, and 1.4 TB RAM. This hardware configuration can provide you with enough compute power, memory, and bandwidth to handle your large and complex deep learning models, as well as your custom TensorFlow ops in C++. The NVIDIA Tesla A100 GPUs are the latest and most advanced GPUs from NVIDIA, which offer high performance, scalability, and efficiency for various ML workloads. They also support multi-instance GPU (MIG) technology, which allows you to partition each GPU into up to seven smaller instances, each with its own memory, cache, and compute cores. This can enable you to run multiple experiments in parallel, or to optimize the resource utilization and cost efficiency of your models. The a2-megagpu-16g machines are part of the Google Cloud Accelerator-Optimized VM (A2) family, which are designed to provide the best performance and flexibility for GPU-intensive applications. They also offer high-speed NVLink interconnects between the GPUs, which can improve the data transfer and communication between the GPUs. Moreover, the a2-megagpu-16g machines have 96 vCPUs and 1.4 TB RAM, which can support the CPU and memory requirements of your models, as well as the data preprocessing and postprocessing tasks.

The other options are not optimal for the following reasons:

A . A cluster with 2 n1-highcpu-64 machines, each with 8 NVIDIA Tesla V100 GPUs (128 GB GPU memory in total), and a n1-highcpu-64 machine with 64 vCPUs and 58 GB RAM is not a good option, as it has less GPU memory, compute power, and bandwidth than the a2-megagpu-16g machines. The NVIDIA Tesla V100 GPUs are the previous generation of GPUs from NVIDIA, which have lower performance, scalability, and efficiency than the NVIDIA Tesla A100 GPUs. They also do not support the MIG technology, which can limit the flexibility and optimization of your models. Moreover, the n1-highcpu-64 machines are part of the Google Cloud N1 VM family, which are general-purpose VMs that do not offer the best performance and features for GPU-intensive applications. They also have lower vCPUs and RAM than the a2-megagpu-16g machines, which can affect the CPU and memory requirements of your models, as well as the data

preprocessing and postprocessing tasks.

C . A cluster with an n1-highcpu-64 machine with a v2-8 TPU and 64 GB RAM is not a good option, as it has less GPU memory, compute power, and bandwidth than the a2-megagpu-16g machines. The v2-8 TPU is a cloud tensor processing unit (TPU) device, which is a custom ASIC chip designed by Google to accelerate ML workloads. However, the v2-8 TPU is the second generation of TPUs, which have lower performance, scalability, and efficiency than the latest v3-8 TPUs. They also have less memory and bandwidth than the NVIDIA Tesla A100 GPUs, which can limit the size and complexity of your models, as well as the data transfer and communication between the devices. Moreover, the n1-highcpu-64 machine has lower vCPUs and RAM than the a2-megagpu-16g machines, which can affect the CPU and memory requirements of your models, as well as the data preprocessing and postprocessing tasks.

D . A cluster with 4 n1-highcpu-96 machines, each with 96 vCPUs and 86 GB RAM is not a good option, as it does not have any GPUs, which are essential for accelerating deep learning models. The n1-highcpu-96 machines are part of the Google Cloud N1 VM family, which are general-purpose VMs that do not offer the best performance and features for GPU-intensive applications. They also have lower RAM than the a2-megagpu-16g machines, which can affect the memory requirements of your models, as well as the data preprocessing and postprocessing tasks.

Reference:

Professional ML Engineer Exam Guide

Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate Google Cloud launches machine learning engineer certification NVIDIA Tesla A100 GPU

Google Cloud Accelerator-Optimized VM (A2) family

Google Cloud N1 VM family Cloud TPU

Question: 114

You are an ML engineer at an ecommerce company and have been tasked with building a model that predicts how much inventory the logistics team should order each month. Which approach should you take?

A. Use a clustering algorithm to group popular items together. Give the list to the logistics team so they can increase inventory of the popular items.

B. Use a regression model to predict how much additional inventory should be purchased each month. Give the results to the logistics team at the beginning of the month so they can increase inventory by the amount predicted by the model.

C. Use a time series forecasting model to predict each item's monthly sales. Give the results to the logistics team so they can base inventory on the amount predicted by the model.

D. Use a classification model to classify inventory levels as UNDER_STOCKED, OVER_STOCKED, and CORRECTLY_STOCKED. Give the report to the logistics team each month so they can fine-tune inventory levels.

Answer: C

Explanation:

The best approach to build a model that predicts how much inventory the logistics team should order each month is to use a time series forecasting model to predict each item's monthly sales. This approach can capture the temporal patterns and trends in the sales data, such as seasonality, cyclicality, and autocorrelation. It can also account for the variability and uncertainty in the demand, and provide confidence intervals and error metrics for the predictions. By using a time series forecasting model, you can provide the logistics team with accurate and reliable estimates of the future sales for each item, which can help them optimize the inventory levels and avoid overstocking or understocking. You can use various methods and tools to build a time series forecasting model, such as ARIMA, LSTM, Prophet, or BigQuery ML.

The other options are not optimal for the following reasons:

A . Using a clustering algorithm to group popular items together is not a good approach, as it does not provide any quantitative or temporal information about the sales or the inventory. It only provides a qualitative and static categorization of the items based on their similarity or dissimilarity. Moreover, clustering is an unsupervised learning technique, which does not use any target variable or feedback to guide the learning process. This can result in arbitrary and inconsistent clusters, which may not reflect the true demand or preferences of the customers.

B . Using a regression model to predict how much additional inventory should be purchased each month is not a good approach, as it does not account for the individual differences and dynamics of each item. It only provides a single aggregated value for the whole inventory, which can be misleading and inaccurate. Moreover, a regression model is not well-suited for handling time series data, as it assumes that the data points are independent and identically distributed, which is not the case for sales data. A regression model can also suffer from overfitting or underfitting, depending on the choice and complexity of the features and the model.

D . Using a classification model to classify inventory levels as UNDER_STOCKED, OVER_STOCKED, and CORRECTLY_STOCKED is not a good approach, as it does not provide any numerical or predictive information about the sales or the inventory. It only provides a discrete and subjective label for the inventory levels, which can be vague and ambiguous. Moreover, a classification model is not well- suited for handling time series data, as it assumes that the data points are independent and identically distributed, which is not the case for sales data. A classification model can also suffer from class imbalance, misclassification, or overfitting, depending on the choice and complexity of the features, the model, and the threshold.

Reference:

Professional ML Engineer Exam Guide

Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate Google Cloud launches machine learning engineer certification

Time Series Forecasting: Principles and Practice

BigQuery ML: Time series analysis

Question: 115

You are building a TensorFlow model for a financial institution that predicts the impact of consumer spending on inflation globally. Due to the size and nature of the data, your model is long-running across all types of hardware, and you have built frequent checkpointing into the training process.
Your organization has asked you to minimize cost. What hardware should you choose?

A. A Vertex AI Workbench user-managed notebooks instance running on an n1-standard-16 with 4 NVIDIA P100 GPUs

B. A Vertex AI Workbench user-managed notebooks instance running on an n1-standard-16 with an NVIDIA P100 GPU

C. A Vertex AI Workbench user-managed notebooks instance running on an n1-standard-16 with a non-preemptible v3-8 TPU

D. A Vertex AI Workbench user-managed notebooks instance running on an n1-standard-16 with a preemptible v3-8 TPU

Answer: D

Explanation:

The best hardware to choose for your model while minimizing cost is a Vertex AI Workbench usermanaged notebooks instance running on an n1-standard-16 with a preemptible v3-8 TPU. This hardware configuration can provide you with

high performance, scalability, and efficiency for your TensorFlow model, as well as low cost and flexibility for your long-running and checkpointing process. The v3-8 TPU is a cloud tensor processing unit (TPU) device, which is a custom ASIC chip designed by Google to accelerate ML workloads. It can handle large and complex models and datasets, and offer fast and stable training and inference. The n1-standard-16 is a general-purpose VM that can support the CPU and memory requirements of your model, as well as the data preprocessing and postprocessing tasks. By choosing a preemptible v3-8 TPU, you can take advantage of the lower price and availability of the TPU devices, as long as you can tolerate the possibility of the device being reclaimed by Google at any time. However, since you have built frequent checkpointing into your training process, you can resume your model from the last saved state, and avoid losing any progress or data. Moreover, you can use the Vertex AI Workbench user-managed notebooks to create and manage your notebooks instances, and leverage the integration with Vertex AI and other Google Cloud services.

The other options are not optimal for the following reasons:

A . A Vertex AI Workbench user-managed notebooks instance running on an n1-standard-16 with 4 NVIDIA P100 GPUs is not a good option, as it has higher cost and lower performance than the v3-8 TPU. The NVIDIA P100 GPUs are the previous generation of GPUs from NVIDIA, which have lower performance, scalability, and efficiency than the latest NVIDIA A100 GPUs or the TPUs. They also have higher price and lower availability than the preemptible TPUs, which can increase the cost and complexity of your solution.

B . A Vertex AI Workbench user-managed notebooks instance running on an n1-standard-16 with an NVIDIA P100 GPU is not a good option, as it has higher cost and lower performance than the v3-8 TPU. It also has less GPU memory and compute power than the option with 4 NVIDIA P100 GPUs, which can limit the size and complexity of your model, and affect the training and inference speed and quality.

C . A Vertex AI Workbench user-managed notebooks instance running on an n1-standard-16 with a non-preemptible v3-8 TPU is not a good option, as it has higher cost and lower flexibility than the preemptible v3-8 TPU. The non-preemptible v3-8 TPU has the same performance, scalability, and efficiency as the preemptible v3-8 TPU, but it has higher price and lower availability, as it is reserved for your exclusive use. Moreover, since your model is long-running and checkpointing, you do not need the guarantee of the device not being reclaimed by Google, and you can benefit from the lower cost and higher availability of the preemptible v3-8 TPU.

Reference:

Professional ML Engineer Exam Guide

Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate Google Cloud launches machine learning engineer certification

Cloud TPU

Vertex AI Workbench user-managed notebooks

Preemptible VMs

NVIDIA Tesla P100 GPU

Question: 116

You work for a company that provides an anti-spam service that flags and hides spam posts on social media platforms. Your company currently uses a list of 200,000 keywords to identify suspected spam posts. If a post contains more than a few of these keywords, the post is identified as spam. You want to start using machine learning to flag spam posts for human review. What is the main advantage of implementing machine learning for this business case?

A. Posts can be compared to the keyword list much more quickly.
B. New problematic phrases can be identified in spam posts.
C. A much longer keyword list can be used to flag spam posts.
D. Spam posts can be flagged using far fewer keywords.

Answer: B

Explanation:

The main advantage of implementing machine learning for this business case is that new problematic phrases can be identified in spam posts. This is because machine learning can learn from the data and the feedback, and adapt to the changing patterns and trends of spam posts. Machine learning can also capture the semantic and contextual meaning of the posts, and not just rely on the presence or absence of keywords. By using machine learning, you can improve the accuracy and coverage of your anti-spam service, and detect new and emerging types of spam posts that may not be captured by the keyword list.

The other options are not advantages of implementing machine learning for this business case for the following reasons:

A . Posts can be compared to the keyword list much more quickly is not an advantage, as it does not improve the quality or effectiveness of the anti-spam service. It only improves the efficiency of the service, which is not the primary objective. Moreover, machine learning may not necessarily be faster than the keyword list, depending on the complexity and size of the model and the data. C . A much longer keyword list can be used to flag spam posts is not an advantage, as it does not address the limitations or challenges of the keyword list approach. It only increases the size and complexity of the keyword list, which can make it harder to maintain and update. Moreover, a longer keyword list may not improve the accuracy or coverage of the anti-spam service, as it may introduce more false positives or false negatives, or miss new and emerging types of spam posts.

D . Spam posts can be flagged using far fewer keywords is not an advantage, as it does not reflect the capabilities or benefits of machine learning. It only reduces the size and complexity of the keyword list, which can make it easier to maintain and update. However, using fewer keywords may not improve the accuracy or coverage of the anti-spam service, as it may lose some information or meaning of the posts, or miss some types of spam posts.

Reference:

Professional ML Engineer Exam Guide

Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate Google Cloud launches machine learning engineer certification

Machine Learning for Spam Detection

Spam Detection Using Machine Learning

Question: 117

One of your models is trained using data provided by a third-party data broker. The data broker does not reliably notify you of formatting changes in the dat

a. You want to make your model training pipeline more robust to issues like this. What should you do?

A. Use TensorFlow Data Validation to detect and flag schema anomalies.

B. Use TensorFlow Transform to create a preprocessing component that will normalize data to the expected distribution, and replace values that don't match the schema with 0.

C. Use tf.math to analyze the data, compute summary statistics, and flag statistical anomalies.

D. Use custom TensorFlow functions at the start of your model training to detect and flag known formatting errors.

Answer: A

Explanation:

TensorFlow Data Validation (TFDV) is a library that helps you understand, validate, and monitor your data for machine learning. It can automatically detect and report schema anomalies, such as missing features, new features, or different

data types, in your data. It can also generate descriptive statistics and data visualizations to help you explore and debug your data. TFDV can be integrated with your model training pipeline to ensure data quality and consistency throughout the machine learning lifecycle. Reference:

TensorFlow Data Validation

Data Validation | TensorFlow

Data Validation | Machine Learning Crash Course | Google Developers

Question: 118

You work for a company that is developing a new video streaming platform. You have been asked to create a recommendation system that will suggest the next video for a user to watch. After a review by an AI Ethics team, you are approved to start development. Each video asset in your company's catalog has useful metadata (e.g., content type, release date, country), but you do not have any historical user event dat

a. How should you build the recommendation system for the first version of the product?

A. Launch the product without machine learning. Present videos to users alphabetically, and start collecting user event data so you can develop a recommender model in the future.

B. Launch the product without machine learning. Use simple heuristics based on content metadata to recommend similar videos to users, and start collecting user event data so you can develop a recommender model in the future.

C. Launch the product with machine learning. Use a publicly available dataset such as MovieLens to train a model using the Recommendations AI, and then apply this trained model to your data.

D. Launch the product with machine learning. Generate embeddings for each video by training an autoencoder on the content metadata using TensorFlow. Cluster content based on the similarity of these embeddings, and then recommend videos from the same cluster.

Answer: B

Explanation:

The best option for building a recommendation system without any user event data is to use simple heuristics based on content metadata. This is a type of content-based filtering, which recommends items that are similar to the ones that the user has interacted with or selected, based on their attributes. For example, if a user selects a comedy movie from the US released in 2020, the system can recommend other comedy movies from the US released in 2020 or nearby years. This approach does not require any machine learning, but it can leverage the existing metadata of the videos to provide relevant recommendations. It also allows the system to start collecting user event data, such as views, likes, ratings, etc., which can be used to train a more sophisticated machine learning model in the future, such as a collaborative filtering model or a hybrid model that combines content and collaborative information. Reference:

Recommendation Systems

Content-Based Filtering

Collaborative Filtering

Hybrid Recommender Systems: A Systematic Literature Review

Question: 119

You recently built the first version of an image segmentation model for a self-driving car. After deploying the model, you observe a decrease in the area under the curve (AUC) metric. When analyzing the video recordings, you also discover that the model fails in highly congested traffic but works as expected when there is less traffic. What is the most likely reason for this result?

A. The model is overfitting in areas with less traffic and underfitting in areas with more traffic.
B. AUC is not the correct metric to evaluate this classification model.
C. Too much data representing congested areas was used for model training.
D. Gradients become small and vanish while backpropagating from the output to input nodes.

Answer: A

Explanation:
The most likely reason for the observed result is that the model is overfitting in areas with less traffic and underfitting in areas with more traffic. Overfitting means that the model learns the specific patterns and noise in the training data, but fails to generalize well to new and unseen data. Underfitting means that the model is not able to capture the complexity and variability of the data, and performs poorly on both training and test data. In this case, the model might have learned to segment the images well when there is less traffic, but it might not have enough data or features to handle the more challenging scenarios when there is more traffic. This could lead to a decrease in the AUC metric, which measures the ability of the model to distinguish between different classes. AUC is a suitable metric for this classification model, as it is not affected by class imbalance or threshold selection. The other options are not likely to be the reason for the result, as they are not related to the traffic density. Too much data representing congested areas would not cause the

model to fail in those areas, but rather help the model learn better. Gradients vanishing or exploding is a problem that occurs during the training process, not after the deployment, and it affects the whole model, not specific scenarios. Reference:

Image Segmentation: U-Net For Self Driving Cars
Intelligent Semantic Segmentation for Self-Driving Vehicles Using Deep Learning
Sharing Pixelopolis, a self-driving car demo from Google I/O built with TensorFlow Lite
Google Cloud launches machine learning engineer certification
Google Professional Machine Learning Engineer Certification
Professional ML Engineer Exam Guide
Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate

Question: 120

You are developing an ML model to predict house prices. While preparing the data, you discover that an important predictor variable, distance from the closest school, is often missing and does not have high variance. Every instance (row) in your data is important. How should you handle the missing data?

A. Delete the rows that have missing values.
B. Apply feature crossing with another column that does not have missing values.
C. Predict the missing values using linear regression.
D. Replace the missing values with zeros.

Answer: C

Explanation:
The best option for handling missing data in this case is to predict the missing values using linear regression. Linear regression is a supervised learning technique that can be used to estimate the relationship between a continuous target variable and one or more predictor variables. In this case, the target variable is the distance from the closest school, and the predictor variables are the other features in the dataset, such as house size, location, number of rooms, etc. By

fitting a linear regression model on the data that has no missing values, we can then use the model to predict the missing values for the distance from the closest school feature. This way, we can preserve all the instances in the dataset and avoid introducing bias or reducing variance. The other options are not suitable for handling missing data in this case, because:

Deleting the rows that have missing values would reduce the size of the dataset and potentially lose important information. Since every instance is important, we want to keep as much data as possible. Applying feature crossing with another column that does not have missing values would create a new feature that combines the values of two existing features. This might increase the complexity of the model and introduce noise or multicollinearity. It would not solve the problem of missing values, as the new feature would still have missing values whenever the distance from the closest school feature is missing.

Replacing the missing values with zeros would distort the distribution of the feature and introduce bias. It would also imply that the houses with missing values are located at the same distance from the closest school, which is unlikely to be true. A zero value might also be outside the range of the feature, as the distance from the closest school is unlikely to be exactly zero for any house. Reference:

Linear Regression
Imputation of missing values
Google Cloud launches machine learning engineer certification
Google Professional Machine Learning Engineer Certification
Professional ML Engineer Exam Guide
Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate

Question: 121

You are an ML engineer responsible for designing and implementing training pipelines for ML models. You need to create an end-to-end training pipeline for a TensorFlow model. The TensorFlow model will be trained on several terabytes of structured dat

a. You need the pipeline to include data quality checks before training and model quality checks after training but prior to deployment. You want to minimize development time and the need for infrastructure maintenance. How should you build and orchestrate your training pipeline?

A. Create the pipeline using Kubeflow Pipelines domain-specific language (DSL) and predefined Google Cloud components. Orchestrate the pipeline using Vertex AI Pipelines.

B. Create the pipeline using TensorFlow Extended (TFX) and standard TFX components. Orchestrate the pipeline using Vertex AI Pipelines.

C. Create the pipeline using Kubeflow Pipelines domain-specific language (DSL) and predefined Google Cloud components. Orchestrate the pipeline using Kubeflow Pipelines deployed on Google Kubernetes Engine.

D. Create the pipeline using TensorFlow Extended (TFX) and standard TFX components. Orchestrate the pipeline using Kubeflow Pipelines deployed on Google Kubernetes Engine.

Answer: B

Explanation:

The best option for creating and orchestrating an end-to-end training pipeline for a TensorFlow model is to use TensorFlow Extended (TFX) and standard TFX components, and deploy the pipeline to Vertex AI Pipelines. TFX is an end-to-end platform for deploying production ML pipelines, which consists of several built-in components that cover the entire ML lifecycle, from data ingestion and validation, to model training and evaluation, to model deployment and monitoring. TFX also supports custom components and integrations with other Google Cloud services, such as BigQuery, Dataflow, and Cloud Storage. Vertex AI Pipelines is a fully managed service that allows you to run TFX pipelines on Google Cloud, without having to worry about infrastructure provisioning, scaling, or maintenance. Vertex AI Pipelines

also provides a user-friendly interface to monitor and manage your pipelines, as well as tools to track and compare experiments. The other options are not as suitable for creating and orchestrating an end-to-end training pipeline for a TensorFlow model, because: Creating the pipeline using Kubeflow Pipelines domain-specific language (DSL) and predefined Google Cloud components would require more development time and effort, as Kubeflow Pipelines DSL is not as expressive or compatible with TensorFlow as TFX. Predefined Google Cloud components might not cover all the stages of the ML lifecycle, and might not be optimized for TensorFlow models. Orchestrating the pipeline using Kubeflow Pipelines deployed on Google Kubernetes Engine would require more infrastructure maintenance, as Kubeflow Pipelines is not a fully managed service, and you would have to provision and manage your own Kubernetes cluster. This would also incur more costs, as you would have to pay for the cluster resources, regardless of the pipeline usage. Reference:

TFX | ML Production Pipelines | TensorFlow

Vertex AI Pipelines | Google Cloud

Kubeflow Pipelines | Google Cloud

Google Cloud launches machine learning engineer certification

Google Professional Machine Learning Engineer Certification

Professional ML Engineer Exam Guide

Question: 122

You manage a team of data scientists who use a cloud-based backend system to submit training jobs. This system has become very difficult to administer, and you want to use a managed service instead. The data scientists you work with use many different frameworks, including Keras, PyTorch, theano, scikit-learn, and custom libraries. What should you do?

A. Use the Vertex AI Training to submit training jobs using any framework.

B. Configure Kubeflow to run on Google Kubernetes Engine and submit training jobs through TFJob.

C. Create a library of VM images on Compute Engine, and publish these images on a centralized repository.

D. Set up Slurm workload manager to receive jobs that can be scheduled to run on your cloud infrastructure.

Answer: A

Explanation:

The best option for using a managed service to submit training jobs with different frameworks is to use Vertex AI Training. Vertex AI Training is a fully managed service that allows you to train custom models on Google Cloud using any framework, such as TensorFlow, PyTorch, scikit-learn, XGBoost, etc. You can also use custom containers to run your own libraries and dependencies. Vertex AI Training handles the infrastructure provisioning, scaling, and monitoring for you, so you can focus on your model development and optimization. Vertex AI Training also integrates with other Vertex AI services, such as Vertex AI Pipelines, Vertex AI Experiments, and Vertex AI Prediction. The other options are not as suitable for using a managed service to submit training jobs with different frameworks, because:

Configuring Kubeflow to run on Google Kubernetes Engine and submit training jobs through TFJob would require more infrastructure maintenance, as Kubeflow is not a fully managed service, and you would have to provision and manage your own Kubernetes cluster. This would also incur more costs, as you would have to pay for the cluster resources, regardless of the training job usage. TFJob is also mainly designed for TensorFlow models, and might not support other frameworks as well as Vertex AI Training.

Creating a library of VM images on Compute Engine, and publishing these images on a centralized repository would require more development time and effort, as you would have to create and maintain different VM images for different frameworks and libraries. You would also have to manually configure and launch the VMs for each training job, and

handle the scaling and monitoring yourself. This would not leverage the benefits of a managed service, such as Vertex AI Training. Setting up Slurm workload manager to receive jobs that can be scheduled to run on your cloud infrastructure would require more configuration and administration, as Slurm is not a native Google Cloud service, and you would have to install and manage it on your own VMs or clusters. Slurm is

also a general-purpose workload manager, and might not have the same level of integration and optimization for ML frameworks and libraries as Vertex AI Training. Reference:

Vertex AI Training | Google Cloud

Kubeflow on Google Cloud | Google Cloud

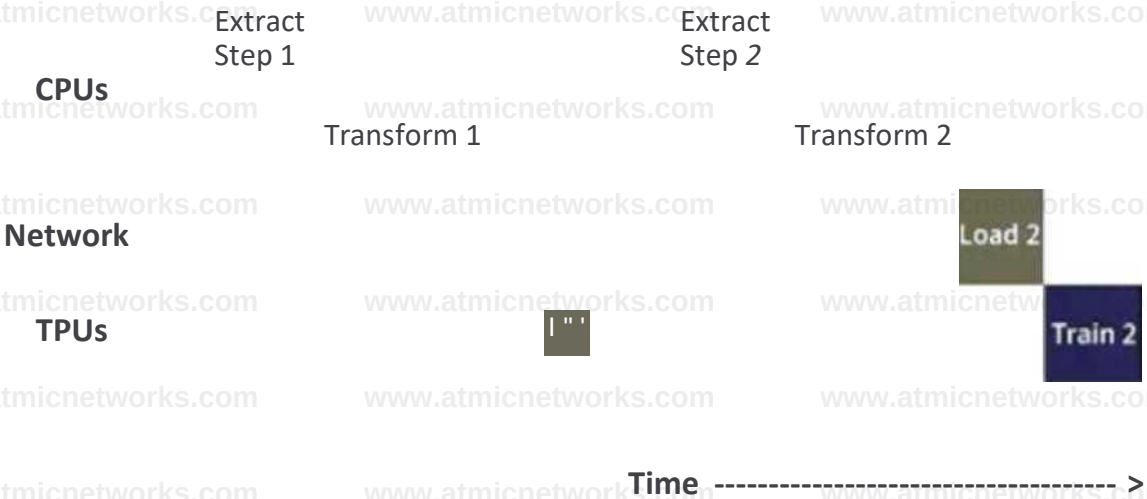
TFJob for training TensorFlow models with Kubernetes | Kubeflow

Compute Engine | Google Cloud

Slurm Workload Manager

Question: 123

You are training an object detection model using a Cloud TPU v2. Training time is taking longer than expected. Based on this simplified trace obtained with a Cloud TPU profile, what action should you take to decrease training time in a cost-efficient way?



A. Move from Cloud TPU v2 to Cloud TPU v3 and increase batch size.
B. Move from Cloud TPU v2 to 8 NVIDIA V100 GPUs and increase batch size.
C. Rewrite your input function to resize and reshape the input images.
D. Rewrite your input function using parallel reads, parallel processing, and prefetch.

Answer: D

Explanation:

The trace in the question shows that the training time is taking longer than expected. This is likely due to the input function not being optimized. To decrease training time in a cost-efficient way, the best option is to rewrite the input function using parallel reads, parallel processing, and prefetch. This will allow the model to process the data more efficiently and decrease training time. Reference: [Cloud TPU Performance Guide] [Data input pipeline performance guide]

Question: 124

While performing exploratory data analysis on a dataset, you find that an important categorical feature has 5% null values. You want to minimize the bias that could result from the missing values. How should you handle the missing values?

A. Remove the rows with missing values, and upsample your dataset by 5%.
B. Replace the missing values with the feature's mean.
C. Replace the missing values with a placeholder category indicating a missing value.
D. Move the rows with missing values to your validation dataset.

Answer: C

Explanation:
The best option for handling missing values in a categorical feature is to replace them with a placeholder category indicating a missing value. This is a type of imputation, which is a method of estimating the missing values based on the observed data. Imputing the missing values with a placeholder category preserves the information that the data is missing, and avoids introducing bias or distortion in the feature distribution. It also allows the machine learning model to learn from the missingness pattern, and potentially use it as a predictor for the target variable. The other options are not suitable for handling missing values in a categorical feature, because:
Removing the rows with missing values and upsampling the dataset by 5% would reduce the size of the dataset and potentially lose important information. It would also introduce sampling bias and overfitting, as the upsampling process would create duplicate or synthetic observations that do not reflect the true population.
Replacing the missing values with the feature's mean would not make sense for a categorical feature, as the mean is a numerical measure that does not capture the mode or frequency of the categories. It would also create a new category that does not exist in the original data, and might confuse the machine learning model.
Moving the rows with missing values to the validation dataset would compromise the validity and reliability of the model evaluation, as the validation dataset would not be representative of the test or production data. It would also reduce the amount of data available for training the model, and might introduce leakage or inconsistency between the training and validation datasets. Reference: Imputation of missing values
Effective Strategies to Handle Missing Values in Data Analysis
How to Handle Missing Values of Categorical Variables?
Google Cloud launches machine learning engineer certification
Google Professional Machine Learning Engineer Certification
Professional ML Engineer Exam Guide
Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate

Question: 125

You are an ML engineer on an agricultural research team working on a crop disease detection tool to detect leaf rust spots in images of crops to determine the presence of a disease. These spots, which can vary in shape and size, are correlated to the severity of the disease. You want to develop a solution that predicts the presence and severity of the disease with high accuracy. What should you do?
A. Create an object detection model that can localize the rust spots.
B. Develop an image segmentation ML model to locate the boundaries of the rust spots.
C. Develop a template matching algorithm using traditional computer vision libraries.
D. Develop an image classification ML model to predict the presence of the disease.

Answer: B

Explanation:
The best option for developing a solution that predicts the presence and severity of the disease with high accuracy is to develop an image segmentation ML model to locate the boundaries of the rust spots. Image segmentation is a technique that partitions an image into multiple regions, each corresponding to a different object or semantic category.

Image segmentation can be used to detect and localize the rust spots in the images of crops, and measure their shape and size. This information can then be used to determine the presence and severity of the disease, as the rust spots are correlated to the disease symptoms. Image segmentation can also handle the variability of the rust spots, as it does not rely on predefined templates or thresholds. Image segmentation can be implemented using deep learning models, such as U-Net, Mask R-CNN, or DeepLab, which can learn from large-scale datasets and achieve high accuracy and robustness. The other options are not as suitable for developing a solution that predicts the presence and severity of the disease with high accuracy, because:

Creating an object detection model that can localize the rust spots would only provide the bounding boxes of the rust spots, not their exact boundaries. This would result in less precise measurements of the shape and size of the rust spots, and might affect the accuracy of the disease prediction. Object detection models are also more complex and computationally expensive than image segmentation models, as they have to perform both classification and localization tasks.

Developing a template matching algorithm using traditional computer vision libraries would require manually designing and selecting the templates for the rust spots, which might not capture the diversity and variability of the rust spots. Template matching algorithms are also sensitive to noise, occlusion, rotation, and scale changes, and might fail to detect the rust spots in different scenarios. Template matching algorithms are also less accurate and robust than deep learning models, as they do not learn from data.

Developing an image classification ML model to predict the presence of the disease would only provide a binary or categorical output, not the location or severity of the disease. Image classification models are also less informative and interpretable than image segmentation models, as they do not provide any spatial information or visual explanation for the prediction. Image classification models might also suffer from class imbalance or mislabeling issues, as the presence of the disease might not be consistent or clear across the images. Reference:

Image Segmentation | Computer Vision | Google Developers

Crop diseases and pests detection based on deep learning: a review | Plant Methods | Full Text Using Deep Learning for Image-Based Plant Disease Detection

Computer Vision, IoT and Data Fusion for Crop Disease Detection Using … On Using Artificial Intelligence and the Internet of Things for Crop … Crop Disease Detection Using Machine Learning and Computer Vision

Question: 126

You have been asked to productionize a proof-of-concept ML model built using Keras. The model was trained in a Jupyter notebook on a data scientist's local machine. The notebook contains a cell that performs data validation and a cell that performs model analysis. You need to orchestrate the steps contained in the notebook and automate the execution of these steps for weekly retraining. You expect much more training data in the future. You want your solution to take advantage of managed services while minimizing cost. What should you do?

A. Move the Jupyter notebook to a Notebooks instance on the largest N2 machine type, and schedule the execution of the steps in the Notebooks instance using Cloud Scheduler.

B. Write the code as a TensorFlow Extended (TFX) pipeline orchestrated with Vertex AI Pipelines. Use standard TFX components for data validation and model analysis, and use Vertex AI Pipelines for model retraining.

C. Rewrite the steps in the Jupyter notebook as an Apache Spark job, and schedule the execution of the job on ephemeral Dataproc clusters using Cloud Scheduler.

D. Extract the steps contained in the Jupyter notebook as Python scripts, wrap each script in an Apache Airflow BashOperator, and run the resulting directed acyclic graph (DAG) in Cloud Composer.

Answer: B

Explanation:

The best option for productionizing a Keras model is to use TensorFlow Extended (TFX), a framework for building end-to-end machine learning pipelines that can handle large-scale data and complex workflows. TFX provides standard components for data ingestion, transformation, validation, analysis, training, tuning, serving, and monitoring. TFX pipelines can be orchestrated with Vertex AI Pipelines, a managed service that runs on Google Cloud Platform and leverages Kubernetes and Argo. Vertex AI Pipelines allows you to automate the execution of your TFX pipeline steps, schedule retraining jobs, and scale up or down the resources as needed. By using TFX and Vertex AI Pipelines, you can take advantage of the following benefits:

You can reuse the existing code in your Jupyter notebook, as TFX supports Keras as a first-class citizen. You can also use the Keras Tuner to optimize your model hyperparameters.

You can ensure data quality and consistency by using the TFX Data Validation component, which can detect anomalies, drift, and skew in your data. You can also use the TFX SchemaGen component to generate a schema for your data and enforce it throughout the pipeline.

You can analyze your model performance and fairness by using the TFX Model Analysis component, which can produce various metrics and visualizations. You can also use the TFX Model Validation component to compare your new model with a baseline model and set thresholds for deploying the model to production.

You can deploy your model to various serving platforms by using the TFX Pusher component, which can push your model to Vertex AI, Cloud AI Platform, TensorFlow Serving, or TensorFlow Lite. You can also use the TFX Model Registry to manage the versions and metadata of your models.

You can monitor your model performance and health by using the TFX Model Monitor component, which can detect data drift, concept drift, and prediction skew in your model. You can also use the TFX Evaluator component to compute metrics and validate your model against a baseline or a slice of data.

You can reduce the cost and complexity of managing your own infrastructure by using Vertex AI Pipelines, which provides a serverless environment for running your TFX pipeline. You can also use the Vertex AI Experiments and Vertex AI TensorBoard to track and visualize your pipeline runs. Reference:

[TensorFlow Extended (TFX)]

[Vertex AI Pipelines] [TFX User Guide]

Question: 127

You are working on a system log anomaly detection model for a cybersecurity organization. You have developed the model using TensorFlow, and you plan to use it for real-time prediction. You need to create a Dataflow pipeline to ingest data via Pub/Sub and write the results to BigQuery. You want to minimize the serving latency as much as possible. What should you do?

A. Containerize the model prediction logic in Cloud Run, which is invoked by Dataflow.
B. Load the model directly into the Dataflow job as a dependency, and use it for prediction.
C. Deploy the model to a Vertex AI endpoint, and invoke this endpoint in the Dataflow job.
D. Deploy the model in a TFServing container on Google Kubernetes Engine, and invoke it in the Dataflow job.

Answer: B

Explanation:

The best option for creating a Dataflow pipeline for real-time anomaly detection is to load the model directly into the Dataflow job as a dependency, and use it for prediction. This option has the following advantages:

It minimizes the serving latency, as the model prediction logic is executed within the same Dataflow pipeline that ingests and processes the data. There is no need to invoke external services or containers, which can introduce network overhead and latency.

It simplifies the deployment and management of the model, as the model is packaged with the Dataflow job and does not require a separate service or container. The model can be updated by redeploying the Dataflow job with a new model version.

It leverages the scalability and reliability of Dataflow, as the model prediction logic can scale up or down with the data volume and handle failures and retries automatically.

The other options are less optimal for the following reasons:

Option A: Containerizing the model prediction logic in Cloud Run, which is invoked by Dataflow, introduces additional latency and complexity. Cloud Run is a serverless platform that runs stateless containers, which means that the model prediction logic needs to be initialized and loaded every time a request is made. This can increase the cold start latency and reduce the throughput. Moreover, Cloud Run has a limit on the number of concurrent requests per container, which can affect the scalability of the model prediction logic. Additionally, this option requires managing two separate services: the Dataflow pipeline and the Cloud Run container.

Option C: Deploying the model to a Vertex AI endpoint, and invoking this endpoint in the Dataflow job, also introduces additional latency and complexity. Vertex AI is a managed service that provides various tools and features for machine learning, such as training, tuning, serving, and monitoring. However, invoking a Vertex AI endpoint from a Dataflow job requires making an HTTP request, which can incur network overhead and latency. Moreover, this option requires managing two separate services: the Dataflow pipeline and the Vertex AI endpoint.

Option D: Deploying the model in a TFServing container on Google Kubernetes Engine, and invoking it in the Dataflow job, also introduces additional latency and complexity. TFServing is a high- performance serving system for TensorFlow models, which can handle multiple versions and variants of a model. However, invoking a TFServing container from a Dataflow job requires making a gRPC or REST request, which can incur network overhead and latency. Moreover, this option requires managing two separate services: the Dataflow pipeline and the Google Kubernetes Engine cluster. Reference:

[Dataflow documentation]

[TensorFlow documentation]

[Cloud Run documentation]

[Vertex AI documentation]

[TFServing documentation]

Question: 128

You are an ML engineer at a mobile gaming company. A data scientist on your team recently trained a TensorFlow model, and you are responsible for deploying this model into a mobile application. You discover that the inference latency of the current model doesn't meet production requirements. You need to reduce the inference time by 50%, and you are willing to accept a small decrease in model accuracy in order to reach the latency requirement. Without training a new model, which model optimization technique for reducing latency should you try first?

A. Weight pruning
B. Dynamic range quantization
C. Model distillation
D. Dimensionality reduction

Answer: B

Explanation:

Dynamic range quantization is a model optimization technique for reducing latency that reduces the numerical precision of the weights and activations of models. This technique can reduce the model size, memory usage, and inference time by up to 4x with negligible accuracy loss. Dynamic range quantization can be applied to a trained

TensorFlow model without retraining, and it is suitable for mobile applications that require low latency and power consumption.

Weight pruning, model distillation, and dimensionality reduction are also model optimization techniques for reducing latency, but they have some limitations or drawbacks compared to dynamic range quantization:

Weight pruning works by removing parameters within a model that have only a minor impact on its predictions. Pruned models are the same size on disk, and have the same runtime latency, but can be compressed more effectively. This makes pruning a useful technique for reducing model download size, but not for reducing inference time.

Model distillation works by training a smaller and simpler model (student) to mimic the behavior of a larger and complex model (teacher). Distilled models can have lower latency and memory usage than the original models, but they require retraining and may not preserve the accuracy of the teacher model.

Dimensionality reduction works by reducing the number of features or dimensions in the input data or the model layers. Dimensionality reduction can improve the computational efficiency and generalization ability of models, but it may also lose some information or introduce noise in the data or the model. Dimensionality reduction also requires retraining or modifying the model architecture.

Reference:

[TensorFlow Model Optimization]

[TensorFlow Model Optimization Toolkit — Post-Training Integer Quantization]

[Model optimization methods to cut latency, adapt to new data]

Question: 129

You work on a data science team at a bank and are creating an ML model to predict loan default risk. You have collected and cleaned hundreds of millions of records worth of training data in a BigQuery table, and you now want to develop and compare multiple models on this data using TensorFlow and Vertex AI. You want to minimize any bottlenecks during the data ingestion state while considering scalability. What should you do?

A. Use the BigQuery client library to load data into a dataframe, and use tf.data.Dataset.from_tensor_slices() to read it.

B. Export data to CSV files in Cloud Storage, and use tf.data.TextLineDataset() to read them.

C. Convert the data into TFRecords, and use tf.data.TFRecordDataset() to read them.

D. Use TensorFlow I/O's BigQuery Reader to directly read the data.

Answer: D

Explanation:

The best option for developing and comparing multiple models on a large-scale BigQuery table using TensorFlow and Vertex AI is to use TensorFlow I/O's BigQuery Reader to directly read the data. This option has the following advantages:

It minimizes any bottlenecks during the data ingestion stage, as the BigQuery Reader can stream data from BigQuery to TensorFlow in parallel and in batches, without loading the entire table into memory or disk. The BigQuery Reader can also perform data transformations and filtering using SQL queries, reducing the need for additional preprocessing steps in TensorFlow.

It leverages the scalability and performance of BigQuery, as the BigQuery Reader can handle hundreds of millions of records worth of training data efficiently and reliably. BigQuery is a serverless, fully managed, and highly scalable data warehouse that can run complex queries over petabytes of data in seconds.

It simplifies the integration with Vertex AI, as the BigQuery Reader can be used with both custom and pre-built TensorFlow models on Vertex AI. Vertex AI is a unified platform for machine learning that provides various tools and features for data ingestion, data labeling, data preprocessing, model training, model tuning, model deployment, model

monitoring, and model explainability.

The other options are less optimal for the following reasons:

Option A: Using the BigQuery client library to load data into a dataframe, and using tf.data.Dataset.from_tensor_slices() to read it, introduces memory and performance issues. This option requires loading the entire BigQuery table into a Pandas dataframe, which can consume a lot of memory and cause out-of-memory errors. Moreover, using tf.data.Dataset.from_tensor_slices() to read the dataframe can be slow and inefficient, as it creates one slice per row of the dataframe, resulting in a large number of small tensors.

Option B: Exporting data to CSV files in Cloud Storage, and using tf.data.TextLineDataset() to read them, introduces additional steps and complexity. This option requires exporting the BigQuery table to one or more CSV files in Cloud Storage, which can take a long time and consume a lot of storage space. Moreover, using tf.data.TextLineDataset() to read the CSV files can be slow and error-prone, as it requires parsing and decoding each line of text, handling missing values and invalid data, and applying data transformations and validations.

Option C: Converting the data into TFRecords, and using tf.data.TFRecordDataset() to read them, introduces additional steps and complexity. This option requires converting the BigQuery table into one or more TFRecord files, which are binary files that store serialized TensorFlow examples. This can take a long time and consume a lot of storage space. Moreover, using tf.data.TFRecordDataset() to read the TFRecord files requires defining and parsing the schema of the TensorFlow examples, which can be tedious and error-prone.

Reference:

[TensorFlow I/O documentation]

[BigQuery documentation]

[Vertex AI documentation]

Question: 130

You have recently created a proof-of-concept (POC) deep learning model. You are satisfied with the overall architecture, but you need to determine the value for a couple of hyperparameters. You want to perform hyperparameter tuning on Vertex AI to determine both the appropriate embedding dimension for a categorical feature used by your model and the optimal learning rate. You configure the following settings:

For the embedding dimension, you set the type to INTEGER with a minValue of 16 and maxValue of 64.

For the learning rate, you set the type to DOUBLE with a minValue of 10e-05 and maxValue of 10e-02.

You are using the default Bayesian optimization tuning algorithm, and you want to maximize model accuracy. Training time is not a concern. How should you set the hyperparameter scaling for each hyperparameter and the maxParallelTrials?

A. Use UNIT_LINEAR_SCALE for the embedding dimension, UNIT_LOG_SCALE for the learning rate, and a large number of parallel trials.

B. Use UNIT_LINEAR_SCALE for the embedding dimension, UNIT_LOG_SCALE for the learning rate, and a small number of parallel trials.

C. Use UNIT_LOG_SCALE for the embedding dimension, UNIT_LINEAR_SCALE for the learning rate, and a large number of parallel trials.

D. Use UNIT_LOG_SCALE for the embedding dimension, UNIT_LINEAR_SCALE for the learning rate, and a small number of parallel trials.

Answer: A

Explanation:

The best option for performing hyperparameter tuning on Vertex AI to determine the appropriate embedding dimension and the optimal learning rate is to use UNIT_LINEAR_SCALE for the embedding dimension, UNIT_LOG_SCALE for the learning rate, and a large number of parallel trials. This option has the following advantages:

It matches the appropriate scaling type for each hyperparameter, based on their range and distribution. The embedding dimension is an integer hyperparameter that varies linearly between 16 and 64, so using UNIT_LINEAR_SCALE makes sense. The learning rate is a double hyperparameter that varies exponentially between 10e-05 and 10e-02, so using UNIT_LOG_SCALE is more suitable. It maximizes the exploration of the hyperparameter space, by using a large number of parallel trials. Since training time is not a concern, using more trials can help find the best combination of hyperparameters that maximizes model accuracy. The default Bayesian optimization tuning algorithm can efficiently sample the hyperparameter space and converge to the optimal values. The other options are less optimal for the following reasons:

Option B: Using UNIT_LINEAR_SCALE for the embedding dimension, UNIT_LOG_SCALE for the learning rate, and a small number of parallel trials, reduces the exploration of the hyperparameter space, by using a small number of parallel trials. Since training time is not a concern, using fewer trials can miss some potentially good combinations of hyperparameters that maximize model accuracy. The default Bayesian optimization tuning algorithm can benefit from more trials to sample the hyperparameter space and converge to the optimal values.

Option C: Using UNIT_LOG_SCALE for the embedding dimension, UNIT_LINEAR_SCALE for the learning rate, and a large number of parallel trials, mismatches the appropriate scaling type for each hyperparameter, based on their range and distribution. The embedding dimension is an integer hyperparameter that varies linearly between 16 and 64, so using UNIT_LOG_SCALE is not suitable. The learning rate is a double hyperparameter that varies exponentially between 10e-05 and 10e-02, so using UNIT_LINEAR_SCALE makes less sense.

Option D: Using UNIT_LOG_SCALE for the embedding dimension, UNIT_LINEAR_SCALE for the learning rate, and a small number of parallel trials, combines the drawbacks of option B and option C. It mismatches the appropriate scaling type for each hyperparameter, based on their range and distribution, and reduces the exploration of the hyperparameter space, by using a small number of parallel trials.

Reference:

[Vertex AI: Hyperparameter tuning overview]

[Vertex AI: Configuring the hyperparameter tuning job]

Question: 131

You are the Director of Data Science at a large company, and your Data Science team has recently begun using the Kubeflow Pipelines SDK to orchestrate their training pipelines. Your team is struggling to integrate their custom Python code into the Kubeflow Pipelines SDK. How should you instruct them to proceed in order to quickly integrate their code with the Kubeflow Pipelines SDK?

A. Use the func_to_container_op function to create custom components from the Python code.

B. Use the predefined components available in the Kubeflow Pipelines SDK to access Dataproc, and run the custom code there.

C. Package the custom Python code into Docker containers, and use the load_component_from_file function to import the containers into the pipeline.

D. Deploy the custom Python code to Cloud Functions, and use Kubeflow Pipelines to trigger the Cloud Function.

Answer: A

Explanation:

The easiest way to integrate custom Python code into the Kubeflow Pipelines SDK is to use the func_to_container_op function, which converts a Python function into a pipeline component. This function automatically builds a Docker image that executes the Python function, and returns a factory function that can be used to create kfp.dsl.ContainerOp instances for the pipeline. This option has the following benefits:

It allows the data science team to reuse their existing Python code without rewriting it or packaging it into containers manually.

It simplifies the component specification and implementation, as the function signature defines the component interface and the function body defines the component logic.

It supports various types of inputs and outputs, such as primitive types, files, directories, and dictionaries.

The other options are less optimal for the following reasons:

Option B: Using the predefined components available in the Kubeflow Pipelines SDK to access Dataproc, and run the custom code there, introduces additional complexity and cost. This option requires creating and managing Dataproc clusters, which are ephemeral and scalable clusters of Compute Engine instances that run Apache Spark and Apache Hadoop. Moreover, this option requires writing the custom code in PySpark or Hadoop MapReduce, which may not be compatible with the existing Python code.

Option C: Packaging the custom Python code into Docker containers, and using the load_component_from_file function to import the containers into the pipeline, introduces additional steps and overhead. This option requires creating and maintaining Dockerfiles, building and pushing Docker images, and writing component specifications in YAML files. Moreover, this option requires managing the dependencies and versions of the Python code and the Docker images.

Option D: Deploying the custom Python code to Cloud Functions, and using Kubeflow Pipelines to trigger the Cloud Function, introduces additional latency and limitations. This option requires creating and deploying Cloud Functions, which are serverless functions that execute in response to events. Moreover, this option requires invoking the Cloud Functions from the Kubeflow Pipelines using HTTP requests, which can incur network overhead and latency. Additionally, this option is subject to the quotas and limits of Cloud Functions, such as the maximum execution time and memory usage.

Reference:

Building Python function-based components | Kubeflow

Building Python Function-based Components | Kubeflow

Question: 132

You work for the AI team of an automobile company, and you are developing a visual defect detection model using TensorFlow and Keras. To improve your model performance, you want to incorporate some image augmentation functions such as translation, cropping, and contrast tweaking. You randomly apply these functions to each training batch. You want to optimize your data processing pipeline for run time and compute resources utilization. What should you do?

A. Embed the augmentation functions dynamically in the tf.Data pipeline.

B. Embed the augmentation functions dynamically as part of Keras generators.

C. Use Dataflow to create all possible augmentations, and store them as TFRecords.

D. Use Dataflow to create the augmentations dynamically per training run, and stage them as TFRecords.

Answer: A

Explanation:

The best option for optimizing the data processing pipeline for run time and compute resources utilization is to embed the augmentation functions dynamically in the tf.Data pipeline. This option has the following advantages:

It allows the data augmentation to be performed on the fly, without creating or storing additional copies of the data. This saves storage space and reduces the data transfer time.

It leverages the parallelism and performance of the tf.Data API, which can efficiently apply the augmentation functions to multiple batches of data in parallel, using multiple CPU cores or GPU devices. The tf.Data API also supports various optimization techniques, such as caching, prefetching, and autotuning, to improve the data processing speed and reduce the latency.

It integrates seamlessly with the TensorFlow and Keras models, which can consume the tf.Data datasets as inputs for training and evaluation. The tf.Data API also supports various data formats, such as images, text, audio, and video, and various data sources, such as files, databases, and web services.

The other options are less optimal for the following reasons:

Option B: Embedding the augmentation functions dynamically as part of Keras generators introduces some limitations and overhead. Keras generators are Python generators that yield batches of data for training or evaluation. However, Keras generators are not compatible with the tf.distribute API, which is used to distribute the training across multiple devices or machines. Moreover, Keras generators are not as efficient or scalable as the tf.Data API, as they run on a single Python thread and do not support parallelism or optimization techniques.

Option C: Using Dataflow to create all possible augmentations, and store them as TFRecords introduces additional complexity and cost. Dataflow is a fully managed service that runs Apache Beam pipelines for data processing and transformation. However, using Dataflow to create all possible augmentations requires generating and storing a large number of augmented images, which can consume a lot of storage space and incur storage and network costs. Moreover, using Dataflow to create the augmentations requires writing and deploying a separate Dataflow pipeline, which can be tedious and time-consuming.

Option D: Using Dataflow to create the augmentations dynamically per training run, and stage them as TFRecords introduces additional complexity and latency. Dataflow is a fully managed service that runs Apache Beam pipelines for data processing and transformation. However, using Dataflow to create the augmentations dynamically per training run requires running a Dataflow pipeline every time the model is trained, which can introduce latency and delay the training process. Moreover, using Dataflow to create the augmentations requires writing and deploying a separate Dataflow pipeline, which can be tedious and time-consuming.

Reference:

[tf.data: Build TensorFlow input pipelines]

[Image augmentation | TensorFlow Core] [Dataflow documentation]

Question: 133

You work for an online publisher that delivers news articles to over 50 million readers. You have built an AI model that recommends content for the company's weekly newsletter. A recommendation is considered successful if the article is opened within two days of the newsletter's published date and the user remains on the page for at least one minute.

All the information needed to compute the success metric is available in BigQuery and is updated hourly. The model is trained on eight weeks of data, on average its performance degrades below the acceptable baseline after five weeks, and training time is 12 hours. You want to ensure that the model's performance is above the acceptable baseline while minimizing cost. How should you monitor the model to determine when retraining is necessary?

A. Use Vertex AI Model Monitoring to detect skew of the input features with a sample rate of 100% and a monitoring frequency of two days.
B. Schedule a cron job in Cloud Tasks to retrain the model every week before the newsletter is created.
C. Schedule a weekly query in BigQuery to compute the success metric.

D. Schedule a daily Dataflow job in Cloud Composer to compute the success metric.

Answer: C

Explanation:

The best option for monitoring the model to determine when retraining is necessary is to schedule a weekly query in BigQuery to compute the success metric. This option has the following advantages: It allows the model performance to be evaluated regularly, based on the actual outcome of the recommendations. By computing the success metric, which is the percentage of articles that are opened within two days and read for at least one minute, you can measure how well the model is achieving its objective and compare it with the acceptable baseline.

It leverages the scalability and efficiency of BigQuery, which is a serverless, fully managed, and highly scalable data warehouse that can run complex queries over petabytes of data in seconds. By using BigQuery, you can access and analyze all the information needed to compute the success metric, such as the newsletter publication date, the article opening date, and the user reading time, without worrying about the infrastructure or the cost.

It simplifies the model monitoring and retraining workflow, as the weekly query can be scheduled and executed automatically using BigQuery's built-in scheduling feature. You can also set up alerts or notifications to inform you when the success metric falls below the acceptable baseline, and trigger the model retraining process accordingly.

The other options are less optimal for the following reasons:

Option A: Using Vertex AI Model Monitoring to detect skew of the input features with a sample rate of 100% and a monitoring frequency of two days introduces additional complexity and overhead. This option requires setting up and managing a Vertex AI Model Monitoring service, which is a managed service that provides various tools and features for machine learning, such as training, tuning, serving, and monitoring. However, using Vertex AI Model Monitoring to detect skew of the input features may not reflect the actual performance of the model, as skew is the discrepancy between the distributions of the features in the training dataset and the serving data, which may not affect the outcome of the recommendations. Moreover, using a sample rate of 100% and a monitoring frequency of two days may incur unnecessary cost and latency, as it requires analyzing all the input features every two days, which may not be needed for the model monitoring.

Option B: Scheduling a cron job in Cloud Tasks to retrain the model every week before the newsletter is created introduces additional cost and risk. This option requires creating and running a cron job in Cloud Tasks, which is a fully managed service that allows you to schedule and execute tasks that are invoked by HTTP requests. However, using Cloud Tasks to retrain the model every week may not be optimal, as it may retrain the model more often than necessary, wasting compute resources and cost.

Moreover, using Cloud Tasks to retrain the model before the newsletter is created may introduce risk, as it may deploy a new model version that has not been tested or validated, potentially affecting the quality of the recommendations.

Option D: Scheduling a daily Dataflow job in Cloud Composer to compute the success metric introduces additional complexity and cost. This option requires creating and running a Dataflow job in Cloud Composer, which is a fully managed service that runs Apache Airflow pipelines for workflow orchestration. Dataflow is a fully managed service that runs Apache Beam pipelines for data processing and transformation. However, using Dataflow and Cloud Composer to compute the success metric may not be necessary, as it may add more steps and overhead to the model monitoring process. Moreover, using Dataflow and Cloud Composer to compute the success metric daily may not be optimal, as it may compute the success metric more often than needed, consuming more compute resources and cost.

Reference:

[BigQuery documentation]

[Vertex AI Model Monitoring documentation]

[Cloud Tasks documentation]

[Cloud Composer documentation]

[Dataflow documentation]

www.atmicnetworks.com

Question: 134

You deployed an ML model into production a year ago. Every month, you collect all raw requests that were sent to your model prediction service during the previous month. You send a subset of these requests to a human labeling service to evaluate your model's performance. After a year, you notice that your model's performance sometimes degrades significantly after a month, while other times it takes several months to notice any decrease in performance. The labeling service is costly, but you also need to avoid large performance degradations. You want to determine how often you should retrain your model to maintain a high level of performance while minimizing cost. What should you do?

A. Train an anomaly detection model on the training dataset, and run all incoming requests through this model. If an anomaly is detected, send the most recent serving data to the labeling service.
B. Identify temporal patterns in your model's performance over the previous year. Based on these patterns, create a schedule for sending serving data to the labeling service for the next year.
C. Compare the cost of the labeling service with the lost revenue due to model performance degradation over the past year. If the lost revenue is greater than the cost of the labeling service, increase the frequency of model retraining; otherwise, decrease the model retraining frequency. D. Run training-serving skew detection batch jobs every few days to compare the aggregate statistics of the features in the training dataset with recent serving data. If skew is detected, send the most recent serving data to the labeling service.

Answer: D

Explanation:
The best option for determining how often to retrain your model to maintain a high level of performance while minimizing cost is to run training-serving skew detection batch jobs every few days. Training-serving skew refers to the discrepancy between the distributions of the features in the training dataset and the serving data. This can cause the model to perform poorly on the new data, as it is not representative of the data that the model was trained on. By running training-serving skew detection batch jobs, you can monitor the changes in the feature distributions over time, and identify when the skew becomes significant enough to affect the model performance. If skew is detected, you can send the most recent serving data to the labeling service, and use the labeled data to retrain your model. This option has the following benefits:
It allows you to retrain your model only when necessary, based on the actual data changes, rather than on a fixed schedule or a heuristic. This can save you the cost of the labeling service and the retraining process, and also avoid overfitting or underfitting your model.
It leverages the existing tools and frameworks for training-serving skew detection, such as TensorFlow Data Validation (TFDV) and Vertex Data Labeling. TFDV is a library that can compute and visualize descriptive statistics for your datasets, and compare the statistics across different datasets. Vertex Data Labeling is a service that can label your data with high quality and low latency, using either human labelers or automated labelers.
It integrates well with the MLOps practices, such as continuous integration and continuous delivery (CI/CD), which can automate the workflow of running the skew detection jobs, sending the data to the labeling service, retraining the model, and deploying the new model version.
The other options are less optimal for the following reasons:
Option A: Training an anomaly detection model on the training dataset, and running all incoming requests through this model, introduces additional complexity and overhead. This option requires building and maintaining a separate model for anomaly detection, which can be challenging and time-consuming. Moreover, this option requires running the anomaly detection model on every request, which can increase the latency and resource consumption of the prediction service. Additionally, this option may not capture the subtle changes in the feature distributions that can affect the model performance, as anomalies are usually defined as rare or extreme events.
Option B: Identifying temporal patterns in your model's performance over the previous year, and creating a schedule for sending serving data to the labeling service for the next year, introduces additional assumptions and risks. This

option requires analyzing the historical data and model performance, and finding the patterns that can explain the variations in the model performance over time. However, this can be difficult and unreliable, as the patterns may not be consistent or predictable, and may depend on various factors that are not captured by the data. Moreover, this option requires creating a schedule based on the past patterns, which may not reflect the future changes in the data or the environment. This can lead to either sending too much or too little data to the labeling service, resulting in either wasted cost or degraded performance.

Option C: Comparing the cost of the labeling service with the lost revenue due to model performance degradation over the past year, and adjusting the frequency of model retraining accordingly, introduces additional challenges and trade-offs. This option requires estimating the cost of the labeling service and the lost revenue due to model performance degradation, which can be difficult and inaccurate, as they may depend on various factors that are not easily quantifiable or measurable. Moreover, this option requires finding the optimal balance between the cost and the performance, which can be subjective and variable, as different stakeholders may have different preferences and expectations. Furthermore, this option may not account for the potential impact of the model performance degradation on other aspects of the business, such as customer satisfaction, retention, or loyalty.

Question: 135

You work for a company that manages a ticketing platform for a large chain of cinemas. Customers use a mobile app to search for movies they're interested in and purchase tickets in the app. Ticket purchase requests are sent to Pub/Sub and are processed with a Dataflow streaming pipeline configured to conduct the following steps:

1. Check for availability of the movie tickets at the selected cinema.
2. Assign the ticket price and accept payment.
3. Reserve the tickets at the selected cinema.
4. Send successful purchases to your database.

Each step in this process has low latency requirements (less than 50 milliseconds). You have developed a logistic regression model with BigQuery ML that predicts whether offering a promo code for free popcorn increases the chance of a ticket purchase, and this prediction should be added to the ticket purchase process. You want to identify the simplest way to deploy this model to production while adding minimal latency. What should you do?

A. Run batch inference with BigQuery ML every five minutes on each new set of tickets issued.

B. Export your model in TensorFlow format, and add a tfx_bsl.public.beam.RunInference step to the Dataflow pipeline.

C. Export your model in TensorFlow format, deploy it on Vertex AI, and query the prediction endpoint from your streaming pipeline.

D. Convert your model with TensorFlow Lite (TFLite), and add it to the mobile app so that the promo code and the incoming request arrive together in Pub/Sub.

Answer: B

Explanation:

The simplest way to deploy a logistic regression model with BigQuery ML to production while adding minimal latency is to export the model in TensorFlow format, and add a tfx_bsl.public.beam.RunInference step to the Dataflow pipeline. This option has the following advantages:

It allows the model prediction to be performed in real time, as part of the Dataflow streaming pipeline that processes the ticket purchase requests. This ensures that the promo code offer is based on the most recent data and customer behavior, and that the offer is delivered to the customer without delay.

It leverages the compatibility and performance of TensorFlow and Dataflow, which are both part of the Google Cloud ecosystem. TensorFlow is a popular and powerful framework for building and deploying machine learning models, and Dataflow is a fully managed service that runs Apache Beam pipelines for data processing and transformation. By using

the tfx_bsl.public.beam.RunInference step, you can easily integrate your TensorFlow model with your Dataflow pipeline, and take advantage of the parallelism and scalability of Dataflow.

It simplifies the model deployment and management, as the model is packaged with the Dataflow pipeline and does not require a separate service or endpoint. The model can be updated by redeploying the Dataflow pipeline with a new model version.

The other options are less optimal for the following reasons:

Option A: Running batch inference with BigQuery ML every five minutes on each new set of tickets issued introduces additional latency and complexity. This option requires running a separate BigQuery job every five minutes, which can incur network overhead and latency. Moreover, this option requires storing and retrieving the intermediate results of the batch inference, which can consume storage space and increase the data transfer time.

Option C: Exporting the model in TensorFlow format, deploying it on Vertex AI, and querying the prediction endpoint from the streaming pipeline introduces additional latency and cost. This option requires creating and managing a Vertex AI endpoint, which is a managed service that provides various tools and features for machine learning, such as training, tuning, serving, and monitoring. However, querying the Vertex AI endpoint from the streaming pipeline requires making an HTTP request, which can incur network overhead and latency. Moreover, this option requires paying for the Vertex AI endpoint usage, which can increase the cost of the model deployment.

Option D: Converting the model with TensorFlow Lite (TFLite), and adding it to the mobile app so that the promo code and the incoming request arrive together in Pub/Sub introduces additional challenges and risks. This option requires converting the model to a TFLite format, which is a lightweight and optimized format for running TensorFlow models on mobile and embedded devices. However, converting the model to TFLite may not preserve the accuracy or functionality of the original model, as some operations or features may not be supported by TFLite. Moreover, this option requires updating the mobile app with the TFLite model, which can be tedious and timeconsuming, and may depend on the user's willingness to update the app. Additionally, this option may expose the model to potential security or privacy issues, as the model is running on the user's device and may be accessed or modified by malicious actors.

Reference:

[Exporting models for prediction | BigQuery ML]

[tfx_bsl.public.beam.run_inference | TensorFlow Extended]

[Vertex AI documentation] [TensorFlow Lite documentation]

Question: 136

You work for a retailer that sells clothes to customers around the world. You have been tasked with ensuring that ML models are built in a secure manner. Specifically, you need to protect sensitive customer data that might be used in the models. You have identified four fields containing sensitive data that are being used by your data science team: AGE, IS_EXISTING_CUSTOMER, LATITUDE_LONGITUDE, and SHIRT_SIZE. What should you do with the data before it is made available to the data science team for training purposes?

A. Tokenize all of the fields using hashed dummy values to replace the real values.
B. Use principal component analysis (PCA) to reduce the four sensitive fields to one PCA vector.
C. Coarsen the data by putting AGE into quantiles and rounding LATITUDE_LONGTTUDE into single precision. The other two fields are already as coarse as possible.
D. Remove all sensitive data fields, and ask the data science team to build their models using nonsensitive data.

Answer: C

Explanation:

The best option for protecting sensitive customer data that might be used in the ML models is to coarsen the data by

putting AGE into quantiles and rounding LATITUDE_LONGITUDE into single precision. This option has the following advantages:

It preserves the utility and relevance of the data for the ML models, as the coarsened data still captures the essential information and patterns that the models need to learn. For example, putting AGE into quantiles can group the customers into different age ranges, which can be useful for predicting their preferences or behavior. Rounding LATITUDE_LONGITUDE into single precision can reduce the precision of the location data, but still retain the general geographic region of the customers, which can be useful for personalizing the recommendations or offers.

It reduces the risk of exposing the personal or private information of the customers, as the coarsened data makes it harder to identify or re-identify the individual customers from the data. For example, putting AGE into quantiles can hide the exact age of the customers, which can be considered sensitive or confidential. Rounding LATITUDE_LONGITUDE into single precision can obscure the exact location of the customers, which can be considered sensitive or confidential.

The other options are less optimal for the following reasons:

Option A: Tokenizing all of the fields using hashed dummy values to replace the real values eliminates the utility and relevance of the data for the ML models, as the tokenized data loses all the information and patterns that the models need to learn. For example, tokenizing AGE using hashed dummy values can make the data meaningless and irrelevant, as the models cannot learn anything from the random tokens. Tokenizing LATITUDE_LONGITUDE using hashed dummy values can make the data meaningless and irrelevant, as the models cannot learn anything from the random tokens.

Option B: Using principal component analysis (PCA) to reduce the four sensitive fields to one PCA vector reduces the utility and relevance of the data for the ML models, as the PCA vector may not capture all the information and patterns that the models need to learn. For example, using PCA to reduce AGE, IS_EXISTING_CUSTOMER, LATITUDE_LONGITUDE, and SHIRT_SIZE to one PCA vector can lose some information or introduce noise in the data, as the PCA vector is a linear combination of the original features, which may not reflect their true relationship or importance. Moreover, using PCA to reduce the four sensitive fields to one PCA vector may not reduce the risk of exposing the personal or private information of the customers, as the PCA vector may still be reversible or linkable to the original data, depending on the amount of variance explained by the PCA vector and the availability of the PCA transformation matrix.

Option D: Removing all sensitive data fields, and asking the data science team to build their models using non-sensitive data reduces the utility and relevance of the data for the ML models, as the nonsensitive data may not contain enough information and patterns that the models need to learn. For example, removing AGE, IS_EXISTING_CUSTOMER, LATITUDE_LONGITUDE, and SHIRT_SIZE from the data can make the data insufficient and unrepresentative, as the models may not be able to learn the factors that influence the customers' preferences or behavior. Moreover, removing all sensitive data fields from the data may not be necessary or feasible, as the data protection legislation may allow the use of sensitive data for the ML models, as long as the data is processed in a secure and ethical manner, and the customers' consent and rights are respected.

Reference:

[Protecting Sensitive Data and AI Models with Confidential Computing | NVIDIA Technical Blog Training machine learning models from sensitive data | Fast Data Science]
[Securing ML applications. Model security and protection - Medium]
[Security of AI/ML systems, ML model security | Cossack Labs]
[Vulnerabilities, security and privacy for machine learning models]

Question: 137

You work for a magazine publisher and have been tasked with predicting whether customers will cancel their annual subscription. In your exploratory data analysis, you find that 90% of individuals renew their subscription every year, and only 10% of individuals cancel their subscription. After training a NN Classifier, your model predicts those who cancel their subscription with 99% accuracy and predicts those who renew their subscription with 82% accuracy. How should you interpret these results?

A. This is not a good result because the model should have a higher accuracy for those who renew their subscription than for those who cancel their subscription.

B. This is not a good result because the model is performing worse than predicting that people will always renew their subscription.

C. This is a good result because predicting those who cancel their subscription is more difficult, since there is less data for this group.

D. This is a good result because the accuracy across both groups is greater than 80%.

Answer: B

Explanation:

This is not a good result because the model is performing worse than predicting that people will always renew their subscription. This option has the following reasons:

It indicates that the model is not learning from the data, but rather memorizing the majority class. Since 90% of the individuals renew their subscription every year, the model can achieve a 90% accuracy by simply predicting that everyone will renew their subscription, without considering the features or the patterns in the data. However, the model's accuracy for predicting those who renew their subscription is only 82%, which is lower than the baseline accuracy of 90%. This suggests that the model is overfitting to the minority class (those who cancel their subscription), and underfitting to the majority class (those who renew their subscription).

It implies that the model is not useful for the business problem, as it cannot identify the customers who are at risk of churning. The goal of predicting whether customers will cancel their annual subscription is to prevent customer churn and increase customer retention. However, the model's accuracy for predicting those who cancel their subscription is 99%, which is too high and unrealistic, as it means that the model can almost perfectly identify the customers who will churn, without any false positives or false negatives. This may indicate that the model is cheating or exploiting some leakage in the data, such as a feature that reveals the outcome of the prediction. Moreover, the model's accuracy for predicting those who renew their subscription is 82%, which is too low and unreliable, as it means that the model can miss many customers who will churn, and falsely label them as renewing customers. This can lead to losing customers and revenue, and failing to take proactive actions to retain them.

Reference:

How to Evaluate Machine Learning Models: Classification Metrics | Machine Learning Mastery Imbalanced Classification: Predicting Subscription Churn | Machine Learning Mastery

Question: 138

You have built a model that is trained on data stored in Parquet files. You access the data through a Hive table hosted on Google Cloud. You preprocessed these data with PySpark and exported it as a CSV file into Cloud Storage. After preprocessing, you execute additional steps to train and evaluate your model. You want to parametrize this model training in Kubeflow Pipelines. What should you do?

A. Remove the data transformation step from your pipeline.

B. Containerize the PySpark transformation step, and add it to your pipeline.

C. Add a ContainerOp to your pipeline that spins a Dataproc cluster, runs a transformation, and then saves the transformed data in Cloud Storage.

D. Deploy Apache Spark at a separate node pool in a Google Kubernetes Engine cluster. Add a ContainerOp to your pipeline that invokes a corresponding transformation job for this Spark instance.

Answer: C

Explanation:

The best option for parametrizing the model training in Kubeflow Pipelines is to add a ContainerOp to the pipeline that spins a Dataproc cluster, runs a transformation, and then saves the transformed data in Cloud Storage. This option has the following advantages:

It allows the data transformation to be performed as part of the Kubeflow Pipeline, which can ensure the consistency and reproducibility of the data processing and the model training. By adding a ContainerOp to the pipeline, you can define the parameters and the logic of the data transformation step, and integrate it with the other steps of the pipeline, such as the model training and evaluation. It leverages the scalability and performance of Dataproc, which is a fully managed service that runs Apache Spark and Apache Hadoop clusters on Google Cloud. By spinning a Dataproc cluster, you can run the PySpark transformation on the Parquet files stored in the Hive table, and take advantage of the parallelism and speed of Spark. Dataproc also supports various features and integrations, such as autoscaling, preemptible VMs, and connectors to other Google Cloud services, that can optimize the data processing and reduce the cost.

It simplifies the data storage and access, as the transformed data is saved in Cloud Storage, which is a scalable, durable, and secure object storage service. By saving the transformed data in Cloud Storage, you can avoid the overhead and complexity of managing the data in the Hive table or the Parquet files. Moreover, you can easily access the transformed data from Cloud Storage, using various tools and frameworks, such as TensorFlow, BigQuery, or Vertex AI.

The other options are less optimal for the following reasons:

Option A: Removing the data transformation step from the pipeline eliminates the parametrization of the model training, as the data processing and the model training are decoupled and independent. This option requires running the PySpark transformation separately from the Kubeflow Pipeline, which can introduce inconsistency and unreproducibility in the data processing and the model training. Moreover, this option requires managing the data in the Hive table or the Parquet files, which can be cumbersome and inefficient.

Option B: Containerizing the PySpark transformation step, and adding it to the pipeline introduces additional complexity and overhead. This option requires creating and maintaining a Docker image that can run the PySpark transformation, which can be challenging and time-consuming. Moreover, this option requires running the PySpark transformation on a single container, which can be slow and inefficient, as it does not leverage the parallelism and performance of Spark.

Option D: Deploying Apache Spark at a separate node pool in a Google Kubernetes Engine cluster, and adding a ContainerOp to the pipeline that invokes a corresponding transformation job for this Spark instance introduces additional complexity and cost. This option requires creating and managing a separate node pool in a Google Kubernetes Engine cluster, which is a fully managed service that runs Kubernetes clusters on Google Cloud. Moreover, this option requires deploying and running Apache Spark on the node pool, which can be tedious and costly, as it requires configuring and maintaining the Spark cluster, and paying for the node pool usage.

Question: 139

You are developing an ML model using a dataset with categorical input variables. You have randomly split half of the data into training and test sets. After applying one-hot encoding on the categorical variables in the training set, you discover that one categorical variable is missing from the test set. What should you do?

A. Randomly redistribute the data, with 70% for the training set and 30% for the test set
B. Use sparse representation in the test set
C. Apply one-hot encoding on the categorical variables in the test data.
D. Collect more data representing all categories

Answer: C

Explanation:

The best option for dealing with the missing categorical variable in the test set is to apply one-hot encoding on the categorical variables in the test data. This option has the following advantages: It ensures the consistency and compatibility of the data format for the ML model, as the one-hot encoding transforms the categorical variables into binary vectors that can be easily processed by the model. By applying one-hot encoding on the categorical variables in the test data, you can match the number and order of the features in the test data with the training data, and avoid any errors or discrepancies in the model prediction.

It preserves the information and relevance of the data for the ML model, as the one-hot encoding creates a separate feature for each possible value of the categorical variable, and assigns a value of 1 to the feature corresponding to the actual value of the variable, and 0 to the rest. By applying one- hot encoding on the categorical variables in the test data, you can retain the original meaning and importance of the categorical variable, and avoid any loss or distortion of the data.

The other options are less optimal for the following reasons:

Option A: Randomly redistributing the data, with 70% for the training set and 30% for the test set, introduces additional complexity and risk. This option requires reshuffling and splitting the data again, which can be tedious and time-consuming. Moreover, this option may not guarantee that the missing categorical variable will be present in the test set, as it depends on the randomness of the data distribution. Furthermore, this option may affect the quality and validity of the ML model, as it may change the data characteristics and patterns that the model has learned from the original training set.

Option B: Using sparse representation in the test set introduces additional overhead and inefficiency. This option requires converting the categorical variables in the test set into sparse vectors, which are vectors that have mostly zero values and only store the indices and values of the non-zero elements. However, using sparse representation in the test set may not be compatible with the ML model, as the model expects the input data to have the same format and dimensionality as the training data, which uses one-hot encoding. Moreover, using sparse representation in the test set may not be efficient or scalable, as it requires additional computation and memory to store and process the sparse vectors.

Option D: Collecting more data representing all categories introduces additional cost and delay. This option requires obtaining and labeling more data that contains the missing categorical variable, which can be expensive and time-consuming. Moreover, this option may not be feasible or

necessary, as the missing categorical variable may not be available or relevant for the test data, depending on the data source or the business problem.

Question: 140

You are developing an image recognition model using PyTorch based on ResNet50 architecture. Your code is working fine on your local laptop on a small subsample. Your full dataset has 200k labeled images You want to quickly scale your training workload while minimizing cost. You plan to use 4 V100 GPUs. What should you do? (Choose Correct Answer and Give Reference and Explanation)

A. Configure a Compute Engine VM with all the dependencies that launches the training Train your model with Vertex Al using a custom tier that contains the required GPUs.

B. Package your code with Setuptools. and use a pre-built container Train your model with Vertex Al using a custom tier that contains the required GPUs.

C. Create a Vertex Al Workbench user-managed notebooks instance with 4 V100 GPUs, and use it to train your model

D. Create a Google Kubernetes Engine cluster with a node pool that has 4 V100 GPUs Prepare and submit a TFJob operator to this node pool.

Answer: B

Explanation:

The best option for scaling the training workload while minimizing cost is to package the code with Setuptools, and use a pre-built container. Train the model with Vertex AI using a custom tier that contains the required GPUs. This option has the following advantages:

It allows the code to be easily packaged and deployed, as Setuptools is a Python tool that helps to create and distribute Python packages, and pre-built containers are Docker images that contain all the dependencies and libraries needed to run the code. By packaging the code with Setuptools, and using a pre-built container, you can avoid the hassle and complexity of building and maintaining your own custom container, and ensure the compatibility and portability of your code across different environments.

It leverages the scalability and performance of Vertex AI, which is a fully managed service that provides various tools and features for machine learning, such as training, tuning, serving, and monitoring. By training the model with Vertex AI, you can take advantage of the distributed and parallel training capabilities of Vertex AI, which can speed up the training process and improve the model quality. Vertex AI also supports various frameworks and models, such as PyTorch and ResNet50, and allows you to use custom containers and custom tiers to customize your training configuration and resources.

It reduces the cost and complexity of the training process, as Vertex AI allows you to use a custom tier that contains the required GPUs, which can optimize the resource utilization and allocation for your training job. By using a custom tier that contains 4 V100 GPUs, you can match the number and type of GPUs that you plan to use for your training job, and avoid paying for unnecessary or underutilized resources. Vertex AI also offers various pricing options and discounts, such as per-

second billing, sustained use discounts, and preemptible VMs, that can lower the cost of the training process.

The other options are less optimal for the following reasons:

Option A: Configuring a Compute Engine VM with all the dependencies that launches the training. Train the model with Vertex AI using a custom tier that contains the required GPUs, introduces additional complexity and overhead. This option requires creating and managing a Compute Engine VM, which is a virtual machine that runs on Google Cloud. However, using a Compute Engine VM to launch the training may not be necessary or efficient, as it requires installing and configuring all the dependencies and libraries needed to run the code, and maintaining and updating the VM. Moreover, using a Compute Engine VM to launch the training may incur additional cost and latency, as it requires paying for the VM usage and transferring the data and the code between the VM and Vertex AI.

Option C: Creating a Vertex AI Workbench user-managed notebooks instance with 4 V100 GPUs, and using it to train the model, introduces additional cost and risk. This option requires creating and managing a Vertex AI Workbench user-managed notebooks instance, which is a service that allows you to create and run Jupyter notebooks on Google Cloud. However, using a Vertex AI Workbench user-managed notebooks instance to train the model may not be optimal or secure, as it requires paying for the notebooks instance usage, which can be expensive and wasteful, especially if the notebooks instance is not used for other purposes. Moreover, using a Vertex AI Workbench usermanaged notebooks instance to train the model may expose the model and the data to potential security or privacy issues, as the notebooks instance is not fully managed by Google Cloud, and may be accessed or modified by unauthorized users or malicious actors.

Option D: Creating a Google Kubernetes Engine cluster with a node pool that has 4 V100 GPUs. Prepare and submit a

TFJob operator to this node pool, introduces additional complexity and cost. This option requires creating and managing a Google Kubernetes Engine cluster, which is a fully managed service that runs Kubernetes clusters on Google Cloud. Moreover, this option requires creating and managing a node pool that has 4 V100 GPUs, which is a group of nodes that share the same configuration and resources. Furthermore, this option requires preparing and submitting a TFJob operator to this node pool, which is a Kubernetes custom resource that defines a TensorFlow training job. However, using Google Kubernetes Engine, node pool, and TFJob operator to train the model may not be necessary or efficient, as it requires configuring and maintaining the cluster, the node pool, and the TFJob operator, and paying for their usage. Moreover, using Google Kubernetes Engine, node pool, and TFJob operator to train the model may not be compatible or scalable, as they are designed for TensorFlow models, not PyTorch models, and may not support distributed or parallel training.

Reference:

[Vertex AI: Training with custom containers]

[Vertex AI: Using custom machine types]

[Setuptools documentation]

[PyTorch documentation]

[ResNet50 | PyTorch]

Question: 141

You are an ML engineer at a manufacturing company You are creating a classification model for a predictive maintenance use case You need to predict whether a crucial machine will fail in the next three days so that the repair crew has enough time to fix the machine before it breaks. Regular maintenance of the machine is relatively inexpensive, but a failure would be very costly You have trained several binary classifiers to predict whether the machine will fail. where a prediction of 1 means that the ML model predicts a failure.

You are now evaluating each model on an evaluation dataset. You want to choose a model that prioritizes detection while ensuring that more than 50% of the maintenance jobs triggered by your model address an imminent machine failure. Which model should you choose?

A. The model with the highest area under the receiver operating characteristic curve (AUC ROC) and precision greater than 0 5

B. The model with the lowest root mean squared error (RMSE) and recall greater than 0.5.

C. The model with the highest recall where precision is greater than 0.5.

D. The model with the highest precision where recall is greater than 0.5.

Answer: C

Explanation:

The best option for choosing a model that prioritizes detection while ensuring that more than 50% of the maintenance jobs triggered by the model address an imminent machine failure is to choose the model with the highest recall where

precision is greater than 0.5. This option has the following advantages:

It maximizes the recall, which is the proportion of actual failures that are correctly predicted by the model. Recall is also known as sensitivity or true positive rate (TPR), and it is calculated as:

mathrmRecallfracmathrmTPmathrmTP+mathrmFN

where TP is the number of true positives (actual failures that are predicted as failures) and FN is the number of false negatives (actual failures that are predicted as non-failures). By maximizing the recall, the model can reduce the number of false negatives, which are the most costly and undesirable outcomes for the predictive maintenance use case, as they represent missed failures that can lead to machine breakdown and downtime.

It constrains the precision, which is the proportion of predicted failures that are actual failures. Precision is also known as positive predictive value (PPV), and it is calculated as:

mathrmPrecisionfracmathrmTPmathrmTP+mathrmFP

where FP is the number of false positives (actual non-failures that are predicted as failures). By constraining the precision to be greater than 0.5, the model can ensure that more than 50% of the maintenance jobs triggered by the model address an imminent machine failure, which can avoid unnecessary or wasteful maintenance costs.

The other options are less optimal for the following reasons:

Option A: Choosing the model with the highest area under the receiver operating characteristic curve (AUC ROC) and precision greater than 0.5 may not prioritize detection, as the AUC ROC does not directly measure the recall. The AUC ROC is a summary metric that evaluates the overall

performance of a binary classifier across all possible thresholds. The ROC curve plots the TPR (recall) against the false positive rate (FPR), which is the proportion of actual non-failures that are incorrectly predicted by the model. The AUC ROC is the area under the ROC curve, and it ranges from 0 to 1, where 1 represents a perfect classifier. However, choosing the model with the highest AUC ROC may not maximize the recall, as the AUC ROC is influenced by both the TPR and the FPR, and it does not account for the precision or the specificity (the proportion of actual non-failures that are correctly predicted by the model).

Option B: Choosing the model with the lowest root mean squared error (RMSE) and recall greater than 0.5 may not prioritize detection, as the RMSE is not a suitable metric for binary classification. The RMSE is a regression metric that measures the average magnitude of the error between the predicted and the actual values. The RMSE is calculated as: mathrmRMSEsqrtfrac1nsumi1n(yi-hatyi)2 where yi is the actual value, hatyi is the predicted value, and n is the number of observations. However, choosing the model with the lowest RMSE may not optimize the detection of failures, as the RMSE is sensitive to outliers and does not account for the class imbalance or the cost of misclassification.

Option D: Choosing the model with the highest precision where recall is greater than 0.5 may not prioritize detection, as the precision may not be the most important metric for the predictive maintenance use case. The precision measures the accuracy of the positive predictions, but it does not reflect the sensitivity or the coverage of the model. By choosing the model with the highest precision, the model may sacrifice the recall, which is the proportion of actual failures that are correctly predicted by the model. This may increase the number of false negatives, which are the most costly and undesirable outcomes for the predictive maintenance use case, as they represent missed failures that can lead to machine breakdown and downtime.

Reference:

Evaluation Metrics (Classifiers) - Stanford University
Evaluation of binary classifiers - Wikipedia
Predictive Maintenance: The greatest benefits and smart use cases

www.atmicnetworks.com

Question: 142

Your organization manages an online message board A few months ago, you discovered an increase in toxic language and bullying on the message board. You deployed an automated text classifier that flags certain comments as toxic or harmful. Now some users are reporting that benign comments referencing their religion are being misclassified as abusive Upon further inspection, you find that your classifier's false positive rate is higher for comments that reference certain underrepresented religious groups. Your team has a limited budget and is already overextended. What should you do?

A. Add synthetic training data where those phrases are used in non-toxic ways B. Remove the model and replace it with human moderation.

C. Replace your model with a different text classifier.

D. Raise the threshold for comments to be considered toxic or harmful

Answer: A

Explanation:

The problem of the text classifier is that it has a high false positive rate for comments that reference certain underrepresented religious groups. This means that the classifier is not able to distinguish between toxic and non-toxic language when those groups are mentioned. One possible reason for this is that the training data does not have enough examples of non-toxic comments that reference those groups, leading to a biased model. Therefore, a possible solution is to add synthetic training data where those phrases are used in non-toxic ways, which can help the model learn to generalize better and reduce the false positive rate. Synthetic data is artificially generated data that mimics the characteristics of real data, and can be used to augment the existing data when the real data is scarce or imbalanced.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 5: Responsible AI, Week 3: Fairness

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 4: Ensuring solution quality, 4.4 Evaluating fairness and bias in ML models

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 9: Responsible AI, Section 9.3: Fairness and Bias

Question: 143

You work on the data science team at a manufacturing company. You are reviewing the company's historical sales data, which has hundreds of millions of records. For your exploratory data analysis, you need to calculate descriptive statistics such as mean, median, and mode; conduct complex statistical tests for hypothesis testing; and plot variations of the features over time You want to use as much of the sales data as possible in your analyses while minimizing computational resources. What should you do?

A. Spin up a Vertex AI Workbench user-managed notebooks instance and import the dataset Use this data to create statistical and visual analyses

B. Visualize the time plots in Google Data Studio. Import the dataset into Vertex AI Workbench user-managed

notebooks Use this data to calculate the descriptive statistics and run the statistical analyses

C. Use BigQuery to calculate the descriptive statistics. Use Vertex Al Workbench user-managed notebooks to visualize the time plots and run the statistical analyses.

D Use BigQuery to calculate the descriptive statistics, and use Google Data Studio to visualize the time plots. Use Vertex Al Workbench user-managed notebooks to run the statistical analyses.

Answer: C

Explanation:

The best option for analyzing large and complex datasets while minimizing computational resources is to use a combination of BigQuery and Vertex AI Workbench. BigQuery is a serverless, scalable, and cost-effective data warehouse that can perform fast and interactive queries on petabytes of data. BigQuery can calculate descriptive statistics such as mean, median, and mode by using SQL functions such as AVG, PERCENTILE_CONT, and MODE. Vertex AI Workbench is a managed service that provides an integrated development environment for data science and machine learning. Vertex AI Workbench allows users to create and run Jupyter notebooks on Google Cloud, and access various tools and libraries for data visualization and statistical analysis. Vertex AI Workbench can connect to BigQuery and use the results of the queries to create time plots and run statistical tests for hypothesis testing. By using BigQuery and Vertex AI Workbench, users can leverage the power and flexibility of Google Cloud to perform exploratory data analysis on large and complex datasets. Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 2: Data Engineering for ML on Google Cloud, Week 1: Introduction to Data Engineering for ML

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 1: Architecting low-code ML solutions, 1.1 Developing ML models by using BigQuery ML

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 3: Data Engineering for ML, Section 3.2: BigQuery for ML

Question: 144

You have trained a DNN regressor with TensorFlow to predict housing prices using a set of predictive features. Your default precision is tf.float64, and you use a standard TensorFlow estimator;

estimator tf.estimator.DNNRegressor(feature_columns[YOUR_LIST_OF_FEATURES], hidden_units-[1024, 512, 256], dropoutNone)

Your model performs well, but Just before deploying it to production, you discover that your current serving latency is 10ms @ 90 percentile and you currently serve on CPUs. Your production requirements expect a model latency of 8ms @ 90 percentile. You are willing to accept a small decrease in performance in order to reach the latency requirement Therefore your plan is to improve latency while evaluating how much the model's prediction decreases. What should you first try to quickly lower the serving latency?

A. Increase the dropout rate to 0.8 in_PREDICT mode by adjusting the TensorFlow Serving parameters

B. Increase the dropout rate to 0.8 and retrain your model.

C. Switch from CPU to GPU serving

D. Apply quantization to your SavedModel by reducing the floating point precision to tf.float16.

Answer: D

Explanation:
Quantization is a technique that reduces the numerical precision of the weights and activations of a neural network, which can improve the inference speed and reduce the memory footprint of the model1.
Reducing the floating point precision from tf.float64 to tf.float16 can potentially halve the latency and memory usage of the model, while having minimal impact on the accuracy2.
Increasing the dropout rate to 0.8 in either mode would not affect the latency, but would likely degrade the performance of the model significantly, as dropout is a regularization technique that randomly drops out units during training to prevent overfitting3.
Switching from CPU to GPU serving may or may not improve the latency, depending on the hardware specifications and the model complexity, but it would also incur additional costs and complexity for deployment4

Question: 145

You are training an ML model using data stored in BigQuery that contains several values that are considered Personally Identifiable Information (PII). You need to reduce the sensitivity of the dataset before training your model. Every column is critical to your model. How should you proceed?

A. Using Dataflow, ingest the columns with sensitive data from BigQuery, and then randomize the values in each sensitive column.
B. Use the Cloud Data Loss Prevention (DLP) API to scan for sensitive data, and use Dataflow with the DLP API to encrypt sensitive values with Format Preserving Encryption
C. Use the Cloud Data Loss Prevention (DLP) API to scan for sensitive data, and use Dataflow to replace all sensitive data by using the encryption algorithm AES-256 with a salt.
D. Before training, use BigQuery to select only the columns that do not contain sensitive data Create an authorized view of the data so that sensitive values cannot be accessed by unauthorized individuals.

Answer: B

Explanation:
The best option for reducing the sensitivity of the dataset before training the model is to use the Cloud Data Loss Prevention (DLP) API to scan for sensitive data, and use Dataflow with the DLP API to encrypt sensitive values with Format Preserving Encryption. This option allows you to keep every column in the dataset, while protecting the sensitive data from unauthorized access or exposure. The Cloud DLP API can detect and classify various types of sensitive data, such as names, email addresses, phone numbers, credit card numbers, and more1. Dataflow can create scalable and reliable pipelines to process large volumes of data from BigQuery and other sources2. Format Preserving Encryption (FPE) is a technique that encrypts sensitive data while preserving its original format and length, which can help maintain the utility and validity of the data3. By using Dataflow
with the DLP API, you can apply FPE to the sensitive values in the dataset, and store the encrypted data in BigQuery or another destination. You can also use the same pipeline to decrypt the data when needed, by using the same encryption key and method4.
The other options are not as suitable as option B, for the following reasons:

Option A: Using Dataflow to ingest the columns with sensitive data from BigQuery, and then randomize the values in each sensitive column, would reduce the sensitivity of the data, but also the utility and accuracy of the data. Randomization is a technique that replaces sensitive data with random values, which can prevent re-identification of the data, but also distort the distribution and relationships of the data3. This can affect the performance and quality of the ML model, especially if every column is critical to the model.

Option C: Using the Cloud DLP API to scan for sensitive data, and use Dataflow to replace all sensitive data by using the encryption algorithm AES-256 with a salt, would reduce the sensitivity of the data, but also the utility and validity of the data. AES-256 is a symmetric encryption algorithm that uses a 256-bit key to encrypt and decrypt data. A salt is a random value that is added to the data before encryption, to increase the randomness and security of the encrypted data. However, AES-256 does not preserve the format or length of the original data, which can cause problems when storing or processing the data. For example, if the original data is a 10-digit phone number, AES-256 would produce a much longer and different string, which can break the schema or logic of the dataset3. Option D: Before training, using BigQuery to select only the columns that do not contain sensitive data, and creating an authorized view of the data so that sensitive values cannot be accessed by unauthorized individuals, would reduce the exposure of the sensitive data, but also the completeness and relevance of the data. An authorized view is a BigQuery view that allows you to share query results with particular users or groups, without giving them access to the underlying tables. However, this option assumes that you can identify the columns that do not contain sensitive data, which may not be easy or accurate. Moreover, this option would remove some columns from the dataset, which can affect the performance and quality of the ML model, especially if every column is critical to the model.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 5: Responsible AI, Week 2: Privacy

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 5: Developing responsible AI solutions, 5.2 Implementing privacy techniques

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 9: Responsible AI, Section 9.4: Privacy

De-identification techniques

Cloud Data Loss Prevention (DLP) API

Dataflow

Using Dataflow and Sensitive Data Protection to securely tokenize and import data from a relational database to BigQuery

[AES encryption] [Salt (cryptography)] [Authorized views]

Question: 146

You want to train an AutoML model to predict house prices by using a small public dataset stored in BigQuery. You need to prepare the data and want to use the simplest most efficient approach. What should you do?

A. Write a query that preprocesses the data by using BigQuery and creates a new table Create a Vertex AI managed dataset with the new table as the data source.

B. Use Dataflow to preprocess the data Write the output in TFRecord format to a Cloud Storage bucket.

C. Write a query that preprocesses the data by using BigQuery Export the query results as CSV files and use those files to create a Vertex AI managed dataset.

D. Use a Vertex AI Workbench notebook instance to preprocess the data by using the pandas library Export the data as CSV files, and use those files to create a Vertex AI managed dataset.

Answer: A

Explanation:

The simplest and most efficient approach for preparing the data for AutoML is to use BigQuery and Vertex AI. BigQuery is a serverless, scalable, and cost-effective data warehouse that can perform fast and interactive queries on large datasets. BigQuery can preprocess the data by using SQL functions such as filtering, aggregating, joining, transforming, and creating new features. The preprocessed data can be stored in a new table in BigQuery, which can be used as the data source for Vertex AI. Vertex AI is a unified platform for building and deploying machine learning solutions on Google Cloud. Vertex AI can create a managed dataset from a BigQuery table, which can be used to train an AutoML model. Vertex AI can also evaluate, deploy, and monitor the AutoML model, and provide online or batch predictions. By using BigQuery and Vertex AI, users can leverage the power and simplicity of Google Cloud to train an AutoML model to predict house prices.

The other options are not as simple or efficient as option A, for the following reasons:

Option B: Using Dataflow to preprocess the data and write the output in TFRecord format to a Cloud Storage bucket would require more steps and resources than using BigQuery and Vertex AI. Dataflow is a service that can create scalable and reliable pipelines to process large volumes of data from various sources. Dataflow can preprocess the data by using Apache Beam, a programming model for defining and executing data processing workflows. TFRecord is a binary file format that can store sequential data efficiently. However, using Dataflow and TFRecord would require writing code, setting up a pipeline, choosing a runner, and managing the output files. Moreover, TFRecord is not a supported format for Vertex AI managed datasets, so the data would need to be converted to CSV or JSONL files before creating a Vertex AI managed dataset.

Option C: Writing a query that preprocesses the data by using BigQuery and exporting the query results as CSV files would require more steps and storage than using BigQuery and Vertex AI. CSV is a text file format that can store tabular data in a comma-separated format. Exporting the query results as CSV files would require choosing a destination Cloud Storage bucket, specifying a file name or a wildcard, and setting the export options. Moreover, CSV files can have limitations such as size, schema, and encoding, which can affect the quality and validity of the data. Exporting the data as CSV files would also incur additional storage costs and reduce the performance of the queries. Option D: Using a Vertex AI Workbench notebook instance to preprocess the data by using the pandas library and exporting the data as CSV files would require more steps and skills than using BigQuery and Vertex AI. Vertex AI Workbench is a service that provides an integrated development environment for data science and machine learning. Vertex AI Workbench allows users to create and run Jupyter notebooks on Google Cloud, and access various tools and libraries for data analysis and machine learning. Pandas is a popular Python library that can manipulate and analyze data in a tabular format. However, using Vertex AI Workbench and pandas would require creating a notebook instance, writing Python code, installing and importing pandas, connecting to BigQuery, loading and preprocessing the data, and exporting the data as CSV files. Moreover, pandas can have limitations such as memory usage, scalability, and compatibility, which can affect the efficiency and reliability of the data processing.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 2: Data Engineering for ML on Google Cloud, Week 1: Introduction to Data Engineering for ML

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 1: Architecting low-code ML solutions, 1.3 Training models by using AutoML

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 4: Low- code ML Solutions, Section 4.3: AutoML BigQuery

Vertex AI

Dataflow

TFRecord

CSV

Vertex AI Workbench

Pandas

www.atmicnetworks.com

Question: 147

You developed a Vertex Al ML pipeline that consists of preprocessing and training steps and each set of steps runs on a separate custom Docker image Your organization uses GitHub and GitHub Actions as CI/CD to run unit and integration tests You need to automate the model retraining workflow so that it can be initiated both manually and when a new version of the code is merged in the main branch You want to minimize the steps required to build the workflow while also allowing for maximum flexibility How should you configure the CI/CD workflow?

A. Trigger a Cloud Build workflow to run tests build custom Docker images, push the images to Artifact Registry and launch the pipeline in Vertex Al Pipelines.

B. Trigger GitHub Actions to run the tests launch a job on Cloud Run to build custom Docker images push the images to Artifact Registry and launch the pipeline in Vertex Al Pipelines.

C. Trigger GitHub Actions to run the tests build custom Docker images push the images to Artifact Registry, and launch the pipeline in Vertex Al Pipelines.

D. Trigger GitHub Actions to run the tests launch a Cloud Build workflow to build custom Dicker images, push the images to Artifact Registry, and launch the pipeline in Vertex Al Pipelines.

Answer: D

Explanation:

The best option for automating the model retraining workflow is to use GitHub Actions and Cloud Build. GitHub Actions is a service that can create and run workflows for continuous integration and continuous delivery (CI/CD) on GitHub. GitHub Actions can run tests, build and deploy code, and trigger other actions based on events such as code changes, pull requests, or manual triggers. Cloud Build is a service that can create and run scalable and reliable pipelines to build, test, and deploy software on Google Cloud. Cloud Build can build custom Docker images, push the images to Artifact Registry, and launch the pipeline in Vertex AI Pipelines. Vertex AI Pipelines is a service that can orchestrate machine learning (ML) workflows using Vertex AI. Vertex AI Pipelines can run preprocessing and training steps on custom Docker images, and evaluate, deploy, and monitor the ML model. By using GitHub Actions and Cloud Build, users can leverage the power and flexibility of Google Cloud to automate the model retraining workflow, while minimizing the steps required to build the workflow.

The other options are not as good as option D, for the following reasons:

Option A: Triggering a Cloud Build workflow to run tests, build custom Docker images, push the images to Artifact Registry, and launch the pipeline in Vertex AI Pipelines would require more configuration and maintenance than using GitHub Actions and Cloud Build. Cloud Build is a service that can create and run pipelines to build, test, and deploy software on Google Cloud, but it is not designed to integrate with GitHub or other source code repositories. To trigger a Cloud Build workflow from GitHub, users would need to set up a webhook, a Cloud Pub/Sub topic, and a Cloud Function1. Moreover, Cloud Build does not support manual triggers, which limits the flexibility of the workflow2.

Option B: Triggering GitHub Actions to run the tests, launching a job on Cloud Run to build custom Docker images, pushing the images to Artifact Registry, and launching the pipeline in Vertex AI Pipelines would require more steps and resources than using GitHub Actions and Cloud Build. Cloud Run is a service that can run stateless containers on a fully managed environment or on Anthos. Cloud Run can build custom Docker images, but it is not optimized for this task. Users would need to write a Dockerfile, a cloudbuild.yaml file, and a Cloud Run service configuration file, and use the gcloud command-line tool to build and deploy the image3. Moreover, Cloud Run is designed for serving HTTP requests, not for running ML pipelines, which can have different performance and scalability requirements.

Option C: Triggering GitHub Actions to run the tests, building custom Docker images, pushing the images to Artifact Registry, and launching the pipeline in Vertex AI Pipelines would require more skills and tools than using GitHub Actions and Cloud Build. GitHub Actions can run tests and build code, but it is not specialized for building Docker images. Users would need to install and configure Docker on the GitHub Actions runner, write a Dockerfile, and use the docker command-line tool to build and push the image. Moreover, GitHub Actions has limitations on the disk space, memory,

and CPU of the runner, which can affect the speed and reliability of the image building process. Reference:

Building CI/CD for Vertex AI pipelines: The first solution

Cloud Build

GitHub Actions

Vertex AI Pipelines

Triggering builds from GitHub

Triggering builds manually

Building containers

Cloud Run

[Building and testing Docker images with GitHub Actions] [Usage limits, billing, and administration]

Question: 148

You are working with a dataset that contains customer transactions. You need to build an ML model to predict customer purchase behavior You plan to develop the model in BigQuery ML, and export it to Cloud Storage for online prediction You notice that the input data contains a few categorical features, including product category and payment method You want to deploy the model as quickly as possible. What should you do?

A. Use the transform clause with the ML. ONE_HOT_ENCODER function on the categorical features at model creation and select the categorical and non-categorical features.

B. Use the ML. ONE_HOT_ENCODER function on the categorical features, and select the encoded categorical features and non-categorical features as inputs to create your model.

C. Use the create model statement and select the categorical and non-categorical features.

D. Use the ML. ONE_HOT_ENCODER function on the categorical features, and select the encoded categorical features and non-categorical features as inputs to create your model.

Answer: A

Explanation:

The best option for building an ML model to predict customer purchase behavior in BigQuery ML is to use the transform clause with the ML.ONE_HOT_ENCODER function on the categorical features at model creation and select the categorical and non-categorical features. This option allows you to encode the categorical features as one-hot vectors, which are binary vectors that have only one nonzero element. One-hot encoding is a common technique for handling categorical features in ML models, as it can reduce the dimensionality and sparsity of the data, and avoid the ordinality problem that arises when using numerical labels for categorical values1. The transform clause is a feature of BigQuery ML that lets you apply SQL expressions to transform the input data at model creation time. The transform clause can perform feature engineering, such as one-hot encoding, on the fly, without requiring you to create and store a new table with the transformed data2. By using the transform clause with the ML.ONE_HOT_ENCODER function, you can create and train an ML model in BigQuery ML with a single SQL statement, and export it to Cloud Storage for online prediction.

The other options are not as good as option A, for the following reasons:

Option B: Using the ML.ONE_HOT_ENCODER function on the categorical features, and selecting the encoded categorical features and non-categorical features as inputs to create your model, would require more steps and storage than using the transform clause. The ML.ONE_HOT_ENCODER function is a BigQuery ML function that returns a one-hot encoded vector for a given categorical value. However, using this function alone would not apply the one-hot encoding to the input data at model creation time. You would need to create a new table with the encoded features, and use that table as the input to create your model. This would incur additional storage costs and reduce the performance of the

queries.

Option C: Using the create model statement and selecting the categorical and non-categorical features, would not handle the categorical features properly and could result in a poor model performance. The create model statement is a BigQuery ML statement that creates and trains an ML model from a SQL query. However, if the input data contains categorical features, you need to encode them as one-hot vectors or use the category_count option to specify the number of categories for each feature. Otherwise, BigQuery ML would treat the categorical features as numerical values, which can introduce bias and noise into the model3.

Option D: Using the ML.ONE_HOT_ENCODER function on the categorical features, and selecting the encoded categorical features and non-categorical features as inputs to create your model, is the same as option B, and has the same drawbacks.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 2: Data Engineering for ML on Google Cloud, Week 2: Feature Engineering

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 1: Architecting low-code ML solutions, 1.1 Developing ML models by using BigQuery ML

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 3: Data Engineering for ML, Section 3.2: BigQuery for ML

One-hot encoding

Using the TRANSFORM clause for feature engineering

Creating a model

ML.ONE_HOT_ENCODER function

Question: 149

You need to develop an image classification model by using a large dataset that contains labeled images in a Cloud Storage Bucket. What should you do?

A. Use Vertex AI Pipelines with the Kubeflow Pipelines SDK to create a pipeline that reads the images from Cloud Storage and trains the model.

B. Use Vertex AI Pipelines with TensorFlow Extended (TFX) to create a pipeline that reads the images from Cloud Storage and trams the model.

C. Import the labeled images as a managed dataset in Vertex AI: and use AutoML to tram the model.

D. Convert the image dataset to a tabular format using Dataflow Load the data into BigQuery and use BigQuery ML to tram the model.

Answer: C

Explanation:

The best option for developing an image classification model by using a large dataset that contains labeled images in a Cloud Storage bucket is to import the labeled images as a managed dataset in Vertex AI and use AutoML to train the model. This option allows you to leverage the power and simplicity of Google Cloud to create and deploy a high-quality image classification model with minimal code and configuration. Vertex AI is a unified platform for building and deploying machine learning solutions on Google Cloud. Vertex AI can create a managed dataset from a Cloud Storage bucket that contains labeled images, which can be used to train an AutoML model. AutoML is a service that can automatically build and optimize machine learning models for various tasks, such as image classification, object detection, natural language processing, and tabular data analysis. AutoML can handle the complex aspects of machine

learning, such as feature engineering, model architecture, hyperparameter tuning, and model evaluation. AutoML can also evaluate, deploy, and monitor the image classification model, and provide online or batch predictions. By using Vertex AI and AutoML, users can develop an image classification model by using a large dataset with ease and efficiency.

The other options are not as good as option C, for the following reasons:

Option A: Using Vertex AI Pipelines with the Kubeflow Pipelines SDK to create a pipeline that reads the images from Cloud Storage and trains the model would require more skills and steps than using Vertex AI and AutoML. Vertex AI Pipelines is a service that can orchestrate machine learning workflows using Vertex AI. Vertex AI Pipelines can run preprocessing and training steps on custom Docker images, and evaluate, deploy, and monitor the machine learning model. Kubeflow Pipelines

SDK is a Python library that can create and run pipelines on Vertex AI Pipelines or on Kubeflow, an open-source platform for machine learning on Kubernetes. However, using Vertex AI Pipelines and Kubeflow Pipelines SDK would require writing code, building Docker images, defining pipeline components and steps, and managing the pipeline execution and artifacts. Moreover, Vertex AI Pipelines and Kubeflow Pipelines SDK are not specialized for image classification, and users would need to use other libraries or frameworks, such as TensorFlow or PyTorch, to build and train the image classification model.

Option B: Using Vertex AI Pipelines with TensorFlow Extended (TFX) to create a pipeline that reads the images from Cloud Storage and trains the model would require more skills and steps than using Vertex AI and AutoML. TensorFlow Extended (TFX) is a framework that can create and run end-to-end machine learning pipelines on TensorFlow, a popular library for building and training deep learning models. TFX can preprocess the data, train and evaluate the model, validate and push the model, and serve the model for online or batch predictions. However, using Vertex AI Pipelines and TFX would require writing code, building Docker images, defining pipeline components and steps, and managing the pipeline execution and artifacts. Moreover, TFX is not optimized for image classification, and users would need to use other libraries or tools, such as TensorFlow Data Validation, TensorFlow Transform, and TensorFlow Hub, to handle the image data and the model architecture.

Option D: Converting the image dataset to a tabular format using Dataflow, loading the data into BigQuery, and using BigQuery ML to train the model would not handle the image data properly and could result in a poor model performance. Dataflow is a service that can create scalable and reliable pipelines to process large volumes of data from various sources. Dataflow can preprocess the data by using Apache Beam, a programming model for defining and executing data processing workflows. BigQuery is a serverless, scalable, and cost-effective data warehouse that can perform fast and interactive queries on large datasets. BigQuery ML is a service that can create and train machine learning models by using SQL queries on BigQuery. However, converting the image data to a tabular format would lose the spatial and semantic information of the images, which are essential for image classification. Moreover, BigQuery ML is not specialized for image classification, and users would need to use other tools or techniques, such as feature hashing, embedding, or one-hot encoding, to handle the categorical features.

Question: 150

You are developing a mode! to detect fraudulent credit card transactions. You need to prioritize detection because missing even one fraudulent transaction could severely impact the credit card holder. You used AutoML to tram a model on users' profile information and credit card transaction dat

a. After training the initial model, you notice that the model is failing to detect many fraudulent transactions. How should you adjust the training parameters in AutoML to improve model performance?

Choose 2 answers

A. Increase the score threshold.

B. Decrease the score threshold.

C. Add more positive examples to the training set.

D. Add more negative examples to the training set.

E. Reduce the maximum number of node hours for training.

Answer: B, C

Explanation:

The best options for adjusting the training parameters in AutoML to improve model performance are to decrease the score threshold and add more positive examples to the training set. These options can help increase the detection rate of fraudulent transactions, which is the priority for this use case. The score threshold is a parameter that determines the minimum probability score that a prediction must have to be classified as positive. Decreasing the score threshold can increase the recall of the model, which is the proportion of actual positive cases that are correctly identified. Increasing the recall can help reduce the number of false negatives, which are fraudulent transactions that are missed by the model. However, decreasing the score threshold can also decrease the precision of the model, which is the proportion of positive predictions that are actually correct. Decreasing the precision can increase the number of false positives, which are legitimate transactions that are flagged as fraudulent by the model. Therefore, there is a trade-off between recall and precision, and the optimal score threshold depends on the business objective and the cost of errors1. Adding more positive examples to the training set can help balance the data distribution and improve the model performance. Positive examples are the instances that belong to the target class, which in this case are fraudulent transactions. Negative examples are the instances that belong to the other class, which in this case are legitimate transactions. Fraudulent transactions are usually rare and imbalanced compared to legitimate transactions, which can cause the model to be biased towards the majority class and fail to learn the characteristics of the minority class. Adding more positive examples can help the model learn more features and patterns of the fraudulent transactions, and increase the detection rate2.

The other options are not as good as options B and C, for the following reasons:

Option A: Increasing the score threshold would decrease the detection rate of fraudulent transactions, which is the opposite of the desired outcome. Increasing the score threshold would decrease the recall of the model, which is the proportion of actual positive cases that are correctly identified. Decreasing the recall would increase the number of false negatives, which are fraudulent transactions that are missed by the model. Increasing the score threshold would increase the precision of the model, which is the proportion of positive predictions that are actually correct. Increasing the precision would decrease the number of false positives, which are legitimate transactions that are flagged as fraudulent by the model. However, in this use case, the cost of false negatives is much higher than the cost of false positives, so increasing the score threshold is not a good option1.

Option D: Adding more negative examples to the training set would not improve the model performance, and could worsen the data imbalance. Negative examples are the instances that belong to the other class, which in this case are legitimate transactions. Legitimate transactions are usually abundant and dominant compared to fraudulent transactions, which can cause the model to be biased towards the majority class and fail to learn the characteristics of the minority class. Adding more negative examples would exacerbate this problem, and decrease the detection rate of the fraudulent transactions2.

Option E: Reducing the maximum number of node hours for training would not improve the model performance, and could limit the model optimization. Node hours are the units of computation that are used to train an AutoML model. The maximum number of node hours is a parameter that determines the upper limit of node hours that can be used for training. Reducing the maximum number of node hours would reduce the training time and cost, but also the model quality and accuracy. Reducing the maximum number of node hours would limit the number of iterations, trials, and evaluations that the model can perform, and prevent the model from finding the optimal hyperparameters and architecture3.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 5: Responsible AI, Week 4: Evaluation

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 2: Developing high-quality ML models, 2.2 Handling imbalanced data

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 4: Low-code ML

Solutions, Section 4.3: AutoML

Understanding the score threshold slider

Handling imbalanced data sets in machine learning

AutoML Vision pricing

Question: 151

You need to deploy a scikit-learn classification model to production. The model must be able to serve requests 24/7 and you expect millions of requests per second to the production application from 8 am to 7 pm. You need to minimize the cost of deployment What should you do?

A. Deploy anonline Vertex Al prediction endpoint Set the max replica count to 1

B. Deploy anonline Vertex Al prediction endpoint Set the max replica count to 100

C. Deploy anonline Vertex Al prediction endpoint with one GPU per replica Set the max replica count to 1.

D. Deploy an online Vertex Al prediction endpoint with one GPU per replica Set the max replica count to 100.

Answer: B

Explanation:

The best option for deploying a scikit-learn classification model to production is to deploy an online Vertex AI prediction endpoint and set the max replica count to 100. This option allows you to leverage the power and scalability of Google Cloud to serve requests 24/7 and handle millions of requests per second. Vertex AI is a unified platform for building and deploying machine learning solutions on Google Cloud. Vertex AI can deploy a trained scikit-learn model to an online prediction endpoint, which can provide low-latency predictions for individual instances. An online prediction endpoint consists of one or more replicas, which are copies of the model that run on virtual machines. The max replica count is a parameter that determines the maximum number of replicas that can be created for the endpoint. By setting the max replica count to 100, you can enable the endpoint to scale up to 100 replicas when the traffic increases, and scale down to zero replicas when the traffic decreases. This can help minimize the cost of deployment, as you only pay for the resources that you use. Moreover, you can use the autoscaling algorithm option to optimize the scaling behavior of the endpoint based on the latency and utilization metrics1.

The other options are not as good as option B, for the following reasons:

Option A: Deploying an online Vertex AI prediction endpoint and setting the max replica count to 1 would not be able to serve requests 24/7 and handle millions of requests per second. Setting the max replica count to 1 would limit the endpoint to only one replica, which can cause performance issues and service disruptions when the traffic increases. Moreover, setting the max replica count to 1 would prevent the endpoint from scaling down to zero replicas when the traffic decreases, which can increase the cost of deployment, as you pay for the resources that you do not use1.

Option C: Deploying an online Vertex AI prediction endpoint with one GPU per replica and setting the max replica count to 1 would not be able to serve requests 24/7 and handle millions of requests per second, and would increase the cost of deployment. Adding a GPU to each replica would increase the computational power of the endpoint, but it would also increase the cost of deployment, as GPUs are more expensive than CPUs. Moreover, setting the max replica count to 1 would limit the endpoint to only one replica, which can cause performance issues and service disruptions when the traffic increases, and prevent the endpoint from scaling down to zero replicas when the traffic decreases1. Furthermore, scikit-learn models do not benefit from GPUs, as scikit-learn is not optimized for GPU acceleration2.

Option D: Deploying an online Vertex AI prediction endpoint with one GPU per replica and setting the max replica count to 100 would be able to serve requests 24/7 and handle millions of requests per second, but it would increase the cost of deployment. Adding a GPU to each replica would increase the computational power of the endpoint, but it would

also increase the cost of deployment, as GPUs are more expensive than CPUs. Setting the max replica count to 100 would enable the endpoint to scale up to 100 replicas when the traffic increases, and scale down to zero replicas when the traffic decreases, which can help minimize the cost of deployment. However, scikit-learn models do not benefit from GPUs, as scikit-learn is not optimized for GPU acceleration2. Therefore, using GPUs for scikit-learn models would be unnecessary and wasteful.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 3: Production ML Systems, Week 2: Serving ML Predictions

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 3: Scaling ML models in production, 3.1 Deploying ML models to production

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 6: Production ML Systems, Section 6.2: Serving ML Predictions

Online prediction

Scaling online prediction scikit-learn FAQ

Question: 152

You work with a team of researchers to develop state-of-the-art algorithms for financial analysis. Your team develops and debugs complex models in TensorFlow. You want to maintain the ease of debugging while also reducing the model training time. How should you set up your training environment?

A. Configure a v3-8 TPU VM SSH into the VM to tram and debug the model.

B. Configure a v3-8 TPU node Use Cloud Shell to SSH into the Host VM to train and debug the model. C. Configure a M-standard-4 VM with 4 NVIDIA P100 GPUs SSH into the VM and use Parameter Server Strategy to train the model.

D. Configure a M-standard-4 VM with 4 NVIDIA P100 GPUs SSH into the VM and use MultiWorkerMirroredStrategy to train the model.

Answer: A

Explanation:

A TPU VM is a virtual machine that has direct access to a Cloud TPU device. TPU VMs provide a simpler and more flexible way to use Cloud TPUs, as they eliminate the need for a separate host VM and network setup. TPU VMs also support interactive debugging tools such as TensorFlow Debugger (tfdbg) and Python Debugger (pdb), which can help researchers develop and troubleshoot complex models. A v3-8 TPU VM has 8 TPU cores, which can provide high performance and scalability for training large models. SSHing into the TPU VM allows the user to run and debug the TensorFlow code directly on the TPU device, without any network overhead or data transfer issues. Reference: 1: TPU VMs Overview

2: TPU VMs Quickstart

3: Debugging TensorFlow Models on Cloud TPUs

Question: 153

You created an ML pipeline with multiple input parameters. You want to investigate the tradeoffs between different parameter combinations. The parameter options are

- input dataset

- Max tree depth of the boosted tree regressor
- Optimizer learning rate

You need to compare the pipeline performance of the different parameter combinations measured in F1 score, time to train and model complexity. You want your approach to be reproducible and track all pipeline runs on the same platform. What should you do?

A. 1 Use BigQueryML to create a boosted tree regressor and use the hyperparameter tuning capability
2 Configure the hyperparameter syntax to select different input datasets. max tree depths, and optimizer teaming rates Choose the grid search option

B. 1 Create a Vertex AI pipeline with a custom model training job as part of the pipeline Configure the pipeline's parameters to include those you are investigating
2 In the custom training step, use the Bayesian optimization method with F1 score as the target to maximize

C. 1 Create a Vertex AI Workbench notebook for each of the different input datasets
2 In each notebook, run different local training jobs with different combinations of the max tree depth and optimizer learning rate parameters
3 After each notebook finishes, append the results to a BigQuery table

D. 1 Create an experiment in Vertex AI Experiments
2. Create a Vertex AI pipeline with a custom model training job as part of the pipeline. Configure the pipelines parameters to include those you are investigating
3. Submit multiple runs to the same experiment using different values for the parameters

Answer: D

Explanation:
The best option for investigating the tradeoffs between different parameter combinations is to create an experiment in Vertex AI Experiments, create a Vertex AI pipeline with a custom model training job as part of the pipeline, configure the pipeline's parameters to include those you are investigating, and submit multiple runs to the same experiment using different values for the parameters. This option allows you to leverage the power and flexibility of Google Cloud to compare
the pipeline performance of the different parameter combinations measured in F1 score, time to train, and model complexity. Vertex AI Experiments is a service that can track and compare the results of multiple machine learning runs. Vertex AI Experiments can record the metrics, parameters, and artifacts of each run, and display them in a dashboard for easy visualization and analysis. Vertex AI Experiments can also help users optimize the hyperparameters of their models by using different search algorithms, such as grid search, random search, or Bayesian optimization1. Vertex AI Pipelines is a service that can orchestrate machine learning workflows using Vertex AI. Vertex AI Pipelines can run preprocessing and training steps on custom Docker images, and evaluate, deploy, and monitor the machine learning model. A custom model training job is a type of pipeline step that can train a custom model by using a user-provided script or container. A custom model training job can accept pipeline parameters as inputs, which can be used to control the training logic or data source. By creating an experiment in Vertex AI Experiments, creating a Vertex AI pipeline with a custom model training job as part of the pipeline, configuring the pipeline's parameters to include those you are investigating, and submitting multiple runs to the same experiment using different values for the parameters, you can create a reproducible and trackable approach to investigate the tradeoffs between different parameter combinations.
The other options are not as good as option D, for the following reasons:
Option A: Using BigQuery ML to create a boosted tree regressor and use the hyperparameter tuning capability, configuring the hyperparameter syntax to select different input datasets, max tree depths, and optimizer learning rates, and choosing the grid search option would not be able to handle different input datasets as a hyperparameter, and would not be as flexible and scalable as using Vertex AI Experiments and Vertex AI Pipelines. BigQuery ML is a service

that can create and train machine learning models by using SQL queries on BigQuery. BigQuery ML can perform hyperparameter tuning by using the ML.FORECAST or ML.PREDICT functions, and specifying the hyperparameters option. BigQuery ML can also use different search algorithms, such as grid search, random search, or Bayesian optimization, to find the optimal hyperparameters. However, BigQuery ML can only tune the hyperparameters that are related to the model architecture or training process, such as max tree depth or learning rate. BigQuery ML cannot tune the hyperparameters that are related to the data source, such as input dataset. Moreover, BigQuery ML is not designed to work with Vertex AI Experiments or Vertex AI Pipelines, which can provide more features and flexibility for tracking and orchestrating machine learning workflows2.

Option B: Creating a Vertex AI pipeline with a custom model training job as part of the pipeline, configuring the pipeline's parameters to include those you are investigating, and using the Bayesian optimization method with F1 score as the target to maximize in the custom training step would not be able to track and compare the results of multiple runs, and would require more skills and steps than using Vertex AI Experiments and Vertex AI Pipelines. Vertex AI Pipelines is a service that can orchestrate machine learning workflows using Vertex AI. Vertex AI Pipelines can run preprocessing and training steps on custom Docker images, and evaluate, deploy, and monitor the machine learning model. A custom model training job is a type of pipeline step that can train a custom model by using a user-provided script or container. A custom model training job can accept pipeline parameters as inputs, which can be used to control the training logic or data source. However, using the Bayesian optimization method with F1 score as the target to maximize in the custom training step would require writing code, implementing the optimization algorithm, and defining the objective function. Moreover, this option would not be able to track and compare the results of multiple runs, as Vertex AI Pipelines does not have a built-in feature for recording and displaying the metrics, parameters, and artifacts of each run3.

Option C: Creating a Vertex AI Workbench notebook for each of the different input datasets, running different local training jobs with different combinations of the max tree depth and optimizer learning rate parameters, and appending the results to a BigQuery table would not be able to track and compare the results of multiple runs on the same platform, and would require more skills and steps than using Vertex AI Experiments and Vertex AI Pipelines. Vertex AI Workbench is a service that provides an integrated development environment for data science and machine learning. Vertex AI Workbench allows users to create and run Jupyter notebooks on Google Cloud, and access various tools and libraries for data analysis and machine learning. However, creating a Vertex AI Workbench notebook for each of the different input datasets, running different local training jobs with different combinations of the max tree depth and optimizer learning rate parameters, and appending the results to a BigQuery table would require creating multiple notebooks, writing code, setting up local environments, connecting to BigQuery, loading and preprocessing the data, training and evaluating the model, and writing the results to a BigQuery table. Moreover, this option would not be able to track and compare the results of multiple runs on the same platform, as BigQuery is a separate service from Vertex AI Workbench, and does not have a dashboard for visualizing and analyzing the metrics, parameters, and artifacts of each run4.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 3: Production ML Systems, Week 3: MLOps

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 1: Architecting low-code ML solutions, 1.1 Developing ML models by using BigQuery ML

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 3: Data Engineering for ML, Section 3.2: BigQuery for ML Vertex AI Experiments

Vertex AI Pipelines

BigQuery ML

Vertex AI Workbench

www.atmicnetworks.com

Question: 154

You received a training-serving skew alert from a Vertex AI Model Monitoring job running in production. You retrained the model with more recent training data, and deployed it back to the Vertex AI endpoint but you are still receiving the same alert. What should you do?

A. Update the model monitoring job to use a lower sampling rate.
B. Update the model monitoring job to use the more recent training data that was used to retrain the model.
C. Temporarily disable the alert Enable the alert again after a sufficient amount of new production traffic has passed through the Vertex AI endpoint.
D. Temporarily disable the alert until the model can be retrained again on newer training data Retrain the model again after a sufficient amount of new production traffic has passed through the Vertex AI endpoint

Answer: B

Explanation:
The best option for resolving the training-serving skew alert is to update the model monitoring job to use the more recent training data that was used to retrain the model. This option can help align the baseline distribution of the model monitoring job with the current distribution of the production
data, and eliminate the false positive alerts. Model Monitoring is a service that can track and compare the results of multiple machine learning runs. Model Monitoring can monitor the model's prediction input data for feature skew and drift. Training-serving skew occurs when the feature data distribution in production deviates from the feature data distribution used to train the model. If the original training data is available, you can enable skew detection to monitor your models for trainingserving skew. Model Monitoring uses TensorFlow Data Validation (TFDV) to calculate the distributions and distance scores for each feature, and compares them with a baseline distribution. The baseline distribution is the statistical distribution of the feature's values in the training data. If the distance score for a feature exceeds an alerting threshold that you set, Model Monitoring sends you an email alert. However, if you retrain the model with more recent training data, and deploy it back to the Vertex AI endpoint, the baseline distribution of the model monitoring job may become outdated and inconsistent with the current distribution of the production data. This can cause the model monitoring job to generate false positive alerts, even if the model performance is not deteriorated. To avoid this problem, you need to update the model monitoring job to use the more recent training data that was used to retrain the model. This can help the model monitoring job to recalculate the baseline distribution and the distance scores, and compare them with the current distribution of the production data. This can also help the model monitoring job to detect any true positive alerts, such as a sudden change in the production data that causes the model performance to degrade1.
The other options are not as good as option B, for the following reasons:
Option A: Updating the model monitoring job to use a lower sampling rate would not resolve the training-serving skew alert, and could reduce the accuracy and reliability of the model monitoring job. The sampling rate is a parameter that determines the percentage of prediction requests that are logged and analyzed by the model monitoring job. Using a lower sampling rate can reduce the storage and computation costs of the model monitoring job, but also the quality and validity of the data. Using a lower sampling rate can introduce sampling bias and noise into the data, and make the model monitoring job miss some important features or patterns of the data. Moreover, using a lower sampling rate would not address the root cause of the training-serving skew alert, which is the mismatch between the baseline distribution and the current distribution of the production data2. Option C: Temporarily disabling the alert, and enabling the alert again after a sufficient amount of new production traffic has passed through the Vertex AI endpoint, would not resolve the trainingserving skew alert, and could expose the model to potential risks and errors. Disabling the alert would stop the model monitoring job from sending email notifications when the distance score for a feature exceeds the alerting threshold, but it would not stop the model monitoring job from calculating and comparing the distributions and distance scores. Therefore, disabling the alert would not address the root cause of the training-serving skew alert,

which is the mismatch between the baseline distribution and the current distribution of the production data. Moreover, disabling the alert would prevent the model monitoring job from detecting any true positive alerts, such as a sudden change in the production data that causes the model performance to degrade. This can expose the model to potential risks and errors, and affect the user satisfaction and trust1.

Option D: Temporarily disabling the alert until the model can be retrained again on newer training data, and retraining the model again after a sufficient amount of new production traffic has passed through the Vertex AI endpoint, would not resolve the training-serving skew alert, and could cause unnecessary costs and efforts. Disabling the alert would stop the model monitoring job from sending email notifications when the distance score for a feature exceeds the alerting threshold, but it would not stop the model monitoring job from calculating and comparing the distributions and distance scores. Therefore, disabling the alert would not address the root cause of the training-serving skew alert, which is the mismatch between the baseline distribution and the current distribution of the production data. Moreover, disabling the alert would prevent the model monitoring job from detecting any true positive alerts, such as a sudden change in the production data that causes the model performance to degrade. This can expose the model to potential risks and errors, and affect the user satisfaction and trust. Retraining the model again on newer training data would create a new model version, but it would not update the model monitoring job to use the newer training data as the baseline distribution. Therefore, retraining the model again on newer training data would not resolve the training-serving skew alert, and could cause unnecessary costs and efforts1.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 3: Production ML Systems, Week 4: Evaluation

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 3: Scaling ML models in production, 3.3 Monitoring ML models in production

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 6: Production ML Systems, Section 6.3: Monitoring ML Models

Using Model Monitoring

Understanding the score threshold slider

Sampling rate

Question: 155

You developed a custom model by using Vertex AI to forecast the sales of your company s products based on historical transactional data You anticipate changes in the feature distributions and the correlations between the features in the near future You also expect to receive a large volume of prediction requests You plan to use Vertex Al Model Monitoring for drift detection and you want to minimize the cost. What should you do?

A. Use the features for monitoring Set a monitoring- frequency value that is higher than the default. B. Use the features for monitoring Set a prediction-sampling-rare value that is closer to 1 than 0.

C. Use the features and the feature attributions for monitoring. Set a monitoring-frequency value that is lower than the default.

D. Use the features and the feature attributions for monitoring Set a prediction-sampling-rate value that is closer to 0 than 1.

Answer: D

Explanation:

The best option for using Vertex AI Model Monitoring for drift detection and minimizing the cost is to use the features and the feature attributions for monitoring, and set a prediction-sampling-rate value that is closer to 0 than 1. This

www.atmicnetworks.com

option allows you to leverage the power and flexibility of Google Cloud to detect feature drift in the input predict requests for custom models, and reduce the storage and computation costs of the model monitoring job. Vertex AI Model Monitoring is a service that can track and compare the results of multiple machine learning runs. Vertex AI Model Monitoring can monitor the model's prediction input data for feature skew and drift. Feature drift occurs when the feature data distribution in production changes over time. If the original training data is not available, you can enable drift detection to monitor your models for feature drift. Vertex AI Model Monitoring uses TensorFlow Data Validation (TFDV) to calculate the distributions and distance scores for each feature, and compares them with a baseline distribution. The baseline distribution is the statistical distribution of the feature's values in the training data. If the training data is not available, the baseline distribution is calculated from the first 1000 prediction requests that the model receives. If the distance score for a feature exceeds an alerting threshold that you set, Vertex AI Model Monitoring sends you an email alert. However, if you use a custom model, you can also enable feature attribution monitoring, which can provide more insights into the feature drift. Feature attribution monitoring analyzes the feature attributions, which are the contributions of each feature to the prediction output. Feature attribution monitoring can help you identify the features that have the most impact on the model performance, and the features that have the most significant drift over time. Feature attribution monitoring can also help you understand the relationship between the features and the prediction output, and the correlation between the features1. The predictionsampling-rate is a parameter that determines the percentage of prediction requests that are logged and analyzed by the model monitoring job. Using a lower prediction-sampling-rate can reduce the storage and computation costs of the model monitoring job, but also the quality and validity of the data. Using a lower prediction-sampling-rate can introduce sampling bias and noise into the data, and make the model monitoring job miss some important features or patterns of the data. However, using a higher prediction-sampling-rate can increase the storage and computation costs of the model monitoring job, and also the amount of data that needs to be processed and analyzed. Therefore, there is a trade-off between the prediction-sampling-rate and the cost and accuracy of the model monitoring job, and the optimal prediction-sampling-rate depends on the business objective and the data characteristics2. By using the features and the feature attributions for monitoring, and setting a prediction-sampling-rate value that is closer to 0 than 1, you can use Vertex AI Model Monitoring for drift detection and minimize the cost.

The other options are not as good as option D, for the following reasons:

Option A: Using the features for monitoring and setting a monitoring-frequency value that is higher than the default would not enable feature attribution monitoring, and could increase the cost of the model monitoring job. The monitoring-frequency is a parameter that determines how often the model monitoring job analyzes the logged prediction requests and calculates the distributions and distance scores for each feature. Using a higher monitoring-frequency can increase the frequency and timeliness of the model monitoring job, but also the computation costs of the model monitoring job. Moreover, using the features for monitoring would not enable feature attribution monitoring, which can provide more insights into the feature drift and the model performance1.

Option B: Using the features for monitoring and setting a prediction-sampling-rate value that is closer to 1 than 0 would not enable feature attribution monitoring, and could increase the cost of the model monitoring job. The prediction-sampling-rate is a parameter that determines the percentage of prediction requests that are logged and analyzed by the model monitoring job. Using a higher prediction-sampling-rate can increase the quality and validity of the data, but also the storage and computation costs of the model monitoring job. Moreover, using the features for monitoring would not enable feature attribution monitoring, which can provide more insights into the feature drift and the model performance12.

Option C: Using the features and the feature attributions for monitoring and setting a monitoringfrequency value that is lower than the default would enable feature attribution monitoring, but could reduce the frequency and timeliness of the model monitoring job. The monitoring-frequency is a parameter that determines how often the model monitoring job analyzes the logged prediction requests and calculates the distributions and distance scores for each feature. Using a lower monitoring-frequency can reduce the computation costs of the model monitoring job, but also the frequency and timeliness of the model monitoring job. This can make the model monitoring job less responsive and effective in detecting and alerting the feature drift1.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 3: Production ML Systems, Week 4: Evaluation

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 3: Scaling ML models in production, 3.3 Monitoring ML models in production

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 6: Production ML Systems, Section 6.3: Monitoring ML Models Using Model Monitoring
Understanding the score threshold slider

Question: 156

You have recently trained a scikit-learn model that you plan to deploy on Vertex Al. This model will support both online and batch prediction. You need to preprocess input data for model inference.
You want to package the model for deployment while minimizing additional code What should you do?

A. 1 Upload your model to the Vertex Al Model Registry by using a prebuilt scikit-learn prediction container
2 Deploy your model to Vertex Al Endpoints, and create a Vertex Al batch prediction job that uses the instanceConfig.inscanceType setting to transform your input data
B. 1 Wrap your model in a custom prediction routine (CPR). and build a container image from the CPR local model
2 Upload your sci-kit learn model container to Vertex Al Model Registry
3 Deploy your model to Vertex Al Endpoints, and create a Vertex Al batch prediction job
C. 1. Create a custom container for your sci-kit learn model,
2 Define a custom serving function for your model
3 Upload your model and custom container to Vertex Al Model Registry
4 Deploy your model to Vertex Al Endpoints, and create a Vertex Al batch prediction job
D. 1 Create a custom container for your sci-kit learn model.
2 Upload your model and custom container to Vertex Al Model Registry
3 Deploy your model to Vertex Al Endpoints, and create a Vertex Al batch prediction job that uses the instanceConfig. instanceType setting to transform your input data

Answer: B

Explanation:
The best option for deploying a scikit-learn model on Vertex AI with minimal additional code is to wrap the model in a custom prediction routine (CPR) and build a container image from the CPR local model. Upload your scikit-learn model container to Vertex Al Model Registry. Deploy your model to Vertex AI Endpoints, and create a Vertex AI batch prediction job. This option allows you to leverage the power and simplicity of Google Cloud to deploy and serve a scikit-learn model that supports both online and batch prediction. Vertex AI is a unified platform for building and deploying machine learning solutions on Google Cloud. Vertex AI can deploy a trained scikit-learn model to an online prediction endpoint, which can provide low-latency predictions for individual instances. Vertex AI can also create a batch prediction job, which can provide high-throughput predictions for a large batch of instances. A custom prediction routine (CPR) is a Python script that defines the logic for preprocessing the input data, running the prediction, and postprocessing the output data. A CPR can
help you customize the prediction behavior of your model, and handle complex or non-standard data formats. A CPR can also help you minimize the additional code, as you only need to write a few functions to implement the prediction logic. A container image is a package that contains the model, the CPR, and the dependencies. A container image can help you standardize and simplify the deployment process, as you only need to upload the container image to Vertex AI

Model Registry, and deploy it to Vertex AI Endpoints. By wrapping the model in a CPR and building a container image from the CPR local model, uploading the scikit-learn model container to Vertex AI Model Registry, deploying the model to Vertex AI Endpoints, and creating a Vertex AI batch prediction job, you can deploy a scikit-learn model on Vertex AI with minimal additional code1.

The other options are not as good as option B, for the following reasons:

Option A: Uploading your model to the Vertex AI Model Registry by using a prebuilt scikit-learn prediction container, deploying your model to Vertex AI Endpoints, and creating a Vertex AI batch prediction job that uses the instanceConfig.instanceType setting to transform your input data would not allow you to preprocess the input data for model inference, and could cause errors or poor performance. A prebuilt scikit-learn prediction container is a container image that is provided by Google Cloud, and contains the scikit-learn framework and the dependencies. A prebuilt scikit-learn prediction container can help you deploy a scikit-learn model without writing any code, but it also limits your customization options. A prebuilt scikit-learn prediction container can only handle standard data formats, such as JSON or CSV, and cannot perform any preprocessing or postprocessing on the input or output data. If your input data requires any transformation or normalization before running the prediction, you cannot use a prebuilt scikit-learn prediction container. The instanceConfig.instanceType setting is a parameter that determines the machine type and the accelerator type for the batch prediction job. The instanceConfig.instanceType setting can help you optimize the performance and the cost of the batch prediction job, but it cannot help you transform your input data2.

Option C: Creating a custom container for your scikit-learn model, defining a custom serving function for your model, uploading your model and custom container to Vertex AI Model Registry, and deploying your model to Vertex AI Endpoints, and creating a Vertex AI batch prediction job would require more skills and steps than using a CPR and a container image. A custom container is a container image that contains the model, the dependencies, and a web server. A custom container can help you customize the prediction behavior of your model, and handle complex or non-standard data formats. A custom serving function is a Python function that defines the logic for running the prediction on the model. A custom serving function can help you implement the prediction logic of your model, and handle complex or non-standard data formats. However, creating a custom container and defining a custom serving function would require more skills and steps than using a CPR and a container image. You would need to write code, build and test the container image, configure the web server, and implement the prediction logic. Moreover, creating a custom container and defining a custom serving function would not allow you to preprocess the input data for model inference, as the custom serving function only runs the prediction on the model3.

Option D: Creating a custom container for your scikit-learn model, uploading your model and custom container to Vertex AI Model Registry, deploying your model to Vertex AI Endpoints, and creating a Vertex AI batch prediction job that uses the instanceConfig.instanceType setting to transform your input data would not allow you to preprocess the input data for model inference, and could cause errors or poor performance. A custom container is a container image that contains the model, the dependencies, and a web server. A custom container can help you customize the prediction behavior of your model, and handle complex or non-standard data formats. However, creating a custom container would require more skills and steps than using a CPR and a container image. You would need to write code, build and test the container image, and configure the web server. The instanceConfig.instanceType setting is a parameter that determines the machine type and the accelerator type for the batch prediction job. The instanceConfig.instanceType setting can help you optimize the performance and the cost of the batch prediction job, but it cannot help you transform your input data23.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 3: Production ML Systems, Week 2: Serving ML Predictions

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 3: Scaling ML models in production, 3.1 Deploying ML models to production

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 6: Production ML Systems, Section 6.2: Serving ML Predictions

Custom prediction routines

Using pre-built containers for prediction

Using custom containers for prediction

Question: 157

You work for a food product company. Your company's historical sales data is stored in BigQuery You need to use Vertex AI's custom training service to train multiple TensorFlow models that read the data from BigQuery and predict future sales You plan to implement a data preprocessing algorithm that performs min-max scaling and bucketing on a large number of features before you start experimenting with the models. You want to minimize preprocessing time, cost and development effort How should you configure this workflow?

A. Write the transformations into Spark that uses the spark-bigquery-connector and use Dataproc to preprocess the data.
B. Write SQL queries to transform the data in-place in BigQuery.
C. Add the transformations as a preprocessing layer in the TensorFlow models.
D. Create a Dataflow pipeline that uses the BigQueryIO connector to ingest the data process it and write it back to BigQuery.

Answer: C

Explanation:
The best option for configuring the workflow is to add the transformations as a preprocessing layer in the TensorFlow models. This option allows you to leverage the power and simplicity of TensorFlow to preprocess and transform the data with simple Python code. TensorFlow is a framework for building and training machine learning models. TensorFlow provides various tools and libraries for data analysis and machine learning. A preprocessing layer is a type of layer in TensorFlow that can perform data preprocessing and feature engineering operations on the input data. A preprocessing layer can help you customize the data transformation and preprocessing logic, and handle complex or non-standard data formats. A preprocessing layer can also help you minimize the preprocessing time, cost, and development effort, as you only need to write a few lines of code to implement the preprocessing layer, and you do not need to create any intermediate data sources or pipelines. By adding the transformations as a preprocessing layer in the TensorFlow models, you can use Vertex AI's custom training service to train multiple TensorFlow models that read the data from BigQuery and predict future sales1.
The other options are not as good as option C, for the following reasons:
Option A: Writing the transformations into Spark that uses the spark-bigquery-connector and using Dataproc to preprocess the data would require more skills and steps than using a preprocessing layer in TensorFlow. Spark is a framework for distributed data processing and machine learning. Spark can read and write data from BigQuery by using the spark-bigquery-connector, which is a library that allows Spark to communicate with BigQuery. Dataproc is a service that can create and manage Spark clusters on Google Cloud. Dataproc can help you run Spark jobs on Google Cloud, and scale the clusters according to the workload. However, writing the transformations into Spark that uses the spark-bigquery-connector and using Dataproc to preprocess the data would require more skills and steps than using a preprocessing layer in TensorFlow. You would need to write code, create and configure the Spark cluster, install and import the spark-bigquery-connector, load and preprocess the data, and write the data back to BigQuery. Moreover, this option would create an intermediate data source in BigQuery, which can increase the storage and computation costs2.
Option B: Writing SQL queries to transform the data in-place in BigQuery would not allow you to use Vertex AI's custom training service to train multiple TensorFlow models that read the data from BigQuery and predict future sales. BigQuery is a service that can perform data analysis and machine learning by using SQL queries. BigQuery can perform data transformation and preprocessing by using SQL functions and clauses, such as MIN, MAX, CASE, and TRANSFORM. BigQuery can also perform machine learning by using BigQuery ML, which is a feature that can create and train machine

learning models by using SQL queries. However, writing SQL queries to transform the data in-place in BigQuery would not allow you to use Vertex AI's custom training service to train multiple TensorFlow models that read the data from BigQuery and predict future sales. Vertex AI's custom training service is a service that can run your custom machine learning code on Vertex AI. Vertex AI's custom training service can support various machine learning frameworks, such as TensorFlow, PyTorch, and scikit- learn. Vertex AI's custom training service cannot support SQL queries, as SQL is not a machine learning framework. Therefore, if you want to use Vertex AI's custom training service, you cannot use SQL queries to transform the data in-place in BigQuery3.

Option D: Creating a Dataflow pipeline that uses the BigQueryIO connector to ingest the data, process it, and write it back to BigQuery would require more skills and steps than using a preprocessing layer in TensorFlow. Dataflow is a service that can create and run data processing and machine learning pipelines on Google Cloud. Dataflow can read and write data from BigQuery by using the BigQueryIO connector, which is a library that allows Dataflow to communicate with BigQuery. Dataflow can perform data transformation and preprocessing by using Apache Beam, which is a framework for distributed data processing and machine learning. However, creating a Dataflow pipeline that uses the BigQueryIO connector to ingest the data, process it, and write it back to BigQuery would require more skills and steps than using a preprocessing layer in TensorFlow. You would need to write code, create and configure the Dataflow pipeline, install and import the BigQueryIO connector, load and preprocess the data, and write the data back to BigQuery. Moreover, this option would create an intermediate data source in BigQuery, which can increase the storage and computation costs4.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 3: Production ML Systems, Week 2: Serving ML Predictions

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 2: Developing ML models, 2.1 Developing ML models by using TensorFlow

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 4: Developing ML Models, Section 4.1: Developing ML Models by Using TensorFlow TensorFlow Preprocessing Layers

Spark and BigQuery

Dataproc

BigQuery ML

Dataflow and BigQuery

Apache Beam

Question: 158

You have created a Vertex AI pipeline that includes two steps. The first step preprocesses 10 TB data completes in about 1 hour, and saves the result in a Cloud Storage bucket The second step uses the processed data to train a model You need to update the model's code to allow you to test different algorithms You want to reduce pipeline execution time and cost, while also minimizing pipeline changes What should you do?

A. Add a pipeline parameter and an additional pipeline step Depending on the parameter value the pipeline step conducts or skips data preprocessing and starts model training.

B. Create another pipeline without the preprocessing step, and hardcode the preprocessed Cloud Storage file location for model training.

C. Configure a machine with more CPU and RAM from the compute-optimized machine family for the data preprocessing step.

D. Enable caching for the pipeline job. and disable caching for the model training step.

Answer: D

Explanation:

The best option for reducing pipeline execution time and cost, while also minimizing pipeline changes, is to enable caching for the pipeline job, and disable caching for the model training step. This option allows you to leverage the power and simplicity of Vertex AI Pipelines to reuse the output of the data preprocessing step, and avoid unnecessary recomputation. Vertex AI Pipelines is a service that can orchestrate machine learning workflows using Vertex AI. Vertex AI Pipelines can run preprocessing and training steps on custom Docker images, and evaluate, deploy, and monitor the machine learning model. Caching is a feature of Vertex AI Pipelines that can store and reuse the output of a pipeline step, and skip the execution of the step if the input parameters and the code have not changed. Caching can help you reduce the pipeline execution time and cost, as you do not need to re-run the same step with the same input and code. Caching can also help you minimize the pipeline changes, as you do not need to add or remove any pipeline steps or parameters. By enabling caching for the pipeline job, and disabling caching for the model training step, you can create a Vertex AI pipeline that includes two steps. The first step preprocesses 10 TB data, completes in about 1 hour, and saves the result in a Cloud Storage bucket. The second step uses the processed data to train a model. You can update the model's code to allow you to test different algorithms, and run the pipeline job with caching enabled. The pipeline job will reuse the output of the data preprocessing step from the cache, and skip the execution of the step. The pipeline job will run the model training step with the updated code, and disable the caching for the step. This way, you can reduce the pipeline execution time and cost, while also minimizing pipeline changes1.

The other options are not as good as option D, for the following reasons:

Option A: Adding a pipeline parameter and an additional pipeline step, depending on the parameter value, the pipeline step conducts or skips data preprocessing and starts model training, would require more skills and steps than enabling caching for the pipeline job, and disabling caching for the model training step. A pipeline parameter is a variable that can be used to control the input or output of a pipeline step. A pipeline parameter can help you customize the pipeline logic and behavior, and experiment with different values. An additional pipeline step is a new instance of a pipeline component that can perform a part of the pipeline workflow, such as data preprocessing or model training. An additional pipeline step can help you extend the pipeline functionality and complexity, and handle different scenarios. However, adding a pipeline parameter and an additional pipeline step, depending on the parameter value, the pipeline step conducts or skips data preprocessing and starts model training, would require more skills and steps than enabling caching for the pipeline job, and disabling caching for the model training step. You would need to write code, define the pipeline parameter, create the additional pipeline step, implement the conditional logic, and compile and run the pipeline. Moreover, this option would not reuse the output of the data preprocessing step from the cache, but rather from the Cloud Storage bucket, which can increase the data transfer and access costs1.

Option B: Creating another pipeline without the preprocessing step, and hardcoding the preprocessed Cloud Storage file location for model training, would require more skills and steps than enabling caching for the pipeline job, and disabling caching for the model training step. A pipeline without the preprocessing step is a pipeline that only includes the model training step, and uses the preprocessed data from the Cloud Storage bucket as the input. A pipeline without the preprocessing step can help you avoid running the data preprocessing step every time, and reduce the pipeline execution time and cost. However, creating another pipeline without the preprocessing step, and hardcoding the preprocessed Cloud Storage file location for model training, would require more skills and steps than enabling caching for the pipeline job, and disabling caching for the model training step. You would need to write code, create a new pipeline, remove the preprocessing step, hardcode the Cloud Storage file location, and compile and run the pipeline. Moreover, this option would not reuse the output of the data preprocessing step from the cache, but rather from the Cloud Storage bucket, which can increase the data transfer and access costs. Furthermore, this option would create another pipeline, which can increase the maintenance and management costs1.

Option C: Configuring a machine with more CPU and RAM from the compute-optimized machine family for the data preprocessing step, would not reduce the pipeline execution time and cost, while also minimizing pipeline changes, but rather increase the pipeline execution cost and complexity. A machine with more CPU and RAM from the compute-

optimized machine family is a virtual machine that has a high ratio of CPU cores to memory, and can provide high performance and scalability for compute-intensive workloads. A machine with more CPU and RAM from the compute-optimized machine family can help you optimize the data preprocessing step, and reduce the pipeline execution time. However, configuring a machine with more CPU and RAM from the compute-optimized machine family for the data preprocessing step, would not reduce the pipeline execution time and cost, while also minimizing pipeline changes, but rather increase the pipeline execution cost and complexity. You would need to write code, configure the machine type parameters for the data preprocessing step, and compile and run the pipeline. Moreover, this option would increase the pipeline execution cost, as machines with more CPU and RAM from the compute-optimized machine family are more expensive than machines with less CPU and RAM from other machine families. Furthermore, this option would not reuse the output of the data preprocessing step from the cache, but rather re-run the data preprocessing step every time, which can increase the pipeline execution time and cost1.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 3: Production ML Systems, Week 3: MLOps

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 3: Scaling ML models in production, 3.2 Automating ML workflows

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 6: Production ML Systems, Section 6.4: Automating ML Workflows

Vertex AI Pipelines

Caching

Pipeline parameters

Machine types

Question: 159

You work for a bank. You have created a custom model to predict whether a loan application should be flagged for human review. The input features are stored in a BigQuery table. The model is performing well and you plan to deploy it to production. Due to compliance requirements the model must provide explanations for each prediction. You want to add this functionality to your model code with minimal effort and provide explanations that are as accurate as possible What should you do?

A. Create an AutoML tabular model by using the BigQuery data with integrated Vertex Explainable AI.

B. Create a BigQuery ML deep neural network model, and use the ML. EXPLAIN_PREDICT method with the num_integral_steps parameter.

C. Upload the custom model to Vertex AI Model Registry and configure feature-based attribution by using sampled Shapley with input baselines.

D. Update the custom serving container to include sampled Shapley-based explanations in the prediction outputs.

Answer: C

Explanation:

The best option for adding explanations to your model code with minimal effort and providing explanations that are as accurate as possible is to upload the custom model to Vertex AI Model Registry and configure feature-based attribution by using sampled Shapley with input baselines. This option allows you to leverage the power and simplicity of Vertex Explainable AI to generate feature attributions for each prediction, and understand how each feature contributes to the model output. Vertex Explainable AI is a service that can help you understand and interpret predictions made by your machine learning models, natively integrated with a number of Google's products and services. Vertex Explainable AI

can provide feature-based and example-based explanations to provide better understanding of model decision making. Feature-based explanations are explanations that show how much each feature in the input influenced the prediction. Feature-based explanations can help you debug and improve model performance, build confidence in the predictions, and understand when and why things go wrong. Vertex Explainable AI supports various feature attribution methods, such as sampled Shapley, integrated gradients, and XRAI. Sampled Shapley is a feature attribution method that is based on the Shapley value, which is a concept from game theory that measures how much each player in a cooperative game contributes to the total payoff. Sampled Shapley approximates the Shapley value for each feature by sampling different subsets of features, and computing the marginal contribution of each feature to the prediction. Sampled Shapley can provide accurate and consistent feature attributions, but it can also be computationally expensive. To reduce the computation cost, you can use input baselines, which are reference inputs that are used to compare with the actual inputs. Input baselines can help you define the starting point or the default state of the features, and calculate the feature attributions relative to the input baselines. By uploading the custom model to Vertex AI Model Registry and configuring feature-based attribution by using sampled Shapley with input baselines, you can add explanations to your model code with minimal effort and provide explanations that are as accurate as possible1.

The other options are not as good as option C, for the following reasons:

Option A: Creating an AutoML tabular model by using the BigQuery data with integrated Vertex Explainable AI would require more skills and steps than uploading the custom model to Vertex AI Model Registry and configuring feature-based attribution by using sampled Shapley with input baselines. AutoML tabular is a service that can automatically build and train machine learning models for structured or tabular data. AutoML tabular can use BigQuery as the data source, and provide feature-based explanations by using integrated gradients as the feature attribution method. However, creating an AutoML tabular model by using the BigQuery data with integrated Vertex Explainable AI would require more skills and steps than uploading the custom model to Vertex AI Model Registry and configuring feature-based attribution by using sampled Shapley with input baselines. You would need to create a new AutoML tabular model, import the BigQuery data, configure the model settings, train and evaluate the model, and deploy the model. Moreover, this option would not use your existing custom model, which is already performing well, but create a new model, which may not have the same performance or behavior as your custom model2.

Option B: Creating a BigQuery ML deep neural network model, and using the ML.EXPLAIN_PREDICT method with the num_integral_steps parameter would not allow you to deploy the model to production, and could provide less accurate explanations than using sampled Shapley with input baselines. BigQuery ML is a service that can create and train machine learning models by using SQL queries on BigQuery. BigQuery ML can create a deep neural network model, which is a type of machine learning model that consists of multiple layers of neurons, and can learn complex patterns and relationships from the data. BigQuery ML can also provide feature-based explanations by using the ML.EXPLAIN_PREDICT method, which is a SQL function that returns the feature attributions for each prediction. The ML.EXPLAIN_PREDICT method uses integrated gradients as the feature attribution method, which is a method that calculates the average gradient of the prediction output with respect to the feature values along the path from the input baseline to the input. The num_integral_steps parameter is a parameter that determines the number of steps along the path from the input baseline to the input. However, creating a BigQuery ML deep neural network model, and using the ML.EXPLAIN_PREDICT method with the num_integral_steps parameter would not allow you to deploy the model to production, and could provide less accurate explanations than using sampled Shapley with input baselines. BigQuery ML does not support deploying the model to Vertex AI Endpoints, which is a service that can provide low-latency predictions for individual instances. BigQuery ML only supports batch prediction, which is a service that can provide high- throughput predictions for a large batch of instances. Moreover, integrated gradients can provide less accurate and consistent explanations than sampled Shapley, as integrated gradients can be sensitive to the choice of the input baseline and the num_integral_steps parameter3.

Option D: Updating the custom serving container to include sampled Shapley-based explanations in the prediction outputs would require more skills and steps than uploading the custom model to Vertex AI Model Registry and configuring feature-based attribution by using sampled Shapley with input baselines. A custom serving container is a container image that contains the model, the dependencies, and a web server. A custom serving container can help you customize the prediction behavior of your model, and handle complex or non-standard data formats. However,

updating the custom serving container to include sampled Shapley-based explanations in the prediction outputs would require more skills and steps than uploading the custom model to Vertex AI Model Registry and configuring feature-based attribution by using sampled Shapley with input baselines. You would need to write code, implement the sampled Shapley algorithm, build and test the container image, and upload and deploy the container image. Moreover, this option would not leverage the power and simplicity of Vertex Explainable AI, which can provide feature-based explanations natively integrated with Vertex AI services4.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 3: Production ML Systems, Week 4: Evaluation

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 3: Scaling ML models in production, 3.3 Monitoring ML models in production

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 6: Production ML Systems, Section 6.3: Monitoring ML Models

Vertex Explainable AI

AutoML Tables

BigQuery ML

Using custom containers for prediction

Question: 160

You recently used XGBoost to train a model in Python that will be used for online serving Your model prediction service will be called by a backend service implemented in Golang running on a Google Kubemetes Engine (GKE) cluster Your model requires pre and postprocessing steps You need to implement the processing steps so that they run at serving time You want to minimize code changes and infrastructure maintenance and deploy your model into production as quickly as possible. What should you do?

A. Use FastAPI to implement an HTTP server Create a Docker image that runs your HTTP server and deploy it on your organization's GKE cluster.

B. Use FastAPI to implement an HTTP server Create a Docker image that runs your HTTP server Upload the image to Vertex Al Model Registry and deploy it to a Vertex Al endpoint.

C. Use the Predictor interface to implement a custom prediction routine Build the custom contain upload the container to Vertex Al Model Registry, and deploy it to a Vertex Al endpoint.

D. Use the XGBoost prebuilt serving container when importing the trained model into Vertex Al Deploy the model to a Vertex Al endpoint Work with the backend engineers to implement the pre- and postprocessing steps in the Golang backend service.

Answer: C

Explanation:

The best option for implementing the processing steps so that they run at serving time, minimizing code changes and infrastructure maintenance, and deploying the model into production as quickly as possible, is to use the Predictor interface to implement a custom prediction routine. Build the custom container, upload the container to Vertex AI Model Registry, and deploy it to a Vertex AI endpoint. This option allows you to leverage the power and simplicity of Vertex AI to serve your XGBoost model with minimal effort and customization. Vertex AI is a unified platform for building and deploying machine learning solutions on Google Cloud. Vertex AI can deploy a trained XGBoost model to an online prediction endpoint, which can provide low-latency predictions for individual instances. A custom prediction routine (CPR) is a Python script that defines the logic for preprocessing the input data, running the prediction, and

postprocessing the output data. A CPR can help you customize the prediction behavior of your model, and handle complex or non-standard data formats. A CPR can also help you minimize the code changes, as you only need to write a few functions to implement the prediction logic. A Predictor interface is a class that inherits from the base class aiplatform.Predictor, and implements the abstract methods predict() and preprocess(). A Predictor interface can help you create a CPR by defining the preprocessing and prediction logic for your model. A container image is a package that contains the model, the CPR, and the dependencies. A container image can help you standardize and simplify the deployment process, as you only need to upload the container image to Vertex AI Model Registry, and deploy it to Vertex AI Endpoints. By using the Predictor interface to implement a CPR, building the custom container, uploading the container to Vertex AI Model Registry, and deploying it to a Vertex AI endpoint, you can implement the processing steps so that they run at serving time, minimize code changes and infrastructure maintenance, and deploy the model into production as quickly as possible1.

The other options are not as good as option C, for the following reasons:

Option A: Using FastAPI to implement an HTTP server, creating a Docker image that runs your HTTP server, and deploying it on your organization's GKE cluster would require more skills and steps than using the Predictor interface to implement a CPR, building the custom container, uploading the container to Vertex AI Model Registry, and deploying it to a Vertex AI endpoint. FastAPI is a framework for building web applications and APIs in Python. FastAPI can help you implement an HTTP server that can handle prediction requests and responses, and perform data preprocessing and postprocessing. A Docker image is a package that contains the model, the HTTP server, and the dependencies. A Docker image can help you standardize and simplify the deployment process, as you only need to build and run the Docker image. GKE is a service that can create and manage Kubernetes clusters on Google Cloud. GKE can help you deploy and scale your Docker image on Google Cloud, and provide high availability and performance. However, using FastAPI to implement an HTTP server, creating a Docker image that runs your HTTP server, and deploying it on your organization's GKE cluster would require more skills and steps than using the Predictor interface to implement a CPR, building the custom container, uploading the container to Vertex AI Model Registry, and deploying it to a Vertex AI endpoint. You would need to write code, create and configure the HTTP server, build and test the Docker image, create and manage the GKE cluster, and deploy and monitor the Docker image. Moreover, this option would not leverage the power and simplicity of Vertex AI, which can provide online prediction natively integrated with Google Cloud services2.

Option B: Using FastAPI to implement an HTTP server, creating a Docker image that runs your HTTP server, uploading the image to Vertex AI Model Registry, and deploying it to a Vertex AI endpoint would require more skills and steps than using the Predictor interface to implement a CPR, building the custom container, uploading the container to Vertex AI Model Registry, and deploying it to a Vertex AI endpoint. FastAPI is a framework for building web applications and APIs in Python. FastAPI can help you implement an HTTP server that can handle prediction requests and responses, and perform data preprocessing and postprocessing. A Docker image is a package that contains the model, the HTTP server, and the dependencies. A Docker image can help you standardize and simplify the deployment process, as you only need to build and run the Docker image. Vertex AI Model Registry is a service that can store and manage your machine learning models on Google Cloud. Vertex AI Model Registry can help you upload and organize your Docker image, and track the model versions and metadata. Vertex AI Endpoints is a service that can provide online prediction for your machine learning models on Google Cloud. Vertex AI Endpoints can help you deploy your Docker image to an online prediction endpoint, which can provide low-latency predictions for individual instances. However, using FastAPI to implement an HTTP server, creating a Docker image that runs your HTTP server, uploading the image to Vertex AI Model Registry, and deploying it to a Vertex AI endpoint would require more skills and steps than using the Predictor interface to implement a CPR, building the custom container, uploading the container to Vertex AI Model Registry, and deploying it to a Vertex AI endpoint. You would need to write code, create and configure the HTTP server, build and test the Docker image, upload the Docker image to Vertex AI Model Registry, and deploy the Docker image to Vertex AI Endpoints. Moreover, this option would not leverage the power and simplicity of Vertex AI, which can provide online prediction natively integrated with Google Cloud services2.

Option D: Using the XGBoost prebuilt serving container when importing the trained model into Vertex AI, deploying the model to a Vertex AI endpoint, working with the backend engineers to implement the pre- and postprocessing steps in the Golang backend service would not allow you to implement the processing steps so that they run at serving time,

and could increase the code changes and infrastructure maintenance. A XGBoost prebuilt serving container is a container image that is provided by Google Cloud, and contains the XGBoost framework and the dependencies. A XGBoost prebuilt serving container can help you deploy a XGBoost model without writing any code, but it also limits your customization options. A XGBoost prebuilt serving container can only handle standard data formats, such as JSON or CSV, and cannot perform any preprocessing or postprocessing on the input or output data. If your input data requires any transformation or normalization before running the prediction, you cannot use a XGBoost prebuilt serving container. A Golang backend service is a service that is implemented in Golang, a programming language that can be used for web development and system programming. A Golang backend service can help you handle the prediction requests and responses from the frontend, and communicate with the Vertex AI endpoint. However, using the XGBoost prebuilt serving container when importing the trained model into Vertex AI, deploying the model to a Vertex AI endpoint, working with the backend engineers to implement the pre- and postprocessing steps in the Golang backend service would not allow you to implement the processing steps so that they run at serving time, and could increase the code changes and infrastructure maintenance. You would need to write code, import the trained model into Vertex AI, deploy the model to a Vertex AI endpoint, implement the pre- and postprocessing steps in the Golang backend service, and test and monitor the Golang backend service. Moreover, this option would not leverage the power and simplicity of Vertex AI, which can provide online prediction natively integrated with Google Cloud services2.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 3: Production ML Systems, Week 2: Serving ML Predictions

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 3: Scaling ML models in production, 3.1 Deploying ML models to production

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 6: Production ML Systems, Section 6.2: Serving ML Predictions

Custom prediction routines

Using pre-built containers for prediction

Using custom containers for prediction

Question: 161

You recently deployed a pipeline in Vertex Al Pipelines that trains and pushes a model to a Vertex Al endpoint to serve real-time traffic. You need to continue experimenting and iterating on your pipeline to improve model performance. You plan to use Cloud Build for CI/CD You want to quickly and easily deploy new pipelines into production and you want to minimize the chance that the new pipeline implementations will break in production. What should you do?

A. Set up a CI/CD pipeline that builds and tests your source code If the tests are successful use the Google Cloud console to upload the built container to Artifact Registry and upload the compiled pipeline to Vertex Al Pipelines.

B. Set up a CI/CD pipeline that builds your source code and then deploys built artifacts into a preproduction environment Run unit tests in the pre-production environment If the tests are successful deploy the pipeline to production.

C. Set up a CI/CD pipeline that builds and tests your source code and then deploys built artifacts into a pre-production environment. After a successful pipeline run in the pre-production environment deploy the pipeline to production

D. Set up a CI/CD pipeline that builds and tests your source code and then deploys built arrets into a pre-production environment After a successful pipeline run in the pre-production environment, rebuild the source code, and deploy the artifacts to production

Answer: C

Explanation:

The best option for continuing experimenting and iterating on your pipeline to improve model performance, using Cloud Build for CI/CD, and deploying new pipelines into production quickly and easily, is to set up a CI/CD pipeline that builds and tests your source code and then deploys built artifacts into a pre-production environment. After a successful pipeline run in the pre-production environment, deploy the pipeline to production. This option allows you to leverage the power and simplicity of Cloud Build to automate, monitor, and manage your pipeline development and deployment workflow. Cloud Build is a service that can create and run continuous integration and continuous delivery (CI/CD) pipelines on Google Cloud. Cloud Build can build your source code, run unit tests, and deploy built artifacts to various Google Cloud services, such as Vertex AI Pipelines, Vertex AI Endpoints, and Artifact Registry. A CI/CD pipeline is a workflow that can automate the process of building, testing, and deploying software. A CI/CD pipeline can help you improve the quality and reliability of your software, accelerate the development and delivery cycle, and reduce the manual effort and errors. A pre-production environment is an environment that can simulate the production environment, but is isolated from the real users and data. A pre-production environment can help you test and validate your software before deploying it to production, and catch any bugs or issues that may affect the user experience or the system performance. By setting up a CI/CD pipeline that builds and tests your source code and then deploys built artifacts into a pre-production environment, you can ensure that your pipeline code is consistent and error-free, and that your pipeline artifacts are compatible and functional. After a successful pipeline run in the pre-production environment, you can deploy the pipeline to production, which is the environment where your software is accessible and usable by the real users and data. By deploying the pipeline to production after a successful pipeline run in the pre-production environment, you can minimize the chance that the new pipeline implementations will break in production, and ensure that your software meets the user expectations and requirements1.

The other options are not as good as option C, for the following reasons:

Option A: Setting up a CI/CD pipeline that builds and tests your source code, and if the tests are successful, using the Google Cloud console to upload the built container to Artifact Registry and upload the compiled pipeline to Vertex AI Pipelines would not allow you to deploy new pipelines into production quickly and easily, and could increase the manual effort and errors. The Google Cloud console is a web-based user interface that can help you access and manage various Google Cloud services, such as Artifact Registry and Vertex AI Pipelines. Artifact Registry is a service that can store and manage your container images and other artifacts on Google Cloud. Artifact Registry can help you upload and organize your container images, and track the image versions and metadata. Vertex AI Pipelines is a service that can orchestrate machine learning workflows using Vertex AI. Vertex AI Pipelines can run preprocessing and training steps on custom Docker images, and evaluate, deploy, and monitor the machine learning model. However, setting up a CI/CD pipeline that builds and tests your source code, and if the tests are successful, using the Google Cloud console to upload the built container to Artifact Registry and upload the compiled pipeline to Vertex AI Pipelines would not allow you to deploy new pipelines into production quickly and easily, and could increase the manual effort and errors. You would need to write code, create and run the CI/CD pipeline, use the Google Cloud console to upload the built container to Artifact Registry, and use the Google Cloud console to upload the compiled pipeline to Vertex AI Pipelines. Moreover, this option would not use a preproduction environment to test and validate your pipeline before deploying it to production, which could increase the chance that the new pipeline implementations will break in production1.

Option B: Setting up a CI/CD pipeline that builds your source code and then deploys built artifacts into a pre-production environment, running unit tests in the pre-production environment, and if the tests are successful, deploying the pipeline to production would not allow you to test and validate your pipeline before deploying it to production, and could cause errors or poor performance. A unit test is a type of test that can verify the functionality and correctness of a small and isolated unit of code, such as a function or a class. A unit test can help you debug and improve your code quality, and catch any bugs or issues that may affect the code logic or output. However, setting up a CI/CD pipeline that builds your source code and then deploys built artifacts into a pre-production environment, running unit tests in the pre-production environment, and if the tests are successful, deploying the pipeline to production would not allow you

to test and validate your pipeline before deploying it to production, and could cause errors or poor performance. You would need to write code, create and run the CI/CD pipeline, deploy the built artifacts to the pre-production environment, run the unit tests in the pre-production environment, and deploy the pipeline to production. Moreover, this option would not run the pipeline in the pre-production environment, which could prevent you from testing and validating the pipeline functionality and compatibility, and catching any bugs or issues that may affect the pipeline workflow or output1.

Option D: Setting up a CI/CD pipeline that builds and tests your source code and then deploys built artifacts into a pre-production environment, after a successful pipeline run in the pre-production environment, rebuilding the source code, and deploying the artifacts to production would not allow you to deploy new pipelines into production quickly and easily, and could increase the complexity and cost of the pipeline development and deployment. Rebuilding the source code is a process that can recompile and repackage the source code into executable artifacts, such as container images and pipeline files. Rebuilding the source code can help you incorporate any changes or updates that may have occurred in the source code, and ensure that the artifacts are consistent and up-to-date. However, setting up a CI/CD pipeline that builds and tests your source code and then deploys built artifacts into a pre-production environment, after a successful pipeline run in the pre-production environment, rebuilding the source code, and deploying the artifacts to production would not allow you to deploy new pipelines into production quickly and easily, and could increase the complexity and cost of the pipeline development and deployment. You would need to write code, create and run the CI/CD pipeline, deploy the built artifacts to the pre-production environment, run the pipeline in the pre-production environment, rebuild the source code, and deploy the artifacts to production. Moreover, this option would increase the pipeline development and deployment time, as rebuilding the source code can be a time-consuming and resource-intensive process1.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 3: Production ML Systems, Week 3: MLOps

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 3: Scaling ML models in production, 3.2 Automating ML workflows

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 6: Production ML Systems, Section 6.4: Automating ML Workflows

Cloud Build

Vertex AI Pipelines

Artifact Registry

Pre-production environment

Question: 162

You work for a bank with strict data governance requirements. You recently implemented a custom model to detect fraudulent transactions You want your training code to download internal data by using an API endpoint hosted in your projects network You need the data to be accessed in the most secure way, while mitigating the risk of data exfiltration. What should you do?

A. Enable VPC Service Controls for peering's, and add Vertex Al to a service perimeter
B. Create a Cloud Run endpoint as a proxy to the data Use Identity and Access Management (1AM) authentication to secure access to the endpoint from the training job.
C. Configure VPC Peering with Vertex Al and specify the network of the training job
D. Download the data to a Cloud Storage bucket before calling the training job

Answer: A

Explanation:

The best option for accessing internal data in the most secure way, while mitigating the risk of data exfiltration, is to enable VPC Service Controls for peerings, and add Vertex AI to a service perimeter. This option allows you to leverage the power and simplicity of VPC Service Controls to isolate and protect your data and services on Google Cloud. VPC Service Controls is a service that can create a secure perimeter around your Google Cloud resources, such as BigQuery, Cloud Storage, and Vertex AI. VPC Service Controls can help you prevent unauthorized access and data exfiltration from your perimeter, and enforce fine-grained access policies based on context and identity. Peerings are connections that can allow traffic to flow between different networks. Peerings can help you connect your Google Cloud network with other Google Cloud networks or external networks, and enable communication between your resources and services. By enabling VPC Service Controls for peerings, you can allow your training code to download internal data by using an API endpoint hosted in your project's network, and restrict the data transfer to only authorized networks and services. Vertex AI is a unified platform for building and deploying machine learning solutions on Google Cloud. Vertex AI can support various types of models, such as linear regression, logistic regression, k-means clustering, matrix factorization, and deep neural networks. Vertex AI can also provide various tools and services for data analysis, model development, model deployment, model monitoring, and model governance. By adding Vertex AI to a service perimeter, you can isolate and protect your

Vertex AI resources, such as models, endpoints, pipelines, and feature store, and prevent data exfiltration from your perimeter1.

The other options are not as good as option A, for the following reasons:

Option B: Creating a Cloud Run endpoint as a proxy to the data, and using Identity and Access Management (IAM) authentication to secure access to the endpoint from the training job would require more skills and steps than enabling VPC Service Controls for peerings, and adding Vertex AI to a service perimeter. Cloud Run is a service that can run your stateless containers on a fully managed environment or on your own Google Kubernetes Engine cluster. Cloud Run can help you deploy and scale your containerized applications quickly and easily, and pay only for the resources you use. A Cloud Run endpoint is a URL that can expose your containerized application to the internet or to other Google Cloud services. A Cloud Run endpoint can help you access and invoke your application from anywhere, and handle the load balancing and traffic routing. A proxy is a server that can act as an intermediary between a client and a target server. A proxy can help you modify, filter, or redirect the requests and responses between the client and the target server, and provide additional functionality or security. IAM is a service that can manage access control for Google Cloud resources. IAM can help you define who (identity) has what access (role) to which resource, and enforce the access policies. By creating a Cloud Run endpoint as a proxy to the data, and using IAM authentication to secure access to the endpoint from the training job, you can access internal data by using an API endpoint hosted in your project's network, and restrict the data access to only authorized identities and roles. However, creating a Cloud Run endpoint as a proxy to the data, and using IAM authentication to secure access to the endpoint from the training job would require more skills and steps than enabling VPC Service Controls for peerings, and adding Vertex AI to a service perimeter. You would need to write code, create and configure the Cloud Run endpoint, implement the proxy logic, deploy and monitor the Cloud Run endpoint, and set up the IAM policies. Moreover, this option would not prevent data exfiltration from your network, as the Cloud Run endpoint can be accessed from outside your network2.

Option C: Configuring VPC Peering with Vertex AI and specifying the network of the training job would not allow you to access internal data by using an API endpoint hosted in your project's network, and could cause errors or poor performance. VPC Peering is a service that can create a peering connection between two VPC networks. VPC Peering can help you connect your Google Cloud network with another Google Cloud network or an external network, and enable communication between your resources and services. By configuring VPC Peering with Vertex AI and specifying the network of the training job, you can allow your training code to access Vertex AI resources, such as models, endpoints, pipelines, and feature store, and use the same network for the training job. However, configuring VPC Peering with Vertex AI and specifying the network of the training job would not allow you to access internal data by using an API endpoint hosted in your project's network, and could cause errors or poor performance. You would need

to write code, create and configure the VPC Peering connection, and specify the network of the training job. Moreover, this option would not isolate and protect your data and services on Google Cloud, as the VPC Peering connection can expose your network to other networks and services3.

Option D: Downloading the data to a Cloud Storage bucket before calling the training job would not allow you to access internal data by using an API endpoint hosted in your project's network, and could increase the complexity and cost of the data access. Cloud Storage is a service that can store and manage your data on Google Cloud. Cloud Storage can help you upload and organize your data, and track the data versions and metadata. A Cloud Storage bucket is a container that can hold your data on Cloud Storage. A Cloud Storage bucket can help you store and access your data from anywhere, and provide various storage classes and options. By downloading the data to a Cloud Storage bucket before calling the training job, you can access the data from Cloud Storage, and use it as the input for the training job. However, downloading the data to a Cloud Storage bucket before calling the training job would not allow you to access internal data by using an API endpoint hosted in your project's network, and could increase the complexity and cost of the data access. You would need to write code, create and configure the Cloud Storage bucket, download the data to the Cloud Storage bucket, and call the training job. Moreover, this option would create an intermediate data source on Cloud Storage, which can increase the storage and transfer costs, and expose the data to unauthorized access or data exfiltration4.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 3: Production ML Systems, Week 1: Data Engineering

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 1: Framing ML problems, 1.2 Defining data needs

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 2: Data Engineering, Section 2.2: Defining Data Needs

VPC Service Controls

Cloud Run

VPC Peering

Cloud Storage

Question: 163

You are deploying a new version of a model to a production Vertex Al endpoint that is serving traffic You plan to direct all user traffic to the new model You need to deploy the model with minimal disruption to your application What should you do?

A. 1 Create a new endpoint.
2 Create a new model Set it as the default version Upload the model to Vertex Al Model Registry.
3 . Deploy the new model to the new endpoint.
4 Update Cloud DNS to point to the new endpoint

B. 1. Create a new endpoint.
2. Create a new model Set the parentModel parameter to the model ID of the currently deployed model and set it as the default version Upload the model to Vertex Al Model Registry
3. Deploy the new model to the new endpoint and set the new model to 100% of the traffic

C. 1 Create a new model Set the parentModel parameter to the model ID of the currently deployed model Upload the model to Vertex Al Model Registry.
2 Deploy the new model to the existing endpoint and set the new model to 100% of the traffic.

D. 1, Create a new model Set it as the default version Upload the model to Vertex Al Model Registry 2 Deploy the new model to the existing endpoint

Answer: C

Explanation:

The best option for deploying a new version of a model to a production Vertex AI endpoint that is serving traffic, directing all user traffic to the new model, and deploying the model with minimal disruption to your application, is to create a new model, set the parentModel parameter to the model ID of the currently deployed model, upload the model to Vertex AI Model Registry, deploy the new model to the existing endpoint, and set the new model to 100% of the traffic. This option allows

you to leverage the power and simplicity of Vertex AI to update your model version and serve online predictions with low latency. Vertex AI is a unified platform for building and deploying machine learning solutions on Google Cloud. Vertex AI can deploy a trained model to an online prediction endpoint, which can provide low-latency predictions for individual instances. A model is a resource that represents a machine learning model that you can use for prediction. A model can have one or more versions, which are different implementations of the same model. A model version can have different parameters, code, or data than another version of the same model. A model version can help you experiment and iterate on your model, and improve the model performance and accuracy. A parentModel parameter is a parameter that specifies the model ID of the model that the new model version is based on. A parentModel parameter can help you inherit the settings and metadata of the existing model, and avoid duplicating the model configuration. Vertex AI Model Registry is a service that can store and manage your machine learning models on Google Cloud. Vertex AI Model Registry can help you upload and organize your models, and track the model versions and metadata. An endpoint is a resource that provides the service endpoint (URL) you use to request the prediction. An endpoint can have one or more deployed models, which are instances of model versions that are associated with physical resources. A deployed model can help you serve online predictions with low latency, and scale up or down based on the traffic. By creating a new model, setting the parentModel parameter to the model ID of the currently deployed model, uploading the model to Vertex AI Model Registry, deploying the new model to the existing endpoint, and setting the new model to 100% of the traffic, you can deploy a new version of a model to a production Vertex AI endpoint that is serving traffic, direct all user traffic to the new model, and deploy the model with minimal disruption to your application1.

The other options are not as good as option C, for the following reasons:

Option A: Creating a new endpoint, creating a new model, setting it as the default version, uploading the model to Vertex AI Model Registry, deploying the new model to the new endpoint, and updating Cloud DNS to point to the new endpoint would require more skills and steps than creating a new model, setting the parentModel parameter to the model ID of the currently deployed model, uploading the model to Vertex AI Model Registry, deploying the new model to the existing endpoint, and setting the new model to 100% of the traffic. Cloud DNS is a service that can provide reliable and scalable Domain Name System (DNS) services on Google Cloud. Cloud DNS can help you manage your DNS records, and resolve domain names to IP addresses. By updating Cloud DNS to point to the new endpoint, you can redirect the user traffic to the new endpoint, and avoid breaking the existing application. However, creating a new endpoint, creating a new model, setting it as the default version, uploading the model to Vertex AI Model Registry, deploying the new model to the new endpoint, and updating Cloud DNS to point to the new endpoint would require more skills and steps than creating a new model, setting the parentModel parameter to the model ID of the currently deployed model, uploading the model to Vertex AI Model Registry, deploying the new model to the existing endpoint, and setting the new model to 100% of the traffic. You would need to write code, create and configure the new endpoint, create and configure the new model, upload the model to Vertex AI Model Registry, deploy the model to the new endpoint, and update Cloud DNS to point to the new endpoint. Moreover, this option would create a new endpoint, which can increase the maintenance and management costs2.

Option B: Creating a new endpoint, creating a new model, setting the parentModel parameter to the model ID of the currently deployed model and setting it as the default version, uploading the model to Vertex AI Model Registry, and deploying the new model to the new endpoint and setting the new model to 100% of the traffic would require more skills and steps than creating a new model, setting the parentModel parameter to the model ID of the currently deployed model, uploading the model to Vertex AI Model Registry, deploying the new model to the existing endpoint,

and setting the new
model to 100% of the traffic. A parentModel parameter is a parameter that specifies the model ID of the model that the new model version is based on. A parentModel parameter can help you inherit the settings and metadata of the existing model, and avoid duplicating the model configuration. A default version is a model version that is used for prediction when no other version is specified. A default version can help you simplify the prediction request, and avoid specifying the model version every time. By setting the parentModel parameter to the model ID of the currently deployed model and setting it as the default version, you can create a new model that is based on the existing model, and use it for prediction without specifying the model version. However, creating a new endpoint, creating a new model, setting the parentModel parameter to the model ID of the currently deployed model and setting it as the default version, uploading the model to Vertex AI Model Registry, and deploying the new model to the new endpoint and setting the new model to 100% of the traffic would require more skills and steps than creating a new model, setting the parentModel parameter to the model ID of the currently deployed model, uploading the model to Vertex AI Model Registry, deploying the new model to the existing endpoint, and setting the new model to 100% of the traffic. You would need to write code, create and configure the new endpoint, create and configure the new model, upload the model to Vertex AI Model Registry, and deploy the model to the new
endpoint. Moreover, this option would create a new endpoint, which can increase the maintenance and management costs2.

Option D: Creating a new model, setting it as the default version, uploading the model to Vertex AI Model Registry, and deploying the new model to the existing endpoint would not allow you to inherit the settings and metadata of the existing model, and could cause errors or poor performance. A default version is a model version that is used for prediction when no other version is specified. A default version can help you simplify the prediction request, and avoid specifying the model version every time. By setting the new model as the default version, you can use the new model for prediction without specifying the model version. However, creating a new model, setting it as the default version, uploading the model to Vertex AI Model Registry, and deploying the new model to the existing endpoint would not allow you to inherit the settings and metadata of the existing model, and could cause errors or poor performance. You would need to write code, create and configure the new model, upload the model to Vertex AI Model Registry, and deploy the model to the existing endpoint. Moreover, this option would not set the parentModel parameter to the model ID of the currently deployed model, which could prevent you from inheriting the settings and metadata of the existing model, and cause inconsistencies or conflicts between the model versions2.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 3: Production ML Systems, Week 2: Serving ML Predictions

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 3: Scaling ML models in production, 3.1 Deploying ML models to production

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 6: Production ML Systems, Section 6.2: Serving ML Predictions

Vertex AI Cloud DNS

Question: 164

You are training an ML model on a large dataset. You are using a TPU to accelerate the training process You notice that the training process is taking longer than expected. You discover that the TPU is not reaching its full capacity. What should you do?

A. Increase the learning rate
B. Increase the number of epochs
C. Decrease the learning rate
D. Increase the batch size

www.atmicnetworks.com

Answer: D

Explanation:

The best option for training an ML model on a large dataset, using a TPU to accelerate the training process, and discovering that the TPU is not reaching its full capacity, is to increase the batch size. This option allows you to leverage the power and simplicity of TPUs to train your model faster and more efficiently. A TPU is a custom-developed application-specific integrated circuit (ASIC) that can accelerate machine learning workloads. A TPU can provide high performance and scalability for various types of models, such as linear regression, logistic regression, k-means clustering, matrix factorization, and deep neural networks. A TPU can also support various tools and frameworks, such as TensorFlow, PyTorch, and JAX. A batch size is a parameter that specifies the number of training examples in one forward/backward pass. A batch size can affect the speed and accuracy of the training process. A larger batch size can help you utilize the parallel processing power of the TPU, and reduce the communication overhead between the TPU and the host CPU. A larger batch size can also help you avoid overfitting, as it can reduce the variance of the gradient updates. By increasing the batch size, you can train your model on a large dataset faster and more efficiently, and make full use of the TPU capacity1.

The other options are not as good as option D, for the following reasons:

Option A: Increasing the learning rate would not help you utilize the parallel processing power of the TPU, and could cause errors or poor performance. A learning rate is a parameter that controls how much the model is updated in each iteration. A learning rate can affect the speed and accuracy of the training process. A larger learning rate can help you converge faster, but it can also cause instability, divergence, or oscillation. By increasing the learning rate, you may not be able to find the optimal solution, and your model may perform poorly on the validation or test data2.

Option B: Increasing the number of epochs would not help you utilize the parallel processing power of the TPU, and could increase the complexity and cost of the training process. An epoch is a measure of the number of times all of the training examples are used once in the training process. An epoch can affect the speed and accuracy of the training process. A larger number of epochs can help you learn more from the data, but it can also cause overfitting, underfitting, or diminishing returns. By increasing the number of epochs, you may not be able to improve the model performance significantly, and your training process may take longer and consume more resources3.

Option C: Decreasing the learning rate would not help you utilize the parallel processing power of the TPU, and could slow down the training process. A learning rate is a parameter that controls how much the model is updated in each iteration. A learning rate can affect the speed and accuracy of the training process. A smaller learning rate can help you find a more precise solution, but it can also cause slow convergence or local minima. By decreasing the learning rate, you may not be able to reach the optimal solution in a reasonable time, and your training process may take longer2.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 2: ML Models and Architectures, Week 1: Introduction to ML Models and Architectures

Google Cloud Professional Machine Learning Engineer Exam Guide, Section 2: Architecting ML solutions, 2.1 Designing ML models

Official Google Cloud Certified Professional Machine Learning Engineer Study Guide, Chapter 4: ML Models and Architectures, Section 4.1: Designing ML Models

Use TPUs

Triose phosphate utilization and beyond: from photosynthesis to end ...

Cloud TPU performance guide

Google TPU: Architecture and Performance Best Practices - Run

Question: 165

You work for a retail company. You have a managed tabular dataset in Vertex AI that contains sales data from three different stores. The dataset includes several features such as store name and sale timestamp. You want to use the data to train a model that makes sales predictions for a new store that will open soon You need to split the data between the

training, validation, and test sets What approach should you use to split the data?

A. Use Vertex AI manual split, using the store name feature to assign one store for each set.
B. Use Vertex AI default data split.
C. Use Vertex AI chronological split and specify the sales timestamp feature as the time vanable.
D. Use Vertex AI random split assigning 70% of the rows to the training set, 10% to the validation set, and 20% to the test set.

Answer: B

Explanation:
The best option for splitting the data between the training, validation, and test sets, using a managed tabular dataset in Vertex AI that contains sales data from three different stores, is to use Vertex AI default data split. This option allows you to leverage the power and simplicity of Vertex AI to automatically and randomly split your data into the three sets by percentage. Vertex AI is a unified platform for building and deploying machine learning solutions on Google Cloud. Vertex AI can support various types of models, such as linear regression, logistic regression, k-means clustering, matrix factorization, and deep neural networks. Vertex AI can also provide various tools and services for data analysis, model development, model deployment, model monitoring, and model governance. A default data split is a data split method that is provided by Vertex AI, and does not require any user input or configuration. A default data split can help you split your data into the training, validation, and test sets by using a random sampling method, and assign a fixed percentage of the data to each set. A default data split can help you simplify the data split process, and works well in most cases. A training set is a subset of the data that is used to train the model, and adjust the model parameters. A training set can help you learn the relationship between the input features and the target variable, and optimize the model performance. A validation set is a subset of the data that is used to validate the model, and tune the model hyperparameters. A validation set can help you evaluate the model performance on unseen data, and avoid overfitting or underfitting. A test set is a subset of the data that is used to test the model, and provide the final evaluation metrics. A test set can help you assess the model performance on new data, and measure the generalization ability of the model. By using Vertex AI default data split, you can split your data into the training, validation, and test sets by using a random sampling method, and assign the following percentages of the data to each set1:

Set	Text	Image	Video
Training	80%	80%	80%
Validation	10%	10%	N/A
Test	10%	10%	20%

The other options are not as good as option B, for the following reasons:
Option A: Using Vertex AI manual split, using the store name feature to assign one store for each set would not allow you to split your data into representative and balanced sets, and could cause errors or poor performance. A manual split is a data split method that allows you to control how your data is split into sets, by using the ml_use label or the data filter expression. A manual split can help you customize the data split logic, and handle complex or non-standard data formats. A store name feature is a feature that indicates the name of the store where the sales data was collected. A store name feature can help you identify the source of the data, and group the data by store. However, using Vertex AI manual split, using the store name feature to assign one store for each set would not allow you to split your data into representative and balanced sets, and could cause errors or poor performance. You would need to write code, create and configure the ml_use label or the data filter expression, and assign one store for each set. Moreover, this option would not ensure that the data in each set has the same distribution and characteristics as the data in the whole dataset, which could prevent you from learning the general pattern of the data, and cause bias or variance in the model2.

Option C: Using Vertex AI chronological split and specifying the sales timestamp feature as the time variable would not allow you to split your data into representative and balanced sets, and could cause errors or poor performance. A chronological split is a data split method that allows you to split your data into sets based on the order of the data. A chronological split can help you preserve the temporal dependency and sequence of the data, and avoid data leakage. A sales timestamp feature is a feature that indicates the date and time when the sales data was collected. A sales timestamp feature can help you track the changes and trends of the data over time, and capture the seasonality and cyclicality of the data. However, using Vertex AI chronological split and specifying the sales timestamp feature as the time variable would not allow you to split your data into representative and balanced sets, and could cause errors or poor performance. You would need to write code, create and configure the time variable, and split the data by the order of the time variable. Moreover, this option would not ensure that the data in each set has the same distribution and characteristics as the data in the whole dataset, which could prevent you from learning the general pattern of the data, and cause bias or variance in the model3.

Option D: Using Vertex AI random split, assigning 70% of the rows to the training set, 10% to the validation set, and 20% to the test set would not allow you to use the default data split method that is provided by Vertex AI, and could increase the complexity and cost of the data split process. A random split is a data split method that allows you to split your data into sets by using a random sampling method, and assign a custom percentage of the data to each set. A random split can help you split your data into representative and balanced sets, and avoid data leakage. However, using Vertex AI random split, assigning 70% of the rows to the training set, 10% to the validation set, and 20% to the test set would not allow you to use the default data split method that is provided by Vertex AI, and could increase the complexity and cost of the data split process. You would need to write code, create and configure the random split method, and assign the custom percentages to each set. Moreover, this option would not use the default data split method that is provided by Vertex AI, which can simplify the data split process, and works well in most cases1.

Reference:

About data splits for AutoML models | Vertex AI | Google Cloud

Manual split for unstructured data

Mathematical split

Question: 166

You have developed a BigQuery ML model that predicts customer churn and deployed the model to Vertex Al Endpoints. You want to automate the retraining of your model by using minimal additional code when model feature values change. You also want to minimize the number of times that your model is retrained to reduce training costs. What should you do?

A. 1. Enable request-response logging on Vertex Al Endpoints.
   2 Schedule a TensorFlow Data Validation job to monitor prediction drift
   3 . Execute model retraining if there is significant distance between the distributions.

B. 1. Enable request-response logging on Vertex Al Endpoints
   2. Schedule a TensorFlow Data Validation job to monitor training/serving skew
   3. Execute model retraining if there is significant distance between the distributions

C. 1 Create a Vertex Al Model Monitoring job configured to monitor prediction drift.
   2. Configure alert monitoring to publish a message to a Pub/Sub queue when a monitonng alert is detected.
   3. Use a Cloud Function to monitor the Pub/Sub queue, and trigger retraining in BigQuery

D. 1. Create a Vertex Al Model Monitoring job configured to monitor training/serving skew
   2. Configure alert monitoring to publish a message to a Pub/Sub queue when a monitoring alert is detected
   3. Use a Cloud Function to monitor the Pub/Sub queue, and trigger retraining in BigQuery.

Answer: C

Explanation:

The best option for automating the retraining of your model by using minimal additional code when model feature values change, and minimizing the number of times that your model is retrained to reduce training costs, is to create a Vertex AI Model Monitoring job configured to monitor prediction drift, configure alert monitoring to publish a message to a Pub/Sub queue when a monitoring alert is detected, and use a Cloud Function to monitor the Pub/Sub queue, and trigger retraining in BigQuery. This option allows you to leverage the power and simplicity of Vertex AI, Pub/Sub, and Cloud Functions to monitor your model performance and retrain your model when needed. Vertex AI is a unified platform for building and deploying machine learning solutions on Google Cloud. Vertex AI can deploy a trained model to an online prediction endpoint, which can provide low-latency predictions for individual instances. Vertex AI can also provide various tools and services for data analysis, model development, model deployment, model monitoring, and model governance. A Vertex AI Model Monitoring job is a resource that can monitor the performance and quality of your deployed models on Vertex AI. A Vertex AI Model Monitoring job can help you detect and diagnose issues with your models, such as data drift, prediction drift, training/serving skew, or model staleness. Prediction drift is a type of model monitoring metric that measures the difference between the distributions of the predictions generated by the model on the training data and the predictions generated by the model on the online data. Prediction drift can indicate that the model

performance is degrading, or that the online data is changing over time. By creating a Vertex AI Model Monitoring job configured to monitor prediction drift, you can track the changes in the model predictions, and compare them with the expected predictions. Alert monitoring is a feature of Vertex AI Model Monitoring that can notify you when a monitoring metric exceeds a predefined threshold. Alert monitoring can help you set up rules and conditions for triggering alerts, and choose the notification channel for receiving alerts. Pub/Sub is a service that can provide reliable and scalable messaging and event streaming on Google Cloud. Pub/Sub can help you publish and subscribe to messages, and deliver them to various Google Cloud services, such as Cloud Functions. A Pub/Sub queue is a resource that can hold messages that are published to a Pub/Sub topic. A Pub/Sub queue can help you store and manage messages, and ensure that they are delivered to the subscribers. By configuring alert monitoring to publish a message to a Pub/Sub queue when a monitoring alert is detected, you can send a notification to a Pub/Sub topic, and trigger a downstream action based on the alert. Cloud Functions is a service that can run your stateless code in response to events on Google Cloud. Cloud Functions can help you create and execute functions without provisioning or managing servers, and pay only for the resources you use. A Cloud Function is a resource that can execute a piece of code in response to an event, such as a Pub/Sub message. A Cloud Function can help you perform various tasks, such as data processing, data transformation, or data analysis. BigQuery is a service that can store and query large-scale data on Google Cloud. BigQuery can help you analyze your data by using SQL queries, and perform various tasks, such as data exploration, data transformation, or data visualization. BigQuery ML is a feature of BigQuery that can create and execute machine learning models in BigQuery by using SQL queries. BigQuery ML can help you build and train various types of models, such as linear regression, logistic regression, k-means clustering, matrix factorization, and deep neural networks. By using a Cloud Function to monitor the Pub/Sub queue, and trigger retraining in BigQuery, you can automate the retraining of your model by using minimal additional code when model feature values change. You can write a Cloud Function that listens to the Pub/Sub queue, and executes a SQL query to retrain your model in BigQuery ML when a prediction drift alert is received. By retraining your model in BigQuery ML, you can update your model parameters and improve your model performance and accuracy1.

The other options are not as good as option C, for the following reasons:

Option A: Enabling request-response logging on Vertex AI Endpoints, scheduling a TensorFlow Data Validation job to monitor prediction drift, and executing model retraining if there is significant distance between the distributions would require more skills and steps than creating a Vertex AI Model Monitoring job configured to monitor prediction drift, configuring alert monitoring to publish a message to a Pub/Sub queue when a monitoring alert is detected, and using a Cloud Function to monitor the Pub/Sub queue, and trigger retraining in BigQuery. Request-response logging is a feature of Vertex AI Endpoints that can record the requests and responses that are sent to and from the online prediction

endpoint. Request-response logging can help you collect and analyze the online prediction data, and troubleshoot any issues with your model. TensorFlow Data Validation is a tool that can analyze and validate your data for machine learning. TensorFlow Data Validation can help you explore, understand, and clean your data, and detect various data issues, such as data drift, data skew, or data anomalies. Prediction drift is a type of data issue that measures the difference between the distributions of the predictions generated by the model on the training data and the predictions generated by the model on the online data. Prediction drift can indicate that the model performance is degrading, or that the online data is changing over time. By enabling request-response logging on Vertex AI Endpoints, and scheduling a TensorFlow Data Validation job to monitor prediction drift, you can collect and analyze the online prediction data, and compare the distributions of the predictions. However, enabling request-response logging on Vertex AI Endpoints, scheduling a TensorFlow Data Validation job to monitor prediction drift, and executing model retraining if there is significant

distance between the distributions would require more skills and steps than creating a Vertex AI Model Monitoring job configured to monitor prediction drift, configuring alert monitoring to publish a message to a Pub/Sub queue when a monitoring alert is detected, and using a Cloud Function to monitor the Pub/Sub queue, and trigger retraining in BigQuery. You would need to write code, enable and configure the request-response logging, create and run the TensorFlow Data Validation job, define and measure the distance between the distributions, and execute the model retraining. Moreover, this option would not automate the retraining of your model, as you would need to manually check the prediction drift and trigger the retraining2.

Option B: Enabling request-response logging on Vertex AI Endpoints, scheduling a TensorFlow Data Validation job to monitor training/serving skew, and executing model retraining if there is significant distance between the distributions would not help you monitor the changes in the model feature values, and could cause errors or poor performance. Training/serving skew is a type of data issue that measures the difference between the distributions of the features used to train the model and the features used to serve the model. Training/serving skew can indicate that the model is not trained on the representative data, or that the data is changing over time. By enabling request-response logging on Vertex AI Endpoints, and scheduling a TensorFlow Data Validation job to monitor training/serving skew, you can collect and analyze the online prediction data, and compare the distributions of the features. However, enabling request-response logging on Vertex AI Endpoints, scheduling a TensorFlow Data Validation job to monitor training/serving skew, and executing model retraining if there is significant distance between the distributions would not help you monitor the changes in the model feature values, and could cause errors or poor performance. You would need to write code, enable and configure the request-response logging, create and run the TensorFlow Data Validation job, define and measure the distance between the distributions, and execute the model retraining. Moreover, this option would not monitor the prediction drift, which is a more direct and relevant metric for measuring the model performance and quality2.

Option D: Creating a Vertex AI Model Monitoring job configured to monitor training/serving skew, configuring alert monitoring to publish a message to a Pub/Sub queue when a monitoring alert is detected, and using a Cloud Function to monitor the Pub/Sub queue, and trigger retraining in BigQuery would not help you monitor the changes in the model feature values, and could cause errors or poor performance. Training/serving skew is a type of data issue that measures the difference between the distributions of the features used to train the model and the features used to serve the model. Training/serving skew can indicate that the model is not trained on the representative data, or that the data is changing over time. By creating a Vertex AI Model Monitoring job configured to monitor training/serving skew, you can track the changes in the model features, and compare them with the expected features. However, creating a Vertex AI Model Monitoring job configured to monitor training/serving skew, configuring alert monitoring to publish a message to a Pub/Sub queue when a monitoring alert is detected, and using a Cloud Function to monitor the Pub/Sub queue, and trigger retraining in BigQuery would not help you monitor the changes in the model feature values, and could cause errors or poor performance. You would need to write code, create and configure the Vertex AI Model Monitoring job, configure the alert monitoring, create and configure the Pub/Sub queue, and write a Cloud Function to trigger the retraining. Moreover, this option would not monitor the prediction drift, which is a more direct and relevant metric for measuring the model performance and quality1.

Reference:

Preparing for Google Cloud Certification: Machine Learning Engineer, Course 3: Production ML Systems, Week 4: ML Governance
Google Cloud Professional Machine Learning Engineer Exam Guide, Section 3: Scaling ML models in production

Question: 167

You have been tasked with deploying prototype code to production. The feature engineering code is in PySpark and runs on Dataproc Serverless. The model training is executed by using a Vertex Al custom training job. The two steps are not connected, and the model training must currently be run manually after the feature engineering step finishes. You need to create a scalable and maintainable production process that runs end-to-end and tracks the connections between steps. What should you do?

A. Create a Vertex Al Workbench notebook Use the notebook to submit the Dataproc Serverless feature engineering job Use the same notebook to submit the custom model training job Run the notebook cells sequentially to tie the steps together end-to-end
B. Create a Vertex Al Workbench notebook Initiate an Apache Spark context in the notebook, and run the PySpark feature engineering code Use the same notebook to run the custom model training job in TensorFlow Run the notebook cells sequentially to tie the steps together end-to-end C. Use the Kubeflow pipelines SDK to write code that specifies two components - The first is a Dataproc Serverless component that launches the feature engineering job - The second is a custom component wrapped in the creare_cusrora_rraining_job_from_ccraponent Utility that launches the custom model training job.
D. Create a Vertex Al Pipelines job to link and run both components Use the Kubeflow pipelines SDK to write code that specifies two components
- The first component initiates an Apache Spark context that runs the PySpark feature engineering code
- The second component runs the TensorFlow custom model training code Create a Vertex Al Pipelines job to link and run both components

Answer: C

Explanation:
The best option for creating a scalable and maintainable production process that runs end-to-end and tracks the connections between steps, using prototype code to production, feature engineering code in PySpark that runs on Dataproc Serverless, and model training that is executed by using a Vertex AI custom training job, is to use the Kubeflow pipelines SDK to write code that specifies two components. The first is a Dataproc Serverless component that launches the feature engineering job. The second is a custom component wrapped in the create_custom_training_job_from_component utility that launches the custom model training job. This option allows you to leverage the power and simplicity of Kubeflow pipelines to orchestrate and automate your machine learning workflows on Vertex AI. Kubeflow pipelines is a platform that can build, deploy, and manage machine learning pipelines on Kubernetes. Kubeflow pipelines can help you create reusable and scalable pipelines, experiment with different pipeline versions and parameters, and monitor and debug your pipelines. Kubeflow pipelines SDK is a set of Python packages that can help you build and run Kubeflow pipelines. Kubeflow pipelines SDK can help you define pipeline components, specify pipeline parameters and inputs, and create pipeline steps and tasks. A component is a self-contained set of code that performs one step in a pipeline, such as data preprocessing, model training, or model evaluation. A component can be created from a Python function, a container image, or a prebuilt component. A custom component is a component that is not provided by Kubeflow pipelines, but is created by the user to perform a specific task. A custom component can be wrapped in a utility function that can help you create a Vertex AI custom training job from the component. A custom training job is a resource that can run your custom training code on Vertex AI. A custom training job can help you train various types of models, such as linear regression, logistic regression,

k-means clustering, matrix factorization, and deep neural networks. By using the Kubeflow pipelines SDK to write code that specifies two components, the first is a Dataproc Serverless component that launches the feature engineering job, and the second is a custom component wrapped in the create_custom_training_job_from_component utility that launches the custom model training job, you can create a scalable and maintainable production process that runs end-to-end and tracks the connections between steps. You can write code that defines the two components, their inputs and outputs, and their dependencies. You can then use the Kubeflow pipelines SDK to create a pipeline that runs the two components in sequence, and submit the pipeline to Vertex AI Pipelines for execution. By using Dataproc Serverless component, you can run your PySpark feature engineering code on Dataproc Serverless, which is a service that can run Spark batch workloads without provisioning and managing your own cluster. By using custom component wrapped in the create_custom_training_job_from_component utility, you can run your custom model training code on Vertex AI, which is a unified platform for building and deploying machine learning solutions on Google Cloud1.

The other options are not as good as option C, for the following reasons:

Option A: Creating a Vertex AI Workbench notebook, using the notebook to submit the Dataproc Serverless feature engineering job, using the same notebook to submit the custom model training job, and running the notebook cells sequentially to tie the steps together end-to-end would require more skills and steps than using the Kubeflow pipelines SDK to write code that specifies two components, the first is a Dataproc Serverless component that launches the feature engineering job, and the second is a custom component wrapped in the create_custom_training_job_from_component utility that launches the custom model training job. Vertex AI Workbench is a service that can provide managed notebooks for machine learning development and experimentation. Vertex AI Workbench can help you create and run JupyterLab notebooks, and access various tools and frameworks, such as TensorFlow, PyTorch, and JAX. By creating a Vertex AI Workbench notebook, using the notebook to submit the Dataproc Serverless feature engineering job, using the same notebook to submit the custom model training job, and running the notebook cells sequentially to tie the steps together end-to-end, you can create a production process that runs end-to-end and tracks the connections between steps. You can write code that submits the Dataproc Serverless feature engineering job and the custom model training job to Vertex AI, and run the code in the notebook cells. However, creating a Vertex AI Workbench notebook, using the notebook to submit the Dataproc Serverless feature engineering job, using the same notebook to submit the custom model training job, and running the notebook cells sequentially to tie the steps together end-to-end would require more skills and steps than using the Kubeflow pipelines SDK to write code that specifies two components, the first is a Dataproc Serverless component that launches the feature engineering job, and the second is a custom component wrapped in the create_custom_training_job_from_component utility that launches the custom model training job. You would need to write code, create and configure the Vertex AI Workbench notebook, submit the Dataproc Serverless feature engineering job and the custom model training job, and run the notebook cells. Moreover, this option would not use the Kubeflow pipelines SDK, which can simplify the pipeline creation and execution process, and provide various features, such as pipeline parameters, pipeline metrics, and pipeline visualization2.

Option B: Creating a Vertex AI Workbench notebook, initiating an Apache Spark context in the notebook, and running the PySpark feature engineering code, using the same notebook to run the custom model training job in TensorFlow, and running the notebook cells sequentially to tie the steps together end-to-end would not allow you to use Dataproc Serverless to run the feature engineering job, and could increase the complexity and cost of the production process. Apache Spark is a framework that can perform large-scale data processing and machine learning. Apache Spark can help you run various tasks, such as data ingestion, data transformation, data analysis, and data visualization. PySpark is a Python API for Apache Spark. PySpark can help you write and run Spark code in Python. An Apache Spark context is a resource that can initialize and configure the Spark environment. An Apache Spark context can help you create and manage Spark objects, such as SparkSession, SparkConf, and SparkContext. By creating a Vertex AI Workbench notebook, initiating an Apache Spark context in the notebook, and running the PySpark feature engineering code, using the same notebook to run the custom model training job in TensorFlow, and running the notebook cells sequentially to tie the steps together end-to-end, you can create a production process that runs end-to-end and tracks the connections between steps. You can write code that initiates an Apache Spark context and runs the PySpark feature engineering code, and runs the custom model training job in TensorFlow, and run the code in the

notebook cells. However, creating a Vertex AI Workbench notebook, initiating an Apache Spark context in the notebook, and running the PySpark feature engineering code, using the same notebook to run the custom model training job in TensorFlow, and running the notebook cells sequentially to tie the steps together end-to-end would not allow you to use Dataproc Serverless to run the feature engineering job, and could increase the complexity and cost of the production process. You would need to write code, create and configure the Vertex AI Workbench notebook, initiate and configure the Apache Spark context, run the PySpark feature engineering code, and run the custom model training job in TensorFlow. Moreover, this option would not use Dataproc Serverless, which is a service that can run Spark batch workloads without provisioning and managing your own cluster, and provide various benefits, such as autoscaling, dynamic resource allocation, and serverless billing2.

Option D: Creating a Vertex AI Pipelines job to link and run both components, using the Kubeflow pipelines SDK to write code that specifies two components, the first component initiates an Apache Spark context that runs the PySpark feature engineering code, and the second component runs the TensorFlow custom model training code, would not allow you to use Dataproc Serverless to run the feature engineering job, and could increase the complexity and cost of the production process.

Vertex AI Pipelines is a service that can run Kubeflow pipelines on Vertex AI. Vertex AI Pipelines can help you create and manage machine learning pipelines, and integrate with various Vertex AI services, such as Vertex AI Workbench, Vertex AI Training, and Vertex AI Prediction. A Vertex AI Pipelines job is a resource that can execute a pipeline on Vertex AI Pipelines. A Vertex AI Pipelines job can help you run your pipeline steps and tasks, and monitor and debug your pipeline execution. By creating a Vertex AI Pipelines job to link and run both components, using the Kubeflow pipelines SDK to write code that specifies two components, the first component initiates an Apache Spark context that runs the PySpark feature engineering code, and the second component runs the TensorFlow custom model training code, you can create a scalable and maintainable production process that runs end-to-end and tracks the connections between steps. You can write code that defines the two components, their inputs and outputs, and their dependencies. You can then use the Kubeflow pipelines SDK to create a pipeline that runs the two components in sequence, and submit the pipeline to Vertex AI Pipelines for execution. However, creating a Vertex AI Pipelines job to link and run both components, using the Kubeflow pipelines SDK to write code that specifies two components, the first component initiates an Apache Spark context that runs the PySpark feature engineering code,

Question: 168

You recently deployed a scikit-learn model to a Vertex Al endpoint You are now testing the model on live production traffic While monitoring the endpoint. you discover twice as many requests per hour than expected throughout the day You want the endpoint to efficiently scale when the demand increases in the future to prevent users from experiencing high latency What should you do?

A. Deploy two models to the same endpoint and distribute requests among them evenly.
B. Configure an appropriate minReplicaCount value based on expected baseline traffic.
C. Set the target utilization percentage in the autcscalir.gMetricspecs configuration to a higher value D. Change the model's machine type to one that utilizes GPUs.

Answer: B

Explanation:

The best option for scaling a Vertex AI endpoint efficiently when the demand increases in the future, using a scikit-learn model that is deployed to a Vertex AI endpoint and tested on live production traffic, is to configure an appropriate minReplicaCount value based on expected baseline traffic. This option allows you to leverage the power and simplicity of Vertex AI to automatically scale your endpoint resources according to the traffic patterns. Vertex AI is a unified platform for building and deploying machine learning solutions on Google Cloud. Vertex AI can deploy a trained model to an online prediction endpoint, which can provide low-latency predictions for individual instances. Vertex AI can also provide various tools and services for data analysis, model development, model deployment, model monitoring, and

model governance. A minReplicaCount value is a parameter that specifies the minimum number of replicas that the endpoint must always have, regardless of the load. A minReplicaCount value can help you ensure that the endpoint has enough resources to handle the expected baseline traffic, and avoid high latency or errors. By configuring an appropriate minReplicaCount value based on expected baseline traffic, you can scale your endpoint efficiently when the demand increases in the future. You can set the minReplicaCount value when you deploy the model to the endpoint, or update it later. Vertex AI will automatically scale up or down the number of replicas within the range of the minReplicaCount and maxReplicaCount values, based on the target utilization percentage and the autoscaling metric1.

The other options are not as good as option B, for the following reasons:

Option A: Deploying two models to the same endpoint and distributing requests among them evenly would not allow you to scale your endpoint efficiently when the demand increases in the future, and could increase the complexity and cost of the deployment process. A model is a resource that represents a machine learning model that you can use for prediction. A model can have one or more versions, which are different implementations of the same model. A model version can help you experiment and iterate on your model, and improve the model performance and accuracy. An endpoint is a resource that provides the service endpoint (URL) you use to request the prediction. An endpoint can have one or more deployed models, which are instances of model versions that are associated with physical resources. A deployed model can help you serve online predictions with low latency, and scale up or down based on the traffic. By deploying two models to the same endpoint and distributing requests among them evenly, you can create a load balancing mechanism that can distribute the traffic across the models, and reduce the load on each model. However, deploying two models to the same endpoint and distributing requests among them evenly would not allow you to scale your endpoint efficiently when the demand increases in the future, and could increase the complexity and cost of the deployment process. You would need to write code, create and configure the two models, deploy the models to the same endpoint, and distribute the requests among them evenly. Moreover, this option would not use the autoscaling feature of Vertex AI, which can automatically adjust the number of replicas based on the traffic patterns, and provide various benefits, such as optimal resource utilization, cost savings, and performance improvement2.

Option C: Setting the target utilization percentage in the autoscalingMetricSpecs configuration to a higher value would not allow you to scale your endpoint efficiently when the demand increases in the future, and could cause errors or poor performance. A target utilization percentage is a parameter that specifies the desired utilization level of each replica. A target utilization percentage can affect the speed and accuracy of the autoscaling process. A higher target utilization percentage can help you reduce the number of replicas, but it can also cause high latency, low throughput, or resource exhaustion. By setting the target utilization percentage in the autoscalingMetricSpecs configuration to a higher value, you can increase the utilization level of each replica, and save some resources. However, setting the target utilization percentage in the autoscalingMetricSpecs configuration to a higher value would not allow you to scale your endpoint efficiently when the demand increases in the future, and could cause errors or poor performance. You would need to write code, create and configure the autoscalingMetricSpecs, and set the target utilization percentage to a higher value. Moreover, this option would not ensure that the endpoint has enough resources to handle the expected baseline traffic, which could cause high latency or errors1.

Option D: Changing the model's machine type to one that utilizes GPUs would not allow you to scale your endpoint efficiently when the demand increases in the future, and could increase the complexity and cost of the deployment process. A machine type is a parameter that specifies the type of virtual machine that the prediction service uses for the deployed model. A machine type can affect the speed and accuracy of the prediction process. A machine type that utilizes GPUs can help you accelerate the computation and processing of the prediction, and handle more prediction requests at the same time. By changing the model's machine type to one that utilizes GPUs, you can improve the prediction performance and efficiency of your model. However, changing the model's machine type to one that utilizes GPUs would not allow you to scale your endpoint efficiently when the demand increases in the future, and could increase the complexity and cost of the deployment process. You would need to write code, create and configure the model, deploy the model to the endpoint, and change the machine type to one that utilizes GPUs. Moreover, this option would not use the autoscaling feature of Vertex AI, which can automatically adjust the number of replicas based on the traffic patterns, and provide various benefits, such as optimal resource utilization, cost savings, and performance

improvement2.

Reference:

Configure compute resources for prediction | Vertex AI | Google Cloud Deploy a model to an endpoint | Vertex AI | Google Cloud

Question: 169

You work at a bank You have a custom tabular ML model that was provided by the bank's vendor. The training data is not available due to its sensitivity. The model is packaged as a Vertex Al Model serving container which accepts a string as input for each prediction instance. In each string the feature values are separated by commas. You want to deploy this model to production for online predictions, and monitor the feature distribution over time with minimal effort What should you do?

A. 1 Upload the model to Vertex Al Model Registry and deploy the model to a Vertex Ai endpoint.

2. Create a Vertex Al Model Monitoring job with feature drift detection as the monitoring objective, and provide an instance schema.

B. 1 Upload the model to Vertex Al Model Registry and deploy the model to a Vertex Al endpoint.

2 Create a Vertex Al Model Monitoring job with feature skew detection as the monitoring objective and provide an instance schema.

C. 1 Refactor the serving container to accept key-value pairs as input format.

2. Upload the model to Vertex Al Model Registry and deploy the model to a Vertex Al endpoint.

3. Create a Vertex Al Model Monitoring job with feature drift detection as the monitoring objective.

D. 1 Refactor the serving container to accept key-value pairs as input format.

2 Upload the model to Vertex Al Model Registry and deploy the model to a Vertex Al endpoint.

3 . Create a Vertex Al Model Monitoring job with feature skew detection as the monitoring objective.

Answer: A

Explanation:

The best option for deploying a custom tabular ML model to production for online predictions, and monitoring the feature distribution over time with minimal effort, using a model that was provided by the bank's vendor, the training data is not available due to its sensitivity, and the model is packaged as a Vertex AI Model serving container which accepts a string as input for each prediction instance, is to upload the model to Vertex AI Model Registry and deploy the model to a Vertex AI endpoint, create a Vertex AI Model Monitoring job with feature drift detection as the monitoring objective, and provide an instance schema. This option allows you to leverage the power and simplicity of Vertex AI to serve and monitor your model with minimal code and configuration. Vertex AI is a unified platform for building and deploying machine learning solutions on Google Cloud. Vertex AI can deploy a trained model to an online prediction endpoint, which can provide low- latency predictions for individual instances. Vertex AI can also provide various tools and services for data analysis, model development, model deployment, model monitoring, and model governance. A Vertex AI Model Registry is a resource that can store and manage your models on Vertex AI. A Vertex AI Model Registry can help you organize and track your models, and access various model information, such as model name, model description, and model labels. A Vertex AI Model serving container is a resource that can run your custom model code on Vertex AI. A Vertex AI Model serving container can help you package your model code and dependencies into a container image, and deploy the container image to an online prediction endpoint. A Vertex AI Model serving container can accept various input formats, such as JSON, CSV, or TFRecord. A string input format is a type of input format that accepts a string as input for each prediction instance. A string input format can help you encode your feature values into a single string, and separate them by commas. By uploading the model to Vertex AI Model Registry and deploying the

model to a Vertex AI endpoint, you can serve your model for online predictions with minimal code and configuration. You can use the Vertex AI API or the gcloud command-line tool to upload the model to Vertex AI Model Registry, and provide the model name, model description, and model labels. You can also use the Vertex AI API or the gcloud command-line tool to deploy the model to a Vertex AI endpoint, and provide the endpoint name, endpoint description, endpoint labels, and endpoint resources. A Vertex AI Model Monitoring job is a resource that can monitor the performance and quality of your deployed models on Vertex AI. A Vertex AI Model Monitoring job can help you detect and diagnose issues with your models, such as data drift, prediction drift, training/serving skew, or model staleness. Feature drift is a type of model monitoring metric that measures the difference between the distributions of the features used to

train the model and the features used to serve the model over time. Feature drift can indicate that the online data is changing over time, and the model performance is degrading. By creating a Vertex AI Model Monitoring job with feature drift detection as the monitoring objective, and providing an instance schema, you can monitor the feature distribution over time with minimal effort. You can use the Vertex AI API or the gcloud command-line tool to create a Vertex AI Model Monitoring job, and provide the monitoring objective, the monitoring frequency, the alerting threshold, and the notification channel. You can also provide an instance schema, which is a JSON file that describes the features and their types in the prediction input data. An instance schema can help Vertex AI Model Monitoring parse and analyze the string input format, and calculate the feature distributions and distance scores1.

The other options are not as good as option A, for the following reasons:

Option B: Uploading the model to Vertex AI Model Registry and deploying the model to a Vertex AI endpoint, creating a Vertex AI Model Monitoring job with feature skew detection as the monitoring objective, and providing an instance schema would not help you monitor the changes in the online data over time, and could cause errors or poor performance. Feature skew is a type of model monitoring metric that measures the difference between the distributions of the features used to train the model and the features used to serve the model at a given point in time. Feature skew can indicate that the model is not trained on the representative data, or that the data is changing over time. By creating a Vertex AI Model Monitoring job with feature skew detection as the monitoring objective, and providing an instance schema, you can monitor the feature distribution at a given point in time with minimal effort. However, uploading the model to Vertex AI Model Registry and deploying the model to a Vertex AI endpoint, creating a Vertex AI Model Monitoring job with feature skew detection as the monitoring objective, and providing an instance schema would not help you monitor the changes in the online data over time, and could cause errors or poor performance. You would need to use the Vertex AI API or the gcloud command-line tool to upload the model to Vertex AI Model Registry, deploy the model to a Vertex AI endpoint, create a Vertex AI Model Monitoring job, and provide an instance schema. Moreover, this option would not monitor the feature drift, which is a more direct and relevant metric for measuring the changes in the online data over time, and the model performance and quality1.

Option C: Refactoring the serving container to accept key-value pairs as input format, uploading the model to Vertex AI Model Registry and deploying the model to a Vertex AI endpoint, creating a Vertex AI Model Monitoring job with feature drift detection as the monitoring objective would require more skills and steps than uploading the model to Vertex AI Model Registry and deploying the model to a Vertex AI endpoint, creating a Vertex AI Model Monitoring job with feature drift detection as the monitoring objective, and providing an instance schema. A key-value pair input format is a type of input format that accepts a key-value pair as input for each prediction instance. A key-value pair input format can help you specify the feature names and values in a JSON object, and separate them by colons. By refactoring the serving container to accept key-value pairs as input format, uploading the model to Vertex AI Model Registry and deploying the model to a Vertex AI endpoint, creating a Vertex AI Model Monitoring job with feature drift detection as the monitoring objective, you can serve and monitor your model with minimal code and configuration. You can write code to refactor the serving container to accept key-value pairs as input format, and use the Vertex AI API or the gcloud command-line tool to upload the model to Vertex AI Model Registry, deploy the model to a Vertex AI endpoint, and create a Vertex AI Model Monitoring job. However, refactoring the serving container to accept key-value pairs as input format, uploading the model to Vertex AI Model Registry and deploying the model to a Vertex AI endpoint, creating a Vertex AI Model Monitoring job with feature drift detection as the monitoring objective would require more skills and steps than uploading the model to Vertex AI Model Registry and deploying the model to a Vertex

AI
endpoint, creating a Vertex AI Model Monitoring job with feature drift detection as the monitoring objective, and providing an instance schema. You would need to write code, refactor the serving container, upload the model to Vertex AI Model Registry, deploy the model to a Vertex AI endpoint, and create a Vertex AI Model Monitoring job. Moreover, this option would not use the instance schema, which is a JSON file that can help Vertex AI Model Monitoring parse and analyze the string input format, and calculate the feature distributions and distance scores1.

Option D: Refactoring the serving container to accept key-value pairs as input format, uploading the model to Vertex AI Model Registry and deploying the model to a Vertex AI endpoint, creating a Vertex AI Model Monitoring job with feature skew detection as the monitoring objective would require more skills and steps than uploading the model to Vertex AI Model Registry and deploying the model to a Vertex AI endpoint, creating a Vertex AI Model Monitoring job with feature drift detection as the monitoring objective, and providing an instance schema, and would not help you monitor the changes in the online data over time, and could cause errors or poor performance. Feature skew is a type of model monitoring metric that measures the difference between the distributions of the features used to train the model and the features used to serve the model at a given point in time. Feature skew can indicate that the model is not trained on the representative data, or that the data is changing over time. By creating a Vertex AI Model Monitoring job with feature skew detection as the monitoring objective, you can monitor the feature distribution at a given point in time with minimal effort. However, refactoring the serving container to accept keyvalue pairs as input format, uploading the model to Vertex AI Model Registry and deploying the model to a Vertex AI endpoint, creating a Vertex AI Model Monitoring job with feature skew detection as the monitoring objective would require more skills and steps than uploading the model to Vertex AI Model Registry and deploying the model to a Vertex AI endpoint, creating a Vertex AI Model Monitoring job with feature drift detection as the monitoring objective, and providing an instance schema, and would not help you monitor the changes in the online data over time, and could cause errors or poor performance. You would need to write code, refactor the serving container, upload the model to Vertex AI Model Registry, deploy the model to a Vertex AI endpoint, and create a Vertex AI Model Monitoring job. Moreover, this option would not monitor the feature drift, which is a more direct and relevant metric for measuring the changes in the online data over time, and the model performance and quality1.

Reference:
Using Model Monitoring | Vertex AI | Google Cloud

Question: 170

You are implementing a batch inference ML pipeline in Google Cloud. The model was developed using TensorFlow and is stored in SavedModel format in Cloud Storage You need to apply the model to a historical dataset containing 10 TB of data that is stored in a BigQuery table How should you perform the inference?

A. Export the historical data to Cloud Storage in Avro format. Configure a Vertex AI batch prediction job to generate predictions for the exported data.
B. Import the TensorFlow model by using the create model statement in BigQuery ML Apply the historical data to the TensorFlow model.
C. Export the historical data to Cloud Storage in CSV format Configure a Vertex AI batch prediction job to generate predictions for the exported data.
D. Configure a Vertex AI batch prediction job to apply the model to the historical data in BigQuery

Answer: D

Explanation:
The best option for implementing a batch inference ML pipeline in Google Cloud, using a model that was developed using TensorFlow and is stored in SavedModel format in Cloud Storage, and a historical dataset containing 10 TB of data

that is stored in a BigQuery table, is to configure a Vertex AI batch prediction job to apply the model to the historical data in BigQuery. This option allows you to leverage the power and simplicity of Vertex AI and BigQuery to perform large-scale batch inference with minimal code and configuration. Vertex AI is a unified platform for building and deploying machine learning solutions on Google Cloud. Vertex AI can run a batch prediction job, which can generate predictions for a large number of instances in batches. Vertex AI can also provide various tools and services for data analysis, model development, model deployment, model monitoring, and model governance. A batch prediction job is a resource that can run your model code on Vertex AI. A batch prediction job can help you generate predictions for a large number of instances in batches, and store the prediction results in a destination of your choice. A batch prediction job can accept various input formats, such as JSON, CSV, or TFRecord. A batch prediction job can also accept various input sources, such as Cloud Storage or BigQuery. A TensorFlow model is a resource that represents a machine learning model that is built using TensorFlow. TensorFlow is a framework that can perform large-scale data processing and machine learning. TensorFlow can help you build and train various types of models, such as linear regression, logistic regression, k-means clustering, matrix factorization, and deep neural networks. A SavedModel format is a type of format that can store a TensorFlow model and its associated assets. A SavedModel format can help you save and load your TensorFlow model, and serve it for prediction. A SavedModel format can be stored in Cloud Storage, which is a service that can store and access large-scale data on Google Cloud. A historical dataset is a collection of data that contains historical information about a certain domain. A historical dataset can help you analyze the past trends and patterns of the data, and make predictions for the future. A historical dataset can be stored in BigQuery, which is a service that can store and query large-scale data on Google Cloud. BigQuery can help you analyze your data by using SQL queries, and perform various tasks, such as data exploration, data transformation, or data visualization. By configuring a Vertex AI batch prediction job to apply the model to the historical data in BigQuery, you can implement a batch inference ML pipeline in Google Cloud with minimal code and configuration. You can use the Vertex AI API or the gcloud command-line tool to configure a batch prediction job, and provide the model name, the model version, the input source, the input format, the output destination, and the output format. Vertex AI will automatically run the batch prediction job, and apply the model to the historical data in BigQuery. Vertex AI will also store the prediction results in a destination of your choice, such as Cloud Storage or BigQuery1.

The other options are not as good as option D, for the following reasons:

Option A: Exporting the historical data to Cloud Storage in Avro format, configuring a Vertex AI batch prediction job to generate predictions for the exported data would require more skills and steps than configuring a Vertex AI batch prediction job to apply the model to the historical data in BigQuery, and could increase the complexity and cost of the batch inference process. Avro is a type of format that can store and serialize data in a binary format. Avro can help you compress and encode your data, and support schema evolution and compatibility. By exporting the historical data to Cloud Storage in Avro format, configuring a Vertex AI batch prediction job to generate predictions for the exported data, you can perform batch inference with minimal code and configuration. You can use the BigQuery API or the bq command-line tool to export the historical data to Cloud Storage in Avro format, and use the Vertex AI API or the gcloud command-line tool to configure a batch prediction job, and provide the model name, the model version, the input source, the input format, the output destination, and the output format. However, exporting the historical data to Cloud Storage in Avro format, configuring a Vertex AI batch prediction job to generate predictions for the exported data would require more skills and steps than configuring a Vertex AI batch prediction job to apply the model to the historical data in BigQuery, and could increase the complexity and cost of the batch inference process. You would need to write code, export the historical data to Cloud Storage, configure a batch prediction job, and generate predictions for the exported data. Moreover, this option would not use BigQuery as the input source for the batch prediction job, which can simplify the batch inference process, and provide various benefits, such as fast query performance, serverless scaling, and cost optimization2.

Option B: Importing the TensorFlow model by using the create model statement in BigQuery ML, applying the historical data to the TensorFlow model would not allow you to use Vertex AI to run the batch prediction job, and could increase the complexity and cost of the batch inference process. BigQuery ML is a feature of BigQuery that can create and execute machine learning models in BigQuery by using SQL queries. BigQuery ML can help you build and train various types of models, such as linear regression, logistic regression, k-means clustering, matrix factorization, and deep neural

networks. A create model statement is a type of SQL statement that can create a machine learning model in BigQuery ML. A create model statement can help you specify the model name, the model type, the model options, and the model query. By importing the TensorFlow model by using the create model statement in BigQuery ML, applying the historical data to the TensorFlow model, you can perform batch inference with minimal code and configuration. You can use the BigQuery API or the bq command-line tool to import the TensorFlow model by using the create model statement in BigQuery ML, and provide the model name, the model type, the model options, and the model query. You can also use the BigQuery API or the bq command-line tool to apply the historical data to the TensorFlow model, and provide the model name, the input data, and the output destination. However, importing the TensorFlow model by using the create model statement in BigQuery ML, applying the historical data to the TensorFlow model would not allow you to use Vertex AI to run the batch prediction job, and could increase the complexity and cost of the batch inference process. You would need to write code, import the TensorFlow model, apply the historical data, and generate predictions. Moreover, this option would not use Vertex AI, which is a unified platform for building and deploying machine learning solutions on Google Cloud, and provide various tools and services for data analysis, model development, model deployment, model monitoring, and model governance3.

Option C: Exporting the historical data to Cloud Storage in CSV format, configuring a Vertex AI batch prediction job to generate predictions for the exported data would require more skills and steps than configuring a Vertex AI batch prediction job to apply the model to the historical data in BigQuery, and could increase the complexity and cost of the batch inference process. CSV is a type of format that can store and serialize data in a comma-separated values format. CSV can help you store and exchange your data, and support various data types and formats. By exporting the historical data to Cloud Storage in CSV format, configuring a Vertex AI batch prediction job to generate predictions for the exported data, you can perform batch inference with minimal code and configuration. You can use the BigQuery API or the bq command-line tool to export the historical data to Cloud Storage in CSV format, and use the Vertex AI API or the gcloud command-line tool to configure a batch prediction job, and provide the model name, the model version, the input source, the input format, the output destination, and the output format. However, exporting the historical data to Cloud Storage in CSV format, configuring a Vertex AI batch prediction job to generate predictions for the exported data would require more skills and steps than configuring a Vertex AI batch prediction job to apply the model to the historical data in BigQuery, and could increase the complexity and cost of the batch inference process. You would need to write code, export the historical data to Cloud Storage, configure a batch prediction job, and generate predictions for the exported data. Moreover, this option would not use BigQuery as the input source for the batch prediction job, which can simplify the batch inference process, and provide various benefits, such as fast query performance, serverless scaling, and cost optimization2.

Reference:

Batch prediction | Vertex AI | Google Cloud

Exporting table data | BigQuery | Google Cloud

Creating and using models | BigQuery ML | Google Cloud

Question: 171

You recently deployed a model to a Vertex Al endpoint Your data drifts frequently so you have enabled request-response logging and created a Vertex Al Model Monitoring job. You have observed that your model is receiving higher traffic than expected. You need to reduce the model monitoring cost while continuing to quickly detect drift. What should you do?

A. Replace the monitoring job with a DataFlow pipeline that uses TensorFlow Data Validation (TFDV). B. Replace the monitoring job with a custom SQL scnpt to calculate statistics on the features and predictions in BigQuery.

C. Decrease the sample_rate parameter in the Randomsampleconfig of the monitoring job.

D. Increase the monitor_interval parameter in the scheduieconfig of the monitoring job.

Answer: C

Explanation:
According to the official exam guide1, one of the skills assessed in the exam is to "configure and optimize model monitoring jobs". The Vertex AI Model Monitoring documentation states that "to reduce the cost of model monitoring, you can configure the sample rate of the requests that are logged and analyzed by model monitoring". Therefore, decreasing the sample_rate parameter in the Randomsampleconfig of the monitoring job would reduce the model monitoring cost while continuing to quickly detect drift. The other options are not relevant or optimal for this scenario.
Reference:
Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate Professional ML Engineer Exam Guide
Google Professional Machine Learning Certification Exam 2023
Latest Google Professional Machine Learning Engineer Actual Free Exam Questions [Vertex AI Model Monitoring]

Question: 172

You work for a retail company. You have created a Vertex AI forecast model that produces monthly item sales predictions. You want to quickly create a report that will help to explain how the model calculates the predictions. You have one month of recent actual sales data that was not included in

the training dataset. How should you generate data for your report?

A. Create a batch prediction job by using the actual sales data Compare the predictions to the actuals in the report.
B. Create a batch prediction job by using the actual sates data and configure the job settings to generate feature attributions. Compare the results in the report.
C. Generate counterfactual examples by using the actual sales data Create a batch prediction job using the actual sales data and the counterfactual examples Compare the results in the report.
D. Train another model by using the same training dataset as the original and exclude some columns. Using the actual sales data create one batch prediction job by using the new model and another one with the original model Compare the two sets of predictions in the report.

Answer: B

Explanation:
According to the official exam guide1, one of the skills assessed in the exam is to "explain the predictions of a trained model". Vertex AI provides feature attributions using Shapley Values, a cooperative game theory algorithm that assigns credit to each feature in a model for a particular outcome2. Feature attributions can help you understand how the model calculates the predictions and debug or optimize the model accordingly. You can use Forecasting with AutoML or Tabular Workflow for Forecasting to generate and query local feature attributions2. The other options are not relevant or optimal for this scenario. Reference: Professional ML Engineer Exam Guide Feature attributions for forecasting
Google Professional Machine Learning Certification Exam 2023
Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

www.atmicnetworks.com

Question: 173

Your team has a model deployed to a Vertex AI endpoint You have created a Vertex AI pipeline that automates the model training process and is triggered by a Cloud Function. You need to prioritize keeping the model up-to-date, but also minimize retraining costs. How should you configure retraining'?

A. Configure Pub/Sub to call the Cloud Function when a sufficient amount of new data becomes available.
B. Configure a Cloud Scheduler job that calls the Cloud Function at a predetermined frequency that fits your team's budget.
C. Enable model monitoring on the Vertex AI endpoint Configure Pub/Sub to call the Cloud Function when anomalies are detected.
D. Enable model monitoring on the Vertex AI endpoint Configure Pub/Sub to call the Cloud Function when feature drift is detected.

Answer: D

Explanation:
According to the official exam guide1, one of the skills assessed in the exam is to "configure and optimize model monitoring jobs". Vertex AI Model Monitoring documentation states that "model monitoring helps you detect when your model's performance degrades over time due to changes in the data that your model receives or returns" and that "you can configure model monitoring to send notifications to Pub/Sub when it detects anomalies or drift in your model's predictions"2. Therefore, enabling model monitoring on the Vertex AI endpoint and configuring Pub/Sub to call the Cloud Function when feature drift is detected would help you keep the model up-to-date and minimize retraining costs. The other options are not relevant or optimal for this scenario. Reference: Professional ML Engineer Exam Guide
Vertex AI Model Monitoring
Google Professional Machine Learning Certification Exam 2023
Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

Question: 174

Your company stores a large number of audio files of phone calls made to your customer call center in an on-premises database. Each audio file is in wav format and is approximately 5 minutes long. You need to analyze these audio files for customer sentiment. You plan to use the Speech-to-Text API. You want to use the most efficient approach. What should you do?

A. 1 Upload the audio files to Cloud Storage
   2. Call the speech: longrunningrecognize API endpoint to generate transcriptions
   3. Call the predict method of an AutoML sentiment analysis model to analyze the transcriptions
B. 1 Upload the audio files to Cloud Storage
   2 Call the speech: longrunningrecognize API endpoint to generate transcriptions.
   3 Create a Cloud Function that calls the Natural Language API by using the analyzesentiment method
C. 1 Iterate over your local Tiles in Python
   2. Use the Speech-to-Text Python library to create a speech.RecognitionAudio object and set the content to the audio file data
   3. Call the speech: recognize API endpoint to generate transcriptions
   4. Call the predict method of an AutoML sentiment analysis model to analyze the transcriptions
D. 1 Iterate over your local files in Python

2 Use the Speech-to-Text Python Library to create a speech.RecognitionAudio object, and set the content to the audio file data

3 . Call the speech: lengrunningrecognize API endpoint to generate transcriptions

4 Call the Natural Language API by using the analyzesenriment method

Answer: B

Explanation:

According to the official exam guide1, one of the skills assessed in the exam is to "design, build, and productionalize ML models to solve business challenges using Google Cloud technologies". The Speech-to-Text API2 allows you to convert audio to text by applying powerful neural network models. The Natural Language API3 enables you to analyze text and extract information about the sentiment, entities, and syntax. The Cloud Functions4 service lets you write and deploy code that

runs in response to events, such as a Pub/Sub message or an HTTP request. Therefore, option B is the most efficient approach to analyze the audio files for customer sentiment, as it leverages the existing Google Cloud services and avoids unnecessary data processing and model training. The other options are not relevant or optimal for this scenario.

Reference: Professional ML Engineer Exam Guide Speech-to-Text API Natural Language API Cloud Functions

Google Professional Machine Learning Certification Exam 2023

Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

Question: 175

You work for a social media company. You want to create a no-code image classification model for an iOS mobile application to identify fashion accessories You have a labeled dataset in Cloud Storage You need to configure a training workflow that minimizes cost and serves predictions with the lowest possible latency What should you do?

A. Train the model by using AutoML, and register the model in Vertex Al Model Registry Configure your mobile application to send batch requests during prediction.

B. Train the model by using AutoML Edge and export it as a Core ML model Configure your mobile application to use the mlmodel file directly.

C. Train the model by using AutoML Edge and export the model as a TFLite model Configure your mobile application to use the tflite file directly

D. Train the model by using AutoML, and expose the model as a Vertex Al endpoint Configure your mobile application to invoke the endpoint during prediction.

Answer: B

Explanation:

AutoML Edge is a service that allows you to train and deploy custom image classification models for mobile devices12. It supports exporting models as Core ML files, which are compatible with iOS applications3.

Using a Core ML model directly on the device eliminates the need for network requests and reduces prediction latency. It also minimizes the cost of serving predictions, as there is no need to pay for cloud resources or network bandwidth.

Option A is incorrect because sending batch requests during prediction does not reduce latency, as the requests still need to be processed by the cloud service. It also incurs more cost than using a local model on the device.

Option C is incorrect because TFLite models are not compatible with iOS applications. TFLite models are designed for Android and other platforms that support TensorFlow Lite4.

Option D is incorrect because exposing the model as a Vertex AI endpoint requires network requests and cloud resources, which increase latency and cost. It also does not leverage the benefits of AutoML Edge, which is optimized for mobile devices.

Question: 176

You work for a retail company. You have been asked to develop a model to predict whether a customer will purchase a product on a given day. Your team has processed the company's sales data, and created a table with the following rows:

- Customer_id
- Product_id
- Date
- Days_since_last_purchase (measured in days)
- Average_purchase_frequency (measured in 1/days)
- Purchase (binary class, if customer purchased product on the Date)

You need to interpret your models results for each individual prediction. What should you do?

A. Create a BigQuery table Use BigQuery ML to build a boosted tree classifier Inspect the partition rules of the trees to understand how each prediction flows through the trees.

B. Create a Vertex AI tabular dataset Train an AutoML model to predict customer purchases Deploy the model to a Vertex AI endpoint and enable feature attributions Use the "explain" method to get feature attribution values for each individual prediction.

C. Create a BigQuery table Use BigQuery ML to build a logistic regression classification model Use the values of the coefficients of the model to interpret the feature importance with higher values corresponding to more importance.

D. Create a Vertex AI tabular dataset Train an AutoML model to predict customer purchases Deploy the model to a Vertex AI endpoint. At each prediction enable L1 regularization to detect non- informative features.

Answer: B

Explanation:

According to the official exam guide1, one of the skills assessed in the exam is to “explain the predictions of a trained model”. Vertex AI provides feature attributions using Shapley Values, a cooperative game theory algorithm that assigns credit to each feature in a model for a particular outcome2. Feature attributions can help you understand how the model calculates the predictions and debug or optimize the model accordingly. You can use AutoML for Tabular Data to generate and query local feature attributions3. The other options are not relevant or optimal for this scenario.

Reference:

Professional ML Engineer Exam Guide

Feature attributions for classification and regression

AutoML for Tabular Data

Google Professional Machine Learning Certification Exam 2023

Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

Question: 177

You work for a company that captures live video footage of checkout areas in their retail stores You need to use the live video footage to build a mode! to detect the number of customers waiting for service in near real

time You want to implement a solution quickly and with minimal effort How should you build the model?

A. Use the Vertex Al Vision Occupancy Analytics model.
B. Use the Vertex Al Vision Person/vehicle detector model
C. Train an AutoML object detection model on an annotated dataset by using Vertex AutoML
D. Train a Seq2Seq+ object detection model on an annotated dataset by using Vertex AutoML

Answer: A

Explanation:
According to the official exam guide1, one of the skills assessed in the exam is to "design, build, and productionalize ML models to solve business challenges using Google Cloud technologies". The Vertex AI Vision Occupancy Analytics model2 is a specialized pre-built vision model that lets you count people or vehicles given specific inputs you add in video frames. It provides advanced features such as active zones counting, line crossing counting, and dwelling detection. This model is suitable for the use case of detecting the number of customers waiting for service in near real time. You can easily create and deploy an occupancy analytics application using Vertex AI Vision3. The other options are not relevant or optimal for this scenario. Reference: Professional ML Engineer Exam Guide Occupancy analytics guide
Create an occupancy analytics app with BigQuery forecasting
Google Professional Machine Learning Certification Exam 2023
Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

Question: 178

You work as an analyst at a large banking firm. You are developing a robust, scalable ML pipeline to train several regression and classification models. Your primary focus for the pipeline is model interpretability. You want to productionize the pipeline as quickly as possible What should you do?

A. Use Tabular Workflow for Wide & Deep through Vertex Al Pipelines to jointly train wide linear models and deep neural networks.
B. Use Google Kubernetes Engine to build a custom training pipeline for XGBoost-based models.
C. Use Tabular Workflow forTabel through Vertex Al Pipelines to train attention-based models.
D. Use Cloud Composer to build the training pipelines for custom deep learning-based models.

Answer: D

Explanation:
According to the official exam guide1, one of the skills assessed in the exam is to "automate and orchestrate ML pipelines using Cloud Composer". Cloud Composer2 is a fully managed workflow orchestration service that uses Apache Airflow to create, schedule, monitor, and manage workflows. Cloud Composer allows you to build custom training pipelines for deep learning-based models and
integrate them with other Google Cloud services. You can also use Cloud Composer to implement model interpretability techniques, such as feature attributions, explainable AI, or model debugging3. The other options are not relevant or optimal for this scenario. Reference:
Professional ML Engineer Exam Guide
Cloud Composer
Model interpretability with Cloud Composer
Google Professional Machine Learning Certification Exam 2023
Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

www.atmicnetworks.com

Question: 179

You developed a Transformer model in TensorFlow to translate text Your training data includes millions of documents in a Cloud Storage bucket. You plan to use distributed training to reduce training time. You need to configure the training job while minimizing the effort required to modify code and to manage the clusters configuration. What should you do?

A. Create a Vertex Al custom training job with GPU accelerators for the second worker pool Use tf .distribute.MultiWorkerMirroredStrategy for distribution.

B. Create a Vertex Al custom distributed training job with Reduction Server Use N1 high-memory machine type instances for the first and second pools, and use N1 high-CPU machine type instances for the third worker pool.

C. Create a training job that uses Cloud TPU VMs Use tf.distribute.TPUStrategy for distribution.

D. Create a Vertex Al custom training job with a single worker pool of A2 GPU machine type instances Use tf .distribute.MirroredStraregy for distribution.

Answer: C

Explanation:
According to the official exam guide1, one of the skills assessed in the exam is to "configure and optimize model training jobs". Cloud TPU VMs2 are a new way to access Cloud TPUs directly on the TPU host machines, offering a simpler and more flexible user experience. Cloud TPU VMs are optimized for ML model training and can reduce training time and cost. You can use Cloud TPU VMs to train Transformer models in TensorFlow by using the tf.distribute.TPUStrategy3, which handles the distribution of computations across the TPU cores. The other options are not relevant or optimal for this scenario. Reference:

Professional ML Engineer Exam Guide Cloud TPU VMs
Distributed training with TPUStrategy
Google Professional Machine Learning Certification Exam 2023
Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

Question: 180

You are developing a process for training and running your custom model in production. You need to be able to show lineage for your model and predictions. What should you do?

A. 1 Create a Vertex Al managed dataset
2 Use a Vertex Ai training pipeline to train your model
3 Generate batch predictions in Vertex Al

B. 1 Use a Vertex Al Pipelines custom training job component to train your model
2. Generate predictions by using a Vertex Al Pipelines model batch predict component

C. 1 Upload your dataset to BigQuery
2 . Use a Vertex Al custom training job to train your model
3 Generate predictions by using Vertex Al SDK custom prediction routines

D. 1 Use Vertex Al Experiments to train your model.
2 Register your model in Vertex Al Model Registry
3 . Generate batch predictions in Vertex Al

Answer: D

Explanation:
According to the official exam guide1, one of the skills assessed in the exam is to "track the lineage of pipeline artifacts". Vertex AI Experiments2 is a service that allows you to track and compare the results of your model training runs. Vertex

AI Experiments automatically logs metadata such as hyperparameters, metrics, and artifacts for each training run. You can use Vertex AI Experiments to train your custom model using TensorFlow, PyTorch, XGBoost, or scikit-learn. Vertex AI Model Registry3 is a service that allows you to manage your trained models in a central location. You can use Vertex AI Model Registry to register your model, add labels and descriptions, and view the model's lineage graph. The lineage graph shows the artifacts and executions that are part of the model's creation, such as the dataset, the training pipeline, and the evaluation metrics. The other options are not relevant or optimal for this scenario. Reference:

Professional ML Engineer Exam Guide

Vertex AI Experiments

Vertex AI Model Registry

Google Professional Machine Learning Certification Exam 2023

Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

Question: 181

You work for a hotel and have a dataset that contains customers' written comments scanned from paper-based customer feedback forms which are stored as PDF files Every form has the same layout. You need to quickly predict an overall satisfaction score from the customer comments on each form. How should you accomplish this task'?

A. Use the Vision API to parse the text from each PDF file Use the Natural Language API analyzesentiment feature to infer overall satisfaction scores.

B. Use the Vision API to parse the text from each PDF file Use the Natural Language API analyzeEntitysentiment feature to infer overall satisfaction scores.

C. Uptrain a Document Al custom extractor to parse the text in the comments section of each PDF file. Use the Natural Language API analyze sentiment feature to infer overall satisfaction scores.

D. Uptrain a Document Al custom extractor to parse the text in the comments section of each PDF file. Use the Natural Language API analyzeEntitySentiment feature to infer overall satisfaction scores.

Answer: C

Explanation:

According to the official exam guide1, one of the skills assessed in the exam is to "design, build, and productionalize ML models to solve business challenges using Google Cloud technologies". Document AI2 is a document understanding platform that takes unstructured data from documents and transforms it into structured data, making it easier to understand, analyze, and consume. Document AI Workbench3 allows you to create custom extractors to parse the text in specific sections of your documents. Natural Language API4 is a service that provides natural language understanding technologies, such as sentiment analysis, entity analysis, and other text annotations. The analyzeSentiment feature5 inspects the given text and identifies the prevailing emotional opinion within the text, especially to determine a writer's attitude as positive, negative, or neutral. Therefore, option C is the best way to accomplish the task of predicting an overall satisfaction score from the customer comments on each form. The other options are not relevant or optimal for this scenario. Reference: Professional ML Engineer Exam Guide Document AI

Document AI Workbench Natural Language API Sentiment analysis

Google Professional Machine Learning Certification Exam 2023

Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

Question: 182

You developed a Vertex Al pipeline that trains a classification model on data stored in a large BigQuery table. The

pipeline has four steps, where each step is created by a Python function that uses the KubeFlow v2 API The components have the following names:

```
dt=datetime.now().strftime("%Y%m%d%H%M%S")
f"export-{dt).yaml", f"preprocess-(dt}.yaml", f"train-(dt}.yaml",
{"calibrate-{dt}.yaml"
```

You launch your Vertex Al pipeline as the following:

```
job = aip.PipelineJob(
    display_name="my-awesome-pipeline temp1az
    e_cath="pipeline.j s on", jcb_id=f"my-aweseme-
    pipeline-{dt■", parameter_values=params,
    enable_caching=True, 1o cation="europe-wes11"
```

You perform many model iterations by adjusting the code and parameters of the training step. You observe high costs associated with the development, particularly the data export and preprocessing steps. You need to reduce model development costs. What should you do?

A.

Change the components YAML filenames to export.yaml, preprocess.yaml, f"train- ■ dr}.yaml", f"calibrate-{dt}.yaml"

B.

Add the {"kubeflow.vl. caching": True} parameter to the set of params provided to your Pipelinejob

C.

Move the first step of your pipeline to a separate step and provide a cached path to Cloud Storage as an input to the mam pipeline

D.

Change the name of the pipeline to f ■my-awesome-pipeline- dt) ”

A. Option A B. Option B C. Option C D. Option D

Answer: A

Explanation:

According to the official exam guide1, one of the skills assessed in the exam is to “automate and orchestrate ML pipelines using Cloud Composer”. Vertex AI Pipelines2 is a service that allows you to orchestrate your ML workflows using Kubeflow Pipelines SDK v2 or TensorFlow Extended. Vertex AI Pipelines supports execution caching, which means that if you run a pipeline and it reaches a component that has already been run with the same inputs and parameters, the component does not run again. Instead, the component uses the output from the previous run. This can save you time and resources when you are iterating on your pipeline. Therefore, option A is the best way to reduce

model development costs, as it enables execution caching for the data export and preprocessing steps, which are likely to be the same for each model iteration. The other options are not relevant or optimal for this scenario.

Reference:

www.atmicnetworks.com

Professional ML Engineer Exam Guide

Vertex AI Pipelines

Google Professional Machine Learning Certification Exam 2023

Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

Question: 183

You work for a startup that has multiple data science workloads. Your compute infrastructure is currently on-premises. and the data science workloads are native to PySpark Your team plans to migrate their data science workloads to Google Cloud You need to build a proof of concept to migrate one data science job to Google Cloud You want to propose a migration process that requires minimal cost and effort. What should you do first?

A. Create a n2-standard-4 VM instance and install Java, Scala and Apache Spark dependencies on it. B. Create a Google Kubemetes Engine cluster with a basic node pool configuration install Java Scala, and Apache Spark dependencies on it.

C. Create a Standard (1 master. 3 workers) Dataproc cluster, and run a Vertex Al Workbench notebook instance on it.

D. Create a Vertex Al Workbench notebook with instance type n2-standard-4.

Answer: C

Explanation:

According to the official exam guide1, one of the skills assessed in the exam is to "design, build, and productionalize ML models to solve business challenges using Google Cloud technologies". Dataproc2 is a fully managed, fast, and easy-to-use service for running Apache Spark and Apache Hadoop clusters on Google Cloud. Dataproc supports PySpark workloads and provides a simple way to migrate your existing Spark jobs to the cloud. You can create a Dataproc cluster with a few clicks or commands, and run your PySpark jobs on it. You can also use Vertex AI Workbench3, a managed notebook service, to create and run PySpark notebooks on Dataproc clusters. This way, you can interactively develop and test your PySpark code on the cloud. Therefore, option C is the best way to build a proof of concept to migrate one data science job to Google Cloud with minimal cost and effort. The other options are not relevant or optimal for this scenario. Reference: Professional ML Engineer Exam Guide Dataproc

Vertex AI Workbench

Google Professional Machine Learning Certification Exam 2023

Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

Question: 184

You work for a bank You have been asked to develop an ML model that will support loan application decisions. You need to determine which Vertex Al services to include in the workflow You want to track the model's training parameters and the metrics per training epoch. You plan to compare the performance of each version of the model to determine the best model based on your chosen metrics. Which Vertex Al services should you use?

A. Vertex ML Metadata Vertex Al Feature Store, and Vertex Al Vizier

B. Vertex Al Pipelines. Vertex Al Experiments, and Vertex Al Vizier

C. Vertex ML Metadata Vertex Al Experiments, and Vertex Al TensorBoard

D. Vertex Al Pipelines. Vertex Al Feature Store, and Vertex Al TensorBoard

Answer: C

Explanation:

According to the official exam guide1, one of the skills assessed in the exam is to "track the lineage of pipeline artifacts". Vertex ML Metadata2 is a service that allows you to store, query, and visualize metadata associated with your ML workflows, such as datasets, models, metrics, and executions. Vertex ML Metadata helps you track the provenance and lineage of your ML artifacts and understand the relationships between them. Vertex AI Experiments3 is a service that allows you to track and compare the results of your model training runs. Vertex AI Experiments automatically logs metadata such as hyperparameters, metrics, and artifacts for each training run. You can use Vertex AI Experiments to train your custom model using TensorFlow, PyTorch, XGBoost, or scikit-learn. Vertex AI TensorBoard4 is a service that allows you to visualize and monitor your ML experiments using TensorBoard, an open source tool for ML visualization. Vertex AI TensorBoard helps you track the model's training parameters and the metrics per training epoch, and compare the performance of each version of the model. Therefore, option C is the best way to determine which Vertex AI services to include in the workflow for the given use case. The other options are not relevant or optimal for this scenario. Reference:

Professional ML Engineer Exam Guide

Vertex ML Metadata

Vertex AI Experiments

Vertex AI TensorBoard

Google Professional Machine Learning Certification Exam 2023

Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

Question: 185

You work for an auto insurance company. You are preparing a proof-of-concept ML application that uses images of damaged vehicles to infer damaged parts Your team has assembled a set of annotated images from damage claim documents in the company's database The annotations associated with each image consist of a bounding box for each identified damaged part and the part name. You have been given a sufficient budget to tram models on Google Cloud You need to quickly create an initial model What should you do?

A. Download a pre-trained object detection mode! from TensorFlow Hub Fine-tune the model in Vertex Al Workbench by using the annotated image data.

B. Train an object detection model in AutoML by using the annotated image data.

C. Create a pipeline in Vertex Al Pipelines and configure the AutoMLTrainingJobRunOp compon it to train a custom object detection model by using the annotated image data.

D. Train an object detection model in Vertex Al custom training by using the annotated image data.

Answer: B

Explanation:

According to the official exam guide1, one of the skills assessed in the exam is to "design, build, and productionalize ML models to solve business challenges using Google Cloud technologies". AutoML Vision2 is a service that allows you to train and deploy custom vision models for image classification and object detection. AutoML Vision simplifies the model development process by providing a graphical user interface and a no-code approach. You can use AutoML Vision to train an object detection model by using the annotated image data, and evaluate the model performance using metrics such as mean average precision (mAP) and intersection over union (IoU)3. Therefore, option B is the best way to quickly create an initial model for the given use case. The other options are not relevant or optimal for this scenario.

Reference: Professional ML Engineer Exam Guide

AutoML Vision

Object detection evaluation

www.atmicnetworks.com

Question: 186

You are analyzing customer data for a healthcare organization that is stored in Cloud Storage. The data contains personally identifiable information (PII) You need to perform data exploration and preprocessing while ensuring the security and privacy of sensitive fields What should you do?

A. Use the Cloud Data Loss Prevention (DLP) API to de-identify the PI! before performing data exploration and preprocessing.
B. Use customer-managed encryption keys (CMEK) to encrypt the Pll data at rest and decrypt the Pll data during data exploration and preprocessing.
C. Use a VM inside a VPC Service Controls security perimeter to perform data exploration and preprocessing.
D. Use Google-managed encryption keys to encrypt the Pll data at rest, and decrypt the Pll data during data exploration and preprocessing.

Answer: A

Explanation:
According to the official exam guide1, one of the skills assessed in the exam is to "design, build, and productionalize ML models to solve business challenges using Google Cloud technologies". Cloud Data Loss Prevention (DLP) API2 is a service that provides programmatic access to a powerful detection engine for personally identifiable information and other privacy-sensitive data in unstructured data streams, such as text blocks and images. Cloud DLP API helps you discover, classify, and protect your sensitive data by using techniques such as de-identification, masking, tokenization, and bucketing. You can use Cloud DLP API to de-identify the PII data before performing data exploration and preprocessing, and retain the data utility for ML purposes. Therefore, option A is the best way to perform data exploration and preprocessing while ensuring the security and privacy of sensitive fields. The other options are not relevant or optimal for this scenario. Reference: Professional ML Engineer Exam Guide Cloud Data Loss Prevention (DLP) API

Google Professional Machine Learning Certification Exam 2023
Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

Question: 187

You are building a predictive maintenance model to preemptively detect part defects in bridges. You plan to use high definition images of the bridges as model inputs. You need to explain the output of the model to the relevant stakeholders so they can take appropriate action. How should you build the model?

A. Use scikit-learn to build a tree-based model, and use SHAP values to explain the model output.
B. Use scikit-lean to build a tree-based model, and use partial dependence plots (PDP) to explain the model output.
C. Use TensorFlow to create a deep learning-based model and use Integrated Gradients to explain the model output.
D. Use TensorFlow to create a deep learning-based model and use the sampled Shapley method to explain the model output.

Answer: C

Explanation:
According to the official exam guide1, one of the skills assessed in the exam is to "explain the predictions of a trained model". TensorFlow2 is an open source framework for developing and deploying machine learning and deep learning models. TensorFlow supports various model explainability methods, such as Integrated Gradients3, which is a technique that assigns an importance score to each input feature by approximating the integral of the gradients along the path from a baseline input to the actual input. Integrated Gradients can help explain the output of a deep learning-based model by highlighting the most influential features in the input images. Therefore, option C is the best way to build the model for the given use case. The other options are not relevant or optimal for this scenario. Reference:
Professional ML Engineer Exam Guide
TensorFlow
Integrated Gradients
Google Professional Machine Learning Certification Exam 2023
Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

Question: 188

You work for a hospital that wants to optimize how it schedules operations. You need to create a model that uses the relationship between the number of surgeries scheduled and beds used You want to predict how many beds will be needed for patients each day in advance based on the scheduled surgeries You have one year of data for the hospital organized in 365 rows The data includes the following variables for each day

- Number of scheduled surgeries
- Number of beds occupied
- Date

You want to maximize the speed of model development and testing What should you do?

A. Create a BigQuery table Use BigQuery ML to build a regression model, with number of beds as the target variable and number of scheduled surgeries and date features (such as day of week) as the predictors
B. Create a BigQuery table Use BigQuery ML to build an ARIMA model, with number of beds as the target variable and date as the time variable.
C. Create a Vertex Al tabular dataset Tram an AutoML regression model, with number of beds as the target variable and number of scheduled minor surgeries and date features (such as day of the week) as the predictors
D. Create a Vertex Al tabular dataset Train a Vertex Al AutoML Forecasting model with number of beds as the target variable, number of scheduled surgeries as a covariate, and date as the time variable.

Answer: D

Explanation:
According to the official exam guide1, one of the skills assessed in the exam is to "design, build, and productionalize ML models to solve business challenges using Google Cloud technologies". Vertex AI AutoML Forecasting2 is a service that allows you to train and deploy custom time-series forecasting models for batch prediction. Vertex AI AutoML Forecasting simplifies the model development process by providing a graphical user interface and a no-code approach. You can use Vertex AI AutoML Forecasting to train a model by using your tabular data, and specify the target variable, the covariates, and the time variable. Vertex AI AutoML Forecasting automatically handles the feature engineering, model selection, and hyperparameter tuning. Therefore, option D is the best way to maximize the speed of model development and testing for the given use case. The other options are not relevant or optimal for this scenario.
Reference: Professional ML Engineer Exam Guide

www.atmicnetworks.com

Question: 189

You recently developed a wide and deep model in TensorFlow. You generated training datasets using a SQL script that preprocessed raw data in BigQuery by performing instance-level transformations of the dat

E. You need to create a training pipeline to retrain the model on a weekly basis. The trained model will be used to generate daily recommendations. You want to minimize model development and training time. How should you develop the training pipeline?

A. Use the Kubeflow Pipelines SDK to implement the pipeline Use the BigQueryJobop component to run the preprocessing script and the customTrainingJobop component to launch a Vertex AI training job.

B. Use the Kubeflow Pipelines SDK to implement the pipeline. Use the dataflowpythonjobopcomponent to preprocess the data and the customTraining JobOp component to launch a Vertex AI training job.

C. Use the TensorFlow Extended SDK to implement the pipeline Use the Examplegen component with the BigQuery executor to ingest the data the Transform component to preprocess the data, and the Trainer component to launch a Vertex AI training job.

D. Use the TensorFlow Extended SDK to implement the pipeline Implement the preprocessing steps as part of the input_fn of the model Use the ExampleGen component with the BigQuery executor to ingest the data and the Trainer component to launch a Vertex AI training job.

Answer: C

Explanation:

TensorFlow Extended (TFX) is a platform for building end-to-end machine learning pipelines using TensorFlow. TFX provides a set of components that can be orchestrated using either the TFX SDK or Kubeflow Pipelines. TFX components can handle different aspects of the pipeline, such as data ingestion, data validation, data transformation, model training, model evaluation, model serving, and more. TFX components can also leverage other Google Cloud services, such as BigQuery, Dataflow, and Vertex AI.

Why not A: Using the Kubeflow Pipelines SDK to implement the pipeline is a valid option, but using the BigQueryJobOp component to run the preprocessing script is not optimal. This would require writing and maintaining a separate SQL script for data transformation, which could introduce inconsistencies and errors. It would also make it harder to reuse the same preprocessing logic for both training and serving.

Why not B: Using the Kubeflow Pipelines SDK to implement the pipeline is a valid option, but using the DataflowPythonJobOp component to preprocess the data is not optimal. This would require writing and maintaining a separate Python script for data transformation, which could introduce inconsistencies and errors. It would also make it harder to reuse the same preprocessing logic for both training and serving.

Why not D: Using the TensorFlow Extended SDK to implement the pipeline is a valid option, but implementing the preprocessing steps as part of the input_fn of the model is not optimal. This would make the preprocessing logic tightly coupled with the model code, which could reduce modularity and flexibility. It would also make it harder to reuse the same preprocessing logic for both training and serving.

Question: 190

You are training a custom language model for your company using a large dataset. You plan to use the ReductionServer

strategy on Vertex AI. You need to configure the worker pools of the distributed training job. What should you do?

A. Configure the machines of the first two worker pools to have GPUs and to use a container image where your training code runs Configure the third worker pool to have GPUs: and use the reduction

server container image.

B. Configure the machines of the first two worker pools to have GPUs and to use a container image where your training code runs. Configure the third worker pool to use the reductionserver container image without accelerators, and choose a machine type that prioritizes bandwidth.

C. Configure the machines of the first two worker pools to have TPUs and to use a container image where your training code runs Configure the third worker pool without accelerators, and use the reductionserver container image without accelerators and choose a machine type that prioritizes bandwidth.

D. Configure the machines of the first two pools to have TPUs. and to use a container image where your training code runs Configure the third pool to have TPUs: and use the reductionserver container image.

Answer: B

Explanation:

According to the web search results, Reduction Server is a faster GPU all-reduce algorithm developed at Google that uses a dedicated set of reducers to aggregate gradients from workers12. Reducers are lightweight CPU VM instances that are significantly cheaper than GPU VMs2. Therefore, the third worker pool should not have any accelerators, and should use a machine type that has high network bandwidth to optimize the communication between workers and reducers2. TPUs are not supported by Reduction Server, so the first two worker pools should have GPUs and use a container image that contains the training code12. The reduction-server container image is provided by Google and should be used for the third worker pool2.

Question: 191

You have trained a model by using data that was preprocessed in a batch Dataflow pipeline Your use case requires real-time inference. You want to ensure that the data preprocessing logic is applied consistently between training and serving. What should you do?

A. Perform data validation to ensure that the input data to the pipeline is the same format as the input data to the endpoint.

B. Refactor the transformation code in the batch data pipeline so that it can be used outside of the pipeline Use the same code in the endpoint.

C. Refactor the transformation code in the batch data pipeline so that it can be used outside of the pipeline Share this code with the end users of the endpoint.

D. Batch the real-time requests by using a time window and then use the Dataflow pipeline to preprocess the batched requests. Send the preprocessed requests to the endpoint.

Answer: B

Explanation:

According to the official exam guide1, one of the skills assessed in the exam is to “design, build, and productionalize ML models to solve business challenges using Google Cloud technologies”. Dataflow2 is a fully managed, fast, and easy-to-use service for running Apache Spark and Apache Hadoop clusters on Google Cloud. Dataflow supports both batch and streaming data processing pipelines. However, if your use

case requires real-time inference, you need to ensure that
the data preprocessing logic is applied consistently between training and serving. One way to achieve this is to refactor the transformation code in the batch data pipeline so that it can be used outside of the pipeline, and use the same code in the endpoint. This way, you can avoid data skew and drift issues that might arise from using different preprocessing methods for training and serving. Therefore, option B is the best way to ensure the data preprocessing logic is applied consistently between training and serving. The other options are not relevant or optimal for this scenario. Reference:
Professional ML Engineer Exam Guide
Dataflow
Google Professional Machine Learning Certification Exam 2023
Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

Question: 192

You need to develop a custom TensorRow model that will be used for online predictions. The training data is stored in BigQuery. You need to apply instance-level data transformations to the data for model training and serving. You want to use the same preprocessing routine during model training and serving. How should you configure the preprocessing routine?

A. Create a BigQuery script to preprocess the data, and write the result to another BigQuery table.
B. Create a pipeline in Vertex Al Pipelines to read the data from BigQuery and preprocess it using a custom preprocessing component.
C. Create a preprocessing function that reads and transforms the data from BigQuery Create a Vertex Al custom prediction routine that calls the preprocessing function at serving time.
D. Create an Apache Beam pipeline to read the data from BigQuery and preprocess it by using TensorFlow Transform and Dataflow.

Answer: D

Explanation:
According to the official exam guide1, one of the skills assessed in the exam is to “design, build, and productionalize ML models to solve business challenges using Google Cloud
technologies”. TensorFlow Transform2 is a library for preprocessing data with TensorFlow. TensorFlow Transform enables you to define and execute distributed pre-processing or feature engineering functions on large data sets, and then export the same functions as a TensorFlow graph for re-use during training or serving. TensorFlow Transform can handle both instance-level and full-pass data transformations. Apache Beam3 is an open source framework for building scalable and portable data pipelines. Apache Beam supports both batch and streaming data processing. Dataflow4 is a fully managed service for running Apache Beam pipelines on Google Cloud. Dataflow handles the provisioning and management of the compute resources, as well as the optimization and execution of the pipelines. Therefore, option D is the best way to configure the preprocessing routine for the given use case, as it allows you to use the same preprocessing logic during model training and serving, and leverage the scalability and performance of Dataflow. The other options are not relevant or optimal for this scenario. Reference: Professional ML Engineer Exam Guide
TensorFlow Transform
Apache Beam
Dataflow
Google Professional Machine Learning Certification Exam 2023
Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

www.atmicnetworks.com

Question: 193

You are pre-training a large language model on Google Cloud. This model includes custom TensorFlow operations in the training loop Model training will use a large batch size, and you expect training to take several weeks You need to configure a training architecture that minimizes both training time and compute costs What should you do?

A.

Implement 8 workers of a2-megagpu-16g machines by using tf.distribute.MultiWorkerMirroredStrategy

B.

Implement a TPU Pod slice with -accelerator-type=v4-128 by using tf.distribute.TPUStrategy

C.

implement 16 workers of o2d-highcpu-32 machines by using tf. distribute.Miirarezstrategy.

D.

Implement 16 workers of a2-highgpu-8g machines by using tf.distribute .MultiWorkerMirroredStrategy

A. Option A B. Option B C. Option C D. Option D

Answer: D

Explanation:

According to the official exam guide1, one of the skills assessed in the exam is to "design, build, and productionalize ML models to solve business challenges using Google Cloud technologies". TPUs2 are Google's custom-developed application-specific integrated circuits (ASICs) used to accelerate machine learning workloads. TPUs are designed to handle large batch sizes, high dimensional data, and complex computations. TPUs can significantly reduce the training time and compute costs of

large language models, especially when used with distributed training strategies, such as MultiWorkerMirroredStrategy3. Therefore, option D is the best way to configure a training architecture that minimizes both training time and compute costs for the given use case. The other options are not relevant or optimal for this scenario. Reference:

Professional ML Engineer Exam Guide

TPUs

MultiWorkerMirroredStrategy

Google Professional Machine Learning Certification Exam 2023

Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

www.atmicnetworks.com

Question: 194

You are building a TensorFlow text-to-image generative model by using a dataset that contains billions of images with their respective captions. You want to create a low maintenance, automated workflow that reads the data from a Cloud Storage bucket collects statistics, splits the dataset into training/validation/test datasets performs data transformations, trains the model using the training/validation datasets. and validates the model by using the test dataset. What should you do?

A. Use the Apache Airflow SDK to create multiple operators that use Dataflow and Vertex Al services Deploy the workflow on Cloud Composer.

B. Use the MLFlow SDK and deploy it on a Google Kubernetes Engine Cluster Create multiple components that use Dataflow and Vertex Al services.

C. Use the Kubeflow Pipelines (KFP) SDK to create multiple components that use Dataflow and Vertex Al services Deploy the workflow on Vertex Al Pipelines.

D. Use the TensorFlow Extended (TFX) SDK to create multiple components that use Dataflow and Vertex Al services Deploy the workflow on Vertex Al Pipelines.

Answer: D

Explanation:

According to the web search results, TensorFlow Extended (TFX) is a platform for building end-to-end machine learning pipelines using TensorFlow1. TFX provides a set of components that can be orchestrated using either the TFX SDK or Kubeflow Pipelines. TFX components can handle different aspects of the pipeline, such as data ingestion, data validation, data transformation, model training, model evaluation, model serving, and more. TFX components can also leverage other Google Cloud services, such as Dataflow2 and Vertex AI3. Dataflow is a fully managed service for running Apache Beam pipelines on Google Cloud. Dataflow handles the provisioning and management of the compute resources, as well as the optimization and execution of the pipelines. Vertex AI is a unified platform for machine learning development and deployment. Vertex AI offers various services and tools for building, managing, and serving machine learning models. Therefore, option D is the best way to create a low maintenance, automated workflow for the given use case, as it allows you to use the TFX SDK to define and execute your pipeline components, and use Dataflow and Vertex AI services to scale and optimize your pipeline. The other options are not relevant or optimal for this scenario. Reference:

TensorFlow Extended

Dataflow

Vertex AI

Google Professional Machine Learning Certification Exam 2023

Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

Question: 195

You are developing an ML pipeline using Vertex Al Pipelines. You want your pipeline to upload a new version of the XGBoost model to Vertex Al Model Registry and deploy it to Vertex Al End points for online inference. You want to use the simplest approach. What should you do?

A. Use the Vertex Al REST API within a custom component based on a vertex-ai/prediction/xgboost- cpu image.

B. Use the Vertex Al ModelEvaluationOp component to evaluate the model.

C. Use the Vertex Al SDK for Python within a custom component based on a python: 3.10 Image.

D. Chain the Vertex AI ModelUploadOp and ModelDeployop components together.

Answer: D

Explanation:
According to the web search results, Vertex AI Pipelines is a serverless orchestrator for running ML pipelines, using either the KFP SDK or TFX1. Vertex AI Pipelines provides a set of prebuilt components that can be used to perform common ML tasks, such as training, evaluation, deployment, and more2. Vertex AI ModelUploadOp and ModelDeployOp are two such components that can be used to upload a new version of the XGBoost model to Vertex AI Model Registry and deploy it to Vertex AI Endpoints for online inference3. Therefore, option D is the best way to use the simplest approach for the given use case, as it only requires chaining two prebuilt components together. The other options are not relevant or optimal for this scenario. Reference:
Vertex AI Pipelines
Google Cloud Pipeline Components
Vertex AI ModelUploadOp and ModelDeployOp
Google Professional Machine Learning Certification Exam 2023
Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

Question: 196

You work for an online retailer. Your company has a few thousand short lifecycle products. Your company has five years of sales data stored in BigQuery. You have been asked to build a model that will make monthly sales predictions for each product. You want to use a solution that can be implemented quickly with minimal effort. What should you do?

A. Use Prophet on Vertex AI Training to build a custom model.
B. Use Vertex AI Forecast to build a NN-based model.
C. Use BigQuery ML to build a statistical AR1MA_PLUS model.
D. Use TensorFlow on Vertex AI Training to build a custom model.

Answer: C

Explanation:
According to the web search results, BigQuery ML1 is a service that allows you to create and execute machine learning models in BigQuery using SQL queries. BigQuery ML supports various types of models, such as linear regression, logistic regression, k-means clustering, matrix factorization, deep neural networks, and time series forecasting1. ARIMA_PLUS2 is a statistical model for time series forecasting that is built in to BigQuery ML. ARIMA_PLUS stands for AutoRegressive Integrated Moving Average with eXogenous regressors. ARIMA_PLUS models the relationship between a target variable and its past values, as well as other external factors that might influence the target variable. ARIMA_PLUS can handle multiple time series, seasonality, holidays, and missing values2. Therefore, option C is the best way to use a solution that can be implemented quickly with minimal effort for the given use case, as it allows you to use SQL queries to build and run a forecasting model in BigQuery without moving the data or writing custom code. The other options are not relevant or optimal for this scenario. Reference: BigQuery ML ARIMA_PLUS
Google Professional Machine Learning Certification Exam 2023
Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

www.atmicnetworks.com

Question: 197

You are creating a model training pipeline to predict sentiment scores from text-based product reviews. You want to have control over how the model parameters are tuned, and you will deploy the model to an endpoint after it has been trained You will use Vertex AI Pipelines to run the pipeline You need to decide which Google Cloud pipeline components to use What components should you choose?

A.

```
TabularDatasetCreateOp, CustomTrainingJobOp, and EndpointCreatsOp
```

B.

```
TextDatasetCreateOp, AutoMLTextTrainingCp, and EndpointCreateOp
```

C.

```
TabularDatasetCreateOp, AutoMLTextTrainingOp, and ModelDeplcyOp
```

D.

```
TextDatasetCreateOp CustomTrainingJobOp and ModelDeployOp
```

A. Option A
B. Option B
C. Option C
D. Option D

Answer: A

Explanation:
According to the web search results, Vertex AI Pipelines is a serverless orchestrator for running ML pipelines, using either the KFP SDK or TFX1. Vertex AI Pipelines provides a set of prebuilt components that can be used to perform common ML tasks, such as training, evaluation, deployment, and more2. Vertex AI ModelEvaluationOp and ModelDeployOp are two such components that can be used to evaluate and deploy a model to an endpoint for online inference3. However, Vertex AI Pipelines does not provide a prebuilt component for hyperparameter tuning. Therefore, to have control over how the model parameters are tuned, you need to use a custom component that calls the Vertex AI HyperparameterTuningJob service4. Therefore, option A is the best way to decide which Google Cloud pipeline components to use for the given use case, as it includes a custom component for hyperparameter tuning, and prebuilt components for model evaluation and deployment. The other options are not relevant or optimal for this scenario.
Reference:
Vertex AI Pipelines
Google Cloud Pipeline Components
Vertex AI ModelEvaluationOp and ModelDeployOp
Vertex AI HyperparameterTuningJob

www.atmicnetworks.com

Question: 198

Your team frequently creates new ML models and runs experiments. Your team pushes code to a single repository hosted on Cloud Source Repositories. You want to create a continuous integration pipeline that automatically retrains the models whenever there is any modification of the code. What should be your first step to set up the CI pipeline?

A. Configure a Cloud Build trigger with the event set as "Pull Request"
B. Configure a Cloud Build trigger with the event set as "Push to a branch"
C. Configure a Cloud Function that builds the repository each time there is acode change.
D. Configure a Cloud Function that builds the repository each time a new branch is created.

Answer: B

Explanation:
According to the web search results, Cloud Build1 is a service that executes your builds on Google Cloud Platform infrastructure. Cloud Build can import source code from Cloud Source Repositories2, Cloud Storage, GitHub, Bitbucket, or any publicly hosted Git repository. Cloud Build allows you to create and manage build triggers, which are automated workflows that run whenever a code change is pushed to your source repository. You can use Cloud Build triggers to automatically retrain your ML
models whenever there is any modification of the code. Therefore, option B is the best way to set up the CI pipeline for the given use case, as it allows you to configure a Cloud Build trigger with the event set as “Push to a branch”, which means the trigger will run whenever a new commit is pushed to a specific branch of your source repository. The other options are not relevant or optimal for this scenario. Reference:
Cloud Build
Cloud Source Repositories
Google Professional Machine Learning Certification Exam 2023
Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

Question: 199

You have built a custom model that performs several memory-intensive preprocessing tasks before it makes a prediction. You deployed the model to a Vertex AI endpoint. and validated that results were received in a reasonable amount of time After routing user traffic to the endpoint, you discover that the endpoint does not autoscale as expected when receiving multiple requests What should you do?

A. Use a machine type with more memory
B. Decrease the number of workers per machine
C. Increase the CPU utilization target in the autoscaling configurations
D. Decrease the CPU utilization target in the autoscaling configurations

Answer: D

Explanation:
According to the web search results, Vertex AI is a unified platform for machine learning development and deployment.

Vertex AI offers various services and tools for building, managing, and serving machine learning models1. Vertex AI allows you to deploy your models to endpoints for online prediction, and configure the compute resources and autoscaling options for your deployed models2. Autoscaling with Vertex AI endpoints is (by default) based on the CPU utilization across all cores of the machine type you have specified. The default threshold of 60% represents 60% on all cores. For example, for a 4 core machine, that means you need 240% utilization to trigger autoscaling3. Therefore, if you discover that the endpoint does not autoscale as expected when receiving multiple requests, you might need to decrease the CPU utilization target in the autoscaling configurations. This way, you can lower the threshold for triggering autoscaling and allocate more resources to handle the prediction requests. Therefore, option D is the best way to solve the problem for the given use case. The other options are not relevant or optimal for this scenario.

Reference: Vertex AI

Deploy a model to an endpoint

Vertex AI endpoint doesn't scale up / down

Google Professional Machine Learning Certification Exam 2023

Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

Question: 200

Your company manages an ecommerce website. You developed an ML model that recommends additional products to users in near real time based on items currently in the user's cart. The workflow will include the following processes.

1 The website will send a Pub/Sub message with the relevant data and then receive a message with the prediction from Pub/Sub.

2 Predictions will be stored in BigQuery

3 . The model will be stored in a Cloud Storage bucket and will be updated frequently

You want to minimize prediction latency and the effort required to update the model How should you reconfigure the architecture?

A. Write a Cloud Function that loads the model into memory for prediction Configure the function to be triggered when messages are sent to Pub/Sub.

B. Create a pipeline in Vertex Al Pipelines that performs preprocessing, prediction and postprocessing Configure the pipeline to be triggered by a Cloud Function when messages are sent to Pub/Sub.

C. Expose the model as a Vertex Al endpoint Write a custom DoFn in a Dataflow job that calls the endpoint for prediction.

D. Use the RunInference API with watchFilePatterr. in a Dataflow job that wraps around the model and serves predictions.

Answer: D

Explanation:

According to the web search results, RunInference API1 is a feature of Apache Beam that enables you to run models as part of your pipeline in a way that is optimized for machine learning inference. RunInference API supports features like batching, caching, and model reloading. RunInference API can be used with various frameworks, such as TensorFlow, PyTorch, Sklearn, XGBoost, ONNX, and TensorRT1. Dataflow2 is a fully managed service for running Apache Beam pipelines on Google Cloud. Dataflow handles the provisioning and management of the compute resources, as well as the optimization and execution of the pipelines. Therefore, option D is the best way to reconfigure the architecture for the given use case, as it allows you to use the RunInference API with watchFilePattern in a Dataflow job that wraps around the model and serves predictions. This way, you can minimize prediction latency and the effort required to update the model, as the RunInference API will automatically reload the model from the Cloud Storage bucket

whenever there is a change in the model file1. The other options are not relevant or optimal for this scenario.

Reference:

RunInference API

Dataflow

Google Professional Machine Learning Certification Exam 2023

Latest Google Professional Machine Learning Engineer Actual Free Exam Questions

Question: 201

You are collaborating on a model prototype with your team. You need to create a Vertex AI Workbench environment for the members of your team and also limit access to other employees in your project. What should you do?

A. 1. Create a new service account and grant it the Notebook Viewer role.
2 Grant the Service Account User role to each team member on the service account.
3 Grant the Vertex AI User role to each team member.
4 . Provision a Vertex AI Workbench user-managed notebook instance that uses the new service account.

B. 1. Grant the Vertex AI User role to the default Compute Engine service account.
2. Grant the Service Account User role to each team member on the default Compute Engine service account.
3. Provision a Vertex AI Workbench user-managed notebook instance that uses the default Compute Engine service account.

C. 1 Create a new service account and grant it the Vertex AI User role.
2 Grant the Service Account User role to each team member on the service account.
3 . Grant the Notebook Viewer role to each team member.
4 Provision a Vertex AI Workbench user-managed notebook instance that uses the new service account.

D. 1 Grant the Vertex AI User role to the primary team member.
2. Grant the Notebook Viewer role to the other team members.
3. Provision a Vertex AI Workbench user-managed notebook instance that uses the primary user's account.

Answer: C

Explanation:

To create a Vertex AI Workbench environment for your team and limit access to other employees in your project, you should follow these steps:

Create a new service account and grant it the Vertex AI User role. This role grants full access to all resources in Vertex AI, including creating and managing notebook instances1.

Grant the Service Account User role to each team member on the service account. This role allows the team members to impersonate the service account and use its permissions2.

Grant the Notebook Viewer role to each team member. This role allows the team members to view and connect to the notebook instance, but not to modify or delete it3.

Provision a Vertex AI Workbench user-managed notebook instance that uses the new service account. This way, the notebook instance will run as the service account and only the team members who have the Service Account User and Notebook Viewer roles will be able to access it.

Reference:

1: Vertex AI access control with IAM | Google Cloud
2: Understanding service accounts | Cloud IAM Documentation
3: Manage access to a Vertex AI Workbench instance | Google Cloud
4:]: Create and manage Vertex AI Workbench instances | Google Cloud

www.atmicnetworks.com

Question: 202

You work at a leading healthcare firm developing state-of-the-art algorithms for various use cases

You have unstructured textual data with custom labels You need to extract and classify various medical phrases with these labels What should you do?

A. Use the Healthcare Natural Language API to extract medical entities.
B. Use a BERT-based model to fine-tune a medical entity extraction model.
C. Use AutoML Entity Extraction to train a medical entity extraction model.
D. Use TensorFlow to build a custom medical entity extraction model.

Answer: B

Explanation:

Medical entity extraction is a task that involves identifying and classifying medical terms or concepts from unstructured textual data, such as electronic health records, clinical notes, or research papers. Medical entity extraction can help with various use cases, such as information retrieval, knowledge discovery, decision support, and data analysis1.

One possible approach to perform medical entity extraction is to use a BERT-based model to finetune a medical entity extraction model. BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language model that can capture the contextual information from both left and right sides of a given token2. BERT can be fine-tuned on a specific downstream task, such as medical entity extraction, by adding a task-specific layer on top of the pre-trained model and updating the model parameters with a small amount of labeled data3.

A BERT-based model can achieve high performance on medical entity extraction by leveraging the large-scale pre-training on general-domain corpora and the fine-tuning on domain-specific data. For example, Nesterov and Umerenkov4 proposed a novel method of doing medical entity extraction from electronic health records as a single-step multi-label classification task by fine-tuning a transformer model pre-trained on a large EHR dataset. They showed that their model can achieve human-level quality for most frequent entities.

Reference:

1: Medical Named Entity Recognition from Un-labelled Medical Records based on Pre-trained Language Models and Domain Dictionary | Data Intelligence | MIT Press
2: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
3: Fine-tuning BERT for Medical Entity Extraction
4: Distantly supervised end-to-end medical entity extraction from electronic health records with human-level quality

Question: 203

You developed a custom model by using Vertex AI to predict your application's user churn rate You are using Vertex AI Model Monitoring for skew detection The training data stored in BigQuery contains two sets of features - demographic and behavioral You later discover that two separate models trained on each set perform better than the original model You need to configure a new model mentioning pipeline that splits traffic among the two models You want to use the same prediction-sampling-rate and monitoring-frequency for each model You also want to minimize management effort What should you do?

A. Keep the training dataset as is Deploy the models to two separate endpoints and submit two Vertex AI Model Monitoring jobs with appropriately selected feature-thresholds parameters B. Keep the training dataset as is Deploy both models to the same endpoint and submit a Vertex AI Model Monitoring job with a monitoring-config-from parameter that accounts for the model IDs and
feature selections

C. Separate the training dataset into two tables based on demographic and behavioral features Deploy the models to two separate endpoints, and submit two Vertex Al Model Monitoring jobs D. Separate the training dataset into two tables based on demographic and behavioral features. Deploy both models to the same endpoint and submit a Vertex Al Model Monitoring job with a monitoring-config-from parameter that accounts for the model IDs and training datasets

Answer: D

Explanation:

Option A is incorrect because it does not separate the training dataset into two tables based on the features, which is necessary to train the two models separately and accurately.

Option B is incorrect because it does not separate the training dataset into two tables based on the features, and because it uses the same monitoring-config-from parameter for both models, which would not account for the different feature selections.

Option C is incorrect because it deploys the models to two separate endpoints, which would increase the management effort and complexity of the pipeline.

Option D is correct because it separates the training dataset into two tables based on the features, which would enable the two models to be trained separately and accurately. It also deploys both models to the same endpoint, which would simplify the pipeline and reduce the management effort. It also submits a Vertex Al Model Monitoring job with a monitoring-config-from parameter that accounts for the model IDs and training datasets, which would enable the skew detection to work properly for each model.

Question: 204

You work for a pharmaceutical company based in Canad

a. Your team developed a BigQuery ML model to predict the number of flu infections for the next month in Canada Weather data is published weekly and flu infection statistics are published monthly.

You need to configure a model retraining policy that minimizes cost What should you do?

A. Download the weather and flu data each week Configure Cloud Scheduler to execute a Vertex Al pipeline to retrain the model weekly.

B. Download the weather and flu data each month Configure Cloud Scheduler to execute a Vertex Al pipeline to retrain the model monthly.

C. Download the weather and flu data each week Configure Cloud Scheduler to execute a Vertex Al pipeline to retrain the model every month.

D. Download the weather data each week, and download the flu data each month Deploy the model to a Vertex Al endpoint with feature drift monitoring. and retrain the model if a monitoring alert is detected.

Answer: D

Explanation:

To configure a model retraining policy that minimizes cost, you should follow these steps:

Download the weather data each week, and download the flu data each month. This way, you can keep your data up to date with the latest information available, without downloading unnecessary or redundant data.

Deploy the model to a Vertex AI endpoint with feature drift monitoring. This feature allows you to detect when the distribution of the input data changes significantly from the training data, which could affect the model performance1.

Retrain the model if a monitoring alert is detected. This way, you can update your model only when needed, instead of retraining it on a fixed schedule, which could incur more cost and time.

Reference:

1: Monitor models for feature drift | Vertex AI | Google Cloud

Question: 205

You are building a MLOps platform to automate your company's ML experiments and model retraining. You need to organize the artifacts for dozens of pipelines How should you store the pipelines' artifacts'?

A. Store parameters in Cloud SQL and store the models' source code and binaries in GitHub

B. Store parameters in Cloud SQL store the models' source code in GitHub, and store the models' binaries in Cloud Storage.

C. Store parameters in Vertex ML Metadata store the models' source code in GitHub and store the models' binaries in Cloud Storage.

D. Store parameters in Vertex ML Metadata and store the models source code and binaries in GitHub.

Answer: C

Explanation:

To organize the artifacts for dozens of pipelines, you should store the parameters in Vertex ML Metadata, store the models' source code in GitHub, and store the models' binaries in Cloud Storage. This option has the following advantages:

Vertex ML Metadata is a service that helps you track and manage the metadata of your ML workflows, such as datasets, models, metrics, and parameters1. It can also help you with data lineage, model versioning, and model performance monitoring2.

GitHub is a popular platform for hosting and collaborating on code repositories. It can help you manage the source code of your models, as well as the configuration files, scripts, and notebooks that are part of your ML pipelines3.

Cloud Storage is a scalable and durable object storage service that can store any type of data, including model binaries4. It can also integrate with other services, such as Vertex AI, Cloud Functions, and Cloud Run, to enable easy deployment and serving of your models5.

Reference:

1: Introduction to Vertex ML Metadata | Vertex AI | Google Cloud

2: Manage metadata for ML workflows | Vertex AI | Google Cloud

3: GitHub - Where the world builds software

4: Cloud Storage | Google Cloud

5: Deploying models | Vertex AI | Google Cloud

Question: 206

You work for a telecommunications company You're building a model to predict which customers may fail to pay their next phone bill. The purpose of this model is to proactively offer at-risk customers assistance such as service discounts and bill deadline extensions. The data is stored in BigQuery, and the predictive features that are available for model training include

- Customer_id -Age
- Salary (measured in local currency) -Sex

-Average bill value (measured in local currency)

- Number of phone calls in the last month (integer) -Average duration of phone calls (measured in minutes)

You need to investigate and mitigate potential bias against disadvantaged groups while preserving model accuracy What should you do?

A. Determine whether there is a meaningful correlation between the sensitive features and the other features Train a BigQuery ML boosted trees classification model and exclude the sensitive features and any meaningfully correlated features

B. Train a BigQuery ML boosted trees classification model with all features Use the ml. global explain method to calculate the global attribution values for each feature of the model If the feature importance value for any of the sensitive features exceeds a threshold, discard the model and tram without this feature

C. Train a BigQuery ML boosted trees classification model with all features Use the ml. exflain_predict method to calculate the attribution values for each feature for each customer in a test set If for any individual customer the importance value for any feature exceeds a predefined threshold, discard the model and train the model again without this feature.

D. Define a fairness metric that is represented by accuracy across the sensitive features Train a BigQuery ML boosted trees classification model with all features Use the trained model to make predictions on a test set Join the data back with the sensitive features, and calculate a fairness metric to investigate whether it meets your requirements.

Answer: D

Explanation:

A fairness metric is a way to measure how well a machine learning model treats different groups of customers, such as by sex or age. A common fairness metric is accuracy, which is the proportion of correct predictions among all predictions. Accuracy across the sensitive features means calculating the accuracy for each group separately, and then comparing them. For example, if the model has 90% accuracy for male customers and 80% accuracy for female customers, there is a 10% accuracy gap that indicates potential bias against female customers.

To investigate and mitigate potential bias, it is important to define a fairness metric and evaluate it on a test set. A test set is a subset of the data that is not used for training the model, but only for evaluating its performance. By joining the test set predictions with the sensitive features, you can calculate the fairness metric and see if it meets your requirements. For example, you may require that the accuracy gap between any two groups is less than 5%. If the fairness metric does not meet your requirements, you may need to adjust the model or the data to reduce bias.

Option A is not the best answer because excluding the sensitive features and any meaningfully correlated features may not eliminate bias. For example, if salary is correlated with sex, and salary is also a predictive feature for the target variable, excluding both features may reduce the model accuracy and still leave some residual bias. Moreover, excluding features based on correlation may

not capture the complex interactions and dependencies among the features that may affect bias. Option B is not the best answer because using the global attribution values for each feature of the model may not reflect the individual-level impact of the features on the predictions. Global attribution values are calculated by averaging the attribution values across all the data points, and they indicate how important each feature is for the overall model performance. However, they do not show how each feature affects each customer's prediction, which may vary depending on the values of the other features. For example, sex may have a low global attribution value, but it may have a high impact on some customers' predictions, especially if it interacts with other features such as salary or age.

Option C is not the best answer because discarding the model and training the model again without a feature based on a single customer's attribution value may not be a robust or scalable way to mitigate bias. Attribution values are calculated by measuring how much each feature contributes to the prediction for a given data point, and they indicate how sensitive the prediction is to the feature value. However, they do not show how the feature affects the overall fairness metric or the model accuracy. For example, sex may have a high attribution value for a customer, but it may not affect the accuracy gap between the groups. Moreover, discarding and retraining the model based on a single customer's attribution value may not be feasible if there are many customers with high attribution values for

different features.

Question: 207

You recently trained a XGBoost model that you plan to deploy to production for online inference Before sending a predict request to your model's binary you need to perform a simple data preprocessing step This step exposes a REST API that accepts requests in your internal VPC Service Controls and returns predictions You want to configure this preprocessing step while minimizing cost and effort What should you do?

A. Store a pickled model in Cloud Storage Build a Flask-based app packages the app in a custom container image, and deploy the model to Vertex Al Endpoints.

B. Build a Flask-based app. package the app and a pickled model in a custom container image, and deploy the model to Vertex Al Endpoints.

C. Build a custom predictor class based on XGBoost Predictor from the Vertex Al SDK. package it and a pickled model in a custom container image based on a Vertex built-in image, and deploy the model to Vertex Al Endpoints.

D. Build a custom predictor class based on XGBoost Predictor from the Vertex Al SDK and package the handler in a custom container image based on a Vertex built-in container image Store a pickled model in Cloud Storage and deploy the model to Vertex Al Endpoints.

Answer: D

Explanation:

Option A is not the best answer because it requires storing the pickled model in Cloud Storage, which may incur additional cost and latency for loading the model. It also requires building a Flask-based app, which may not be necessary for a simple data preprocessing step.

Option B is not the best answer because it requires building a Flask-based app, which may not be necessary for a simple data preprocessing step. It also requires packaging the app and the pickled model in a custom container image, which may increase the size and complexity of the image.

Option C is not the best answer because it requires packaging the pickled model in a custom container image, which may increase the size and complexity of the image. It also does not leverage the Vertex built-in container image, which may provide some optimizations and integrations for XGBoost models.

Option D is the best answer because it leverages the Vertex built-in container image, which may provide some optimizations and integrations for XGBoost models. It also allows storing the pickled model in Cloud Storage, which may reduce the size and complexity of the image. It also allows building a custom predictor class based on XGBoost Predictor from the Vertex AI SDK, which may simplify the data preprocessing step and the prediction logic.

Question: 208

You work at a bank. You need to develop a credit risk model to support loan application decisions You decide to implement the model by using a neural network in TensorFlow Due to regulatory requirements, you need to be able to explain the models predictions based on its features When the model is deployed, you also want to monitor the model's performance overtime You decided to use Vertex Al for both model development and deployment What should you do?

A. Use Vertex Explainable Al with the sampled Shapley method, and enable Vertex Al Model Monitoring to check for feature distribution drift.

B. Use Vertex Explainable AI with the sampled Shapley method, and enable Vertex AI Model Monitoring to check for feature distribution skew.

C. Use Vertex Explainable AI with the XRAI method, and enable Vertex AI Model Monitoring to check for feature distribution drift.

D. Use Vertex Explainable AI with the XRAI method and enable Vertex AI Model Monitoring to check for feature distribution skew.

Answer: A

Explanation:

To develop a credit risk model that meets the regulatory requirements and can be monitored over time, you should follow these steps:

Use Vertex Explainable AI with the sampled Shapley method. Vertex Explainable AI is a service that provides feature attributions for machine learning models, which can help you understand how each feature contributes to the prediction1. The sampled Shapley method is a technique that estimates the Shapley values for each feature, which are based on the marginal contribution of each feature to the prediction across all possible feature subsets2. The sampled Shapley method is suitable for neural networks and other complex models, as it can capture the non-linear and interaction effects of the features3.

Enable Vertex AI Model Monitoring to check for feature distribution drift. Vertex AI Model Monitoring is a service that helps you track and manage the performance and quality of your deployed models over time4. Feature distribution drift is a type of data drift that occurs when the distribution of the input features changes significantly from the training data, which can affect the model accuracy and reliability. By checking for feature distribution drift, you can detect when your model needs to be retrained or updated with new data.

Reference:

1: Introduction to Vertex Explainable AI | Vertex AI | Google Cloud
2: Shapley value - Wikipedia
3: Explainable AI: Interpreting, Explaining and Visualizing Deep Learning
4: Introduction to Vertex AI Model Monitoring | Vertex AI | Google Cloud [5]: Monitor models for data drift | Vertex AI | Google Cloud

Question: 209

You are investigating the root cause of a misclassification error made by one of your models. You used Vertex AI Pipelines to tram and deploy the model. The pipeline reads data from BigQuery. creates a copy of the data in Cloud Storage in TFRecord format trains the model in Vertex AI Training on that copy, and deploys the model to a Vertex AI endpoint. You have identified the specific version of that model that misclassified: and you need to recover the data this model was trained on. How should you find that copy of the data'?

A. Use Vertex AI Feature Store Modify the pipeline to use the feature store; and ensure that all training data is stored in it Search the feature store for the data used for the training.

B. Use the lineage feature of Vertex AI Metadata to find the model artifact Determine the version of the model and identify the step that creates the data copy, and search in the metadata for its location.

C. Use the logging features in the Vertex AI endpoint to determine the timestamp of the models deployment Find the pipeline run at that timestamp Identify the step that creates the data copy; and search in the logs for its location.

D. Find the job ID in Vertex AI Training corresponding to the training for the model Search in the logs of that job for the data used for the training.

www.atmicnetworks.com

Answer: B

Explanation:

Option A is not the best answer because it requires modifying the pipeline to use the Vertex AI Feature Store, which may not be feasible or necessary for recovering the data that the model was trained on. The Vertex AI Feature Store is a service that helps you manage, store, and serve feature values for your machine learning models1, but it is not designed for storing the raw data or the TFRecord files.

Option B is the best answer because it leverages the lineage feature of Vertex AI Metadata, which is a service that helps you track and manage the metadata of your machine learning workflows, such as datasets, models, metrics, and parameters2. The lineage feature allows you to view the relationships and dependencies among the artifacts and executions in your pipeline, and trace back the origin and history of any artifact3. By using the lineage feature, you can find the model artifact, determine the version of the model, identify the step that creates the data copy, and search in the metadata for its location.

Option C is not the best answer because it relies on the logging features in the Vertex AI endpoint, which may not be accurate or reliable for finding the data copy. The logging features in the Vertex AI endpoint help you monitor and troubleshoot the online predictions made by your deployed models, but they do not provide information about the training data or the pipeline steps4. Moreover, the timestamp of the model deployment may not match the timestamp of the pipeline run, as there may

be delays or errors in the deployment process.

Option D is not the best answer because it requires finding the job ID in Vertex AI Training, which may not be easy or straightforward. Vertex AI Training is a service that helps you train your custom models on Google Cloud, but it does not provide a direct way to link the training job to the model version or the pipeline run. Moreover, searching in the logs of the job may not reveal the location of the data copy, as the logs may only contain information about the training process and the metrics. Reference:

1: Introduction to Vertex AI Feature Store | Vertex AI | Google Cloud
2: Introduction to Vertex AI Metadata | Vertex AI | Google Cloud
3: View lineage for ML workflows | Vertex AI | Google Cloud
4: Monitor online predictions | Vertex AI | Google Cloud [5]: Train custom models | Vertex AI | Google Cloud

Question: 210

You work for a manufacturing company. You need to train a custom image classification model to detect product defects at the end of an assembly line Although your model is performing well some images in your holdout set are consistently mislabeled with high confidence You want to use Vertex AI to understand your model's results What should you do?

A.

Configure feature-based explanations by using Integrated Gradients Set visualization type to PIXELS. and set clip_percent_upperboundto 95

B.

Create an index by using Vertex AI Matching Engine Query the index with your mislabeled images

C.

Configure feature-based explanations by using XRAI Set visualization type to OUTLINES and set polarity to positive

D.

Configure example-based explanations Specify the embedding output layer to be used for the latent space representation

Answer: C

Explanation:

Vertex Explainable AI is a set of tools and frameworks to help you understand and interpret predictions made by your machine learning models, natively integrated with a number of Google's products and services1. With Vertex Explainable AI, you can generate feature-based explanations

that show how much each input feature contributed to the model's prediction2. This can help you debug and improve your model performance, and build confidence in your model's behavior. Feature-based explanations are supported for custom image classification models deployed on Vertex AI Prediction3. Reference:

Explainable AI | Google Cloud

Introduction to Vertex Explainable AI | Vertex AI | Google Cloud

Supported model types for feature-based explanations | Vertex AI | Google Cloud

Question: 211

You are using Keras and TensorFlow to develop a fraud detection model Records of customer transactions are stored in a large table in BigQuery. You need to preprocess these records in a costeffective and efficient way before you use them to train the model. The trained model will be used to perform batch inference in BigQuery. How should you implement the preprocessing workflow?

A. Implement a preprocessing pipeline by using Apache Spark, and run the pipeline on Dataproc Save the preprocessed data as CSV files in a Cloud Storage bucket.

B. Load the data into a pandas DataFrame Implement the preprocessing steps using panda's transformations. and train the model directly on the DataFrame.

C. Perform preprocessing in BigQuery by using SQL Use the BigQueryClient in TensorFlow to read the data directly from BigQuery.

D. Implement a preprocessing pipeline by using Apache Beam, and run the pipeline on Dataflow Save the preprocessed data as CSV files in a Cloud Storage bucket.

Answer: C

Explanation:

Option A is not the best answer because it requires using Apache Spark and Dataproc, which may incur additional cost and complexity for running and managing the cluster. It also requires saving the preprocessed data as CSV files in a Cloud Storage bucket, which may increase the storage cost and the data transfer latency.

Option B is not the best answer because it requires loading the data into a pandas DataFrame, which may not be scalable or efficient for large datasets. It also requires training the model directly on the DataFrame, which may not leverage the distributed computing capabilities of BigQuery.

Option C is the best answer because it allows performing preprocessing in BigQuery by using SQL, which is a cost-

effective and efficient way to manipulate large datasets. It also allows using the BigQueryClient in TensorFlow to read the data directly from BigQuery, which is a convenient and fast way to access the data for training the model1. Option D is not the best answer because it requires using Apache Beam and Dataflow, which may incur additional cost and complexity for running and managing the pipeline. It also requires saving the preprocessed data as CSV files in a Cloud Storage bucket, which may increase the storage cost and the data transfer latency.

Reference:

1: Read data from BigQuery | TensorFlow I/O

Question: 212

You are training models in Vertex AI by using data that spans across multiple Google Cloud Projects You need to find track, and compare the performance of the different versions of your models Which Google Cloud services should you include in your ML workflow?

A. Dataplex. Vertex AI Feature Store and Vertex AI TensorBoard
B. Vertex AI Pipelines, Vertex AI Feature Store, and Vertex AI Experiments
C. Dataplex. Vertex AI Experiments, and Vertex AI ML Metadata
D. Vertex AI Pipelines: Vertex AI Experiments and Vertex AI Metadata

Answer: B

Explanation:

Vertex AI Pipelines is a service that allows you to orchestrate and automate your machine learning (ML) workflows using pipelines1. A pipeline is a description of an ML workflow, including all of the components in the workflow, how the components are connected as a graph, and the runtime parameters that the pipeline accepts1. Vertex AI Pipelines helps you manage the end-to-end lifecycle of your ML projects, from data preprocessing to model deployment1.

Vertex AI Feature Store is a service that enables you to serve, share, and reuse ML features across different models and projects2. A feature is a measurable property or characteristic of an entity, such as the age of a person or the price of a product2. Vertex AI Feature Store helps you reduce data duplication, ensure data consistency, and improve model performance2.

Vertex AI Experiments is a service that helps you track and compare the performance of different versions of your models3. You can use Vertex AI Experiments to run multiple training jobs with different hyperparameters, architectures, or data sources, and then compare the results using metrics, visualizations, and reports3. Vertex AI Experiments helps you identify the best model for your use case and optimize your model performance3. Reference:

Vertex AI Pipelines | Google Cloud

Vertex AI Feature Store | Google Cloud

Vertex AI Experiments | Google Cloud

Question: 213

You need to use TensorFlow to train an image classification model. Your dataset is located in a Cloud Storage directory and contains millions of labeled images Before training the model, you need to prepare the dat

a. You want the data preprocessing and model training workflow to be as efficient scalable, and low maintenance as possible. What should you do?

A. 1 Create a Dataflow job that creates sharded TFRecord files in a Cloud Storage directory.
2 Reference tf .data.TFRecordDataset in the training script.
3 . Train the model by using Vertex AI Training with a V100 GPU.

B. 1 Create a Dataflow job that moves the images into multiple Cloud Storage directories, where each directory is

named according to the corresponding label.

2 Reference tfds.fclder_da-asst.imageFclder in the training script.

3. Train the model by using Vertex AI Training with a V100 GPU.

C. 1 Create a Jupyter notebook that uses an n1-standard-64, V100 GPU Vertex Al Workbench instance.

2. Write a Python script that creates sharded TFRecord files in a directory inside the instance

3. Reference tf. da-a.TFRecrrdDataset in the training script.

4. Train the model by using the Workbench instance.

D. 1 Create a Jupyter notebook that uses an n1-standard-64, V100 GPU Vertex Al Workbench instance.

2 Write a Python scnpt that copies the images into multiple Cloud Storage directories, where each directory is named according to the corresponding label.

3 Reference tf ds. f older_dataset. imageFolder in the training script.

4 . Train the model by using the Workbench instance.

Answer: A

Explanation:

TFRecord is a binary file format that stores your data as a sequence of binary strings1. TFRecord files are efficient, scalable, and easy to process1. Sharding is a technique that splits a large file into smaller files, which can improve parallelism and performance2. Dataflow is a service that allows you to create and run data processing pipelines on Google Cloud3. Dataflow can create sharded TFRecord files from your images in a Cloud Storage directory4.

tf.data.TFRecordDataset is a class that allows you to read and parse TFRecord files in TensorFlow. You can use this class to create a tf.data.Dataset object that represents your input data for training.

tf.data.Dataset is a high-level API that provides various methods to transform, batch, shuffle, and prefetch your data.

Vertex AI Training is a service that allows you to train your custom models on Google Cloud using various hardware accelerators, such as GPUs. Vertex AI Training supports TensorFlow models and can read data from Cloud Storage. You can use Vertex AI Training to train your image classification model by using a V100 GPU, which is a powerful and fast GPU for deep learning.

Reference:

TFRecord and tf.Example | TensorFlow Core

Sharding | TensorFlow Core

Dataflow | Google Cloud

Creating sharded TFRecord files | Google Cloud [tf.data.TFRecordDataset | TensorFlow Core v2.6.0] [tf.data: Build TensorFlow input pipelines | TensorFlow Core] [Vertex AI Training | Google Cloud] [NVIDIA Tesla V100 GPU | NVIDIA]

Question: 214

You are building a custom image classification model and plan to use Vertex Al Pipelines to implement the end-to-end training. Your dataset consists of images that need to be preprocessed before they can be used to train the model. The preprocessing steps include resizing the images, converting them to grayscale, and extracting features. You have already implemented some Python functions for the preprocessing tasks. Which components should you use in your pipeline'?

A.

DataprocSparkBatchOp and CustcmTrainingJobOp

B.

DataflowPythonJobOp WaitGcpResourcesOp and Custom!rainingJobOp

C.

dsl.ParallelFor, dsl.component and CustomTrainingJcbOp

D.

ZmageDatasetlmportDataOp. dsl.component, and AutoMllmageTrainingJobRunOp

A. Option A B. Option B C. Option C D. Option D

Answer: B

Explanation:

Question: 215

You work for a retail company that is using a regression model built with BigQuery ML to predict product sales. This model is being used to serve online predictions Recently you developed a new version of the model that uses a different architecture (custom model) Initial analysis revealed that both models are performing as expected You want to deploy the new version of the model to production and monitor the performance over the next two months You need to minimize the impact to the existing and future model users How should you deploy the model?

A. Import the new model to the same Vertex Al Model Registry as a different version of the existing model. Deploy the new model to the same Vertex Al endpoint as the existing model, and use traffic splitting to route 95% of production traffic to the BigQuery ML model and 5% of production traffic to the new model.

B. Import the new model to the same Vertex Al Model Registry as the existing model Deploy the models to one Vertex Al endpoint Route 95% of production traffic to the BigQuery ML model and 5% of production traffic to the new model

C. Import the new model to the same Vertex Al Model Registry as the existing model Deploy each model to a separate Vertex Al endpoint.

D. Deploy the new model to a separate Vertex Al endpoint Create a Cloud Run service that routes the prediction requests to the corresponding endpoints based on the input feature values.

Answer: A

Explanation:

Vertex AI Model Registry is a central repository where you can manage the lifecycle of your ML models1. You can import models from various sources, such as BigQuery ML, AutoML, or custom models, and assign them to different versions and aliases1. You can also deploy models to endpoints, which are resources that provide a service URL for online prediction2.

By importing the new model to the same Vertex AI Model Registry as a different version of the existing model, you can keep track of the model versions and compare their performance metrics1. You can also use aliases to label the model versions according to their readiness for production, such as default or staging1.

By deploying the new model to the same Vertex AI endpoint as the existing model, you can use traffic splitting to gradually shift the production traffic from the old model to the new model2. Traffic splitting is a feature that allows you to specify the percentage of prediction requests that each deployed model in an endpoint should handle2. This way, you can minimize the impact to the existing and future model users, and monitor the performance of the new model over time2.

The other options are not suitable for your scenario, because they either require creating a separate endpoint or a Cloud Run service, which would increase the complexity and maintenance of your deployment, or they do not allow you to use traffic splitting, which would create a sudden change in your prediction results. Reference:

Introduction to Vertex AI Model Registry | Google Cloud

Deploy a model to an endpoint | Vertex AI | Google Cloud

Question: 216

You work for a large retailer and you need to build a model to predict customer churn. The company has a dataset of historical customer data, including customer demographics, purchase history, and website activity. You need to create the model in BigQuery ML and thoroughly evaluate its performance. What should you do?

A. Create a linear regression model in BigQuery ML and register the model in Vertex Al Model Registry Evaluate the model performance in Vertex Al.

B. Create a logistic regression model in BigQuery ML and register the model in Vertex Al Model Registry. Evaluate the model performance in Vertex Al.

C. Create a linear regression model in BigQuery ML Use the ml. evaluate function to evaluate the model performance.

D. Create a logistic regression model in BigQuery ML Use the ml.confusion_matrix function to evaluate the model performance.

Answer: B

Explanation:

Customer churn is a binary classification problem, where the target variable is whether a customer has churned or not. Therefore, a logistic regression model is more suitable than a linear regression model, which is used for regression problems. A logistic regression model can output the probability of a customer churning, which can be used to rank the customers by their churn risk and take appropriate actions1.

BigQuery ML is a service that allows you to create and execute machine learning models in BigQuery using standard SQL queries2. You can use BigQuery ML to create a logistic regression model for customer churn prediction by using the CREATE MODEL statement and specifying the LOGISTIC_REG model type3. You can use the historical customer data as the input table for the model, and specify the features and the label columns3.

Vertex AI Model Registry is a central repository where you can manage the lifecycle of your ML models4. You can import models from various sources, such as BigQuery ML, AutoML, or custom models, and assign them to different versions

and aliases4. You can also deploy models to endpoints, which are resources that provide a service URL for online prediction.

By registering the BigQuery ML model in Vertex AI Model Registry, you can leverage the Vertex AI features to evaluate and monitor the model performance4. You can use Vertex AI Experiments to track and compare the metrics of different model versions, such as accuracy, precision, recall, and AUC. You can also use Vertex AI Explainable AI to generate feature attributions that show how much each input feature contributed to the model's prediction.

The other options are not suitable for your scenario, because they either use the wrong model type, such as linear regression, or they do not use Vertex AI to evaluate the model performance, which would limit the insights and actions you can take based on the model results.

Reference:

Logistic Regression for Machine Learning

Introduction to BigQuery ML | Google Cloud

Creating a logistic regression model | BigQuery ML | Google Cloud

Introduction to Vertex AI Model Registry | Google Cloud

[Deploy a model to an endpoint | Vertex AI | Google Cloud]

[Vertex AI Experiments | Google Cloud]

Question: 217

You are developing a model to identify traffic signs in images extracted from videos taken from the dashboard of a vehicle. You have a dataset of 100 000 images that were cropped to show one out of ten different traffic signs. The images have been labeled accordingly for model training and are stored in a Cloud Storage bucket You need to be able to tune the model during each training run. How should you train the model?

A. Train a model for object detection by using Vertex Al AutoML.

B. Train a model for image classification by using Vertex Al AutoML.

C. Develop the model training code for object detection and tram a model by using Vertex Al custom training.

D. Develop the model training code for image classification and train a model by using Vertex Al custom training.

Answer: D

Explanation:

Image classification is a task where the model assigns a label to an image based on its content, such as "stop sign" or "speed limit"1. Object detection is a task where the model locates and identifies multiple objects in an image, and draws bounding boxes around them2. Since your dataset consists of images that were cropped to show one out of ten different traffic signs, you are dealing with an image classification problem, not an object detection problem. Therefore, you need to train a model for image classification, not object detection.

Vertex AI AutoML is a service that allows you to train and deploy high-quality ML models with minimal effort and machine learning expertise3. You can use Vertex AI AutoML to train a model for image classification by uploading your images and labels to a Vertex AI dataset, and then launching an AutoML training job4. However, Vertex AI AutoML does not allow you to tune the model during each training run, as it automatically selects the best model architecture and hyperparameters for your data4.

Vertex AI custom training is a service that allows you to train and deploy your own custom ML models using your own code and frameworks5. You can use Vertex AI custom training to train a model for image classification by writing your own model training code, such as using TensorFlow or PyTorch, and then creating and running a custom training job. Vertex AI custom training allows you to tune the model during each training run, as you can specify the model architecture and hyperparameters in your code, and use Vertex AI Hyperparameter Tuning to optimize them . Therefore, the best option for your scenario is to develop the model training code for image classification and train a model by using Vertex AI custom training.

Reference:
Image classification | TensorFlow Core
Object detection | TensorFlow Core
Introduction to Vertex AI AutoML | Google Cloud
AutoML Vision | Google Cloud
Introduction to Vertex AI custom training | Google Cloud
[Custom training with TensorFlow | Vertex AI | Google Cloud] [Hyperparameter tuning overview | Vertex AI | Google Cloud]

Question: 218

You have deployed a scikit-learn model to a Vertex AI endpoint using a custom model server. You enabled auto scaling; however, the deployed model fails to scale beyond one replica, which led to dropped requests. You notice that CPU utilization remains low even during periods of high load. What should you do?

A. Attach a GPU to the prediction nodes.
B. Increase the number of workers in your model server.
C. Schedule scaling of the nodes to match expected demand.
D. Increase the minReplicaCount in your DeployedModel configuration.

Answer: B

Explanation:
Auto scaling is a feature that allows you to automatically adjust the number of prediction nodes based on the traffic and load of your deployed model1. However, auto scaling depends on the CPU utilization of your prediction nodes, which is the percentage of CPU resources used by your model server1. If your CPU utilization is low, even during periods of high load, it means that your model server is not fully utilizing the available CPU resources, and thus auto scaling will not trigger more replicas2.
One possible reason for low CPU utilization is that your model server is using a single worker process to handle prediction requests3. A worker process is a subprocess that runs your model code and handles prediction requests3. If you have only one worker process, it can only handle one request at a time, which can lead to dropped requests when the traffic is high3. To increase the CPU utilization and the throughput of your model server, you can increase the number of worker processes, which will allow your model server to handle multiple requests in parallel3.
To increase the number of workers in your model server, you need to modify your custom model server code and use the --workers flag to specify the number of worker processes you want to use3. For example, if you are using a Gunicorn server, you can use the following command to start your model server with four worker processes:
gunicorn --bind :$PORT --workers 4 --threads 1 --timeout 60 main:app
By increasing the number of workers in your model server, you can increase the CPU utilization of your prediction nodes, and thus enable auto scaling to scale beyond one replica.
The other options are not suitable for your scenario, because they either do not address the root cause of low CPU utilization, such as attaching a GPU or scheduling scaling, or they do not enable auto scaling, such as increasing the minReplicaCount, which is a fixed number of nodes that will always run regardless of the traffic1.
Reference:
Scaling prediction nodes | Vertex AI | Google Cloud
Troubleshooting | Vertex AI | Google Cloud
Using a custom prediction routine with online prediction | Vertex AI | Google Cloud

www.atmicnetworks.com

Question: 219

You work for a pet food company that manages an online forum Customers upload photos of their pets on the forum to share with others About 20 photos are uploaded daily You want to automatically and in near real time detect whether each uploaded photo has an animal You want to prioritize time and minimize cost of your application development and deployment What should you do?

A. Send user-submitted images to the Cloud Vision API Use object localization to identify all objects in the image and compare the results against a list of animals.

B. Download an object detection model from TensorFlow Hub. Deploy the model to a Vertex AI endpoint. Send new user-submitted images to the model endpoint to classify whether each photo has an animal.

C. Manually label previously submitted images with bounding boxes around any animals Build an AutoML object detection model by using Vertex AI Deploy the model to a Vertex AI endpoint Send new user-submitted images to your model endpoint to detect whether each photo has an animal. D. Manually label previously submitted images as having animals or not Create an image dataset on Vertex AI Train a classification model by using Vertex AutoML to distinguish the two classes Deploy the model to a Vertex AI endpoint Send new user-submitted images to your model endpoint to classify whether each photo has an animal.

Answer: A

Explanation:

Cloud Vision API is a service that allows you to analyze images using pre-trained machine learning models1. You can use Cloud Vision API to perform various tasks, such as face detection, text extraction, logo recognition, and object localization1. Object localization is a feature that allows you to detect multiple objects in an image and draw bounding boxes around them2. You can also get the labels and confidence scores for each detected object2.

By sending user-submitted images to the Cloud Vision API, you can use object localization to identify all objects in the image and compare the results against a list of animals. You can use the OBJECT_LOCALIZATION feature type in the AnnotateImageRequest to request object localization3. You can then use the localizedObjectAnnotations field in the AnnotateImageResponse to get the list of detected objects, their labels, and their confidence scores. You can compare the labels with a predefined list of animals, such as dogs, cats, birds, etc., and determine whether the image has an animal or not.

This option is the best for your scenario, because it allows you to automatically and in near real time detect whether each uploaded photo has an animal, without requiring any manual labeling, model training, or model deployment. You can also prioritize time and minimize cost of your application development and deployment, as you can use the Cloud Vision API as a ready-to-use service, without needing any machine learning expertise or infrastructure.

The other options are not suitable for your scenario, because they either require manual labeling, model training, or model deployment, which would increase the time and cost of your application development and deployment, or they use object detection models, which are more complex and computationally expensive than object localization models, and are not necessary for your simple task of detecting whether an image has an animal or not.

Reference:

Cloud Vision API | Google Cloud

Object localization | Cloud Vision API | Google Cloud

AnnotateImageRequest | Cloud Vision API | Google Cloud [AnnotateImageResponse | Cloud Vision API | Google Cloud]

Question: 220

You work at a mobile gaming startup that creates online multiplayer games Recently, your company observed an increase in players cheating in the games, leading to a loss of revenue and a poor user experience. You built a binary

classification model to determine whether a player cheated after a completed game session, and then send a message to other downstream systems to ban the player that cheated Your model has performed well during testing, and you now need to deploy the model to production You want your serving solution to provide immediate classifications after a completed game session to avoid further loss of revenue. What should you do?

A. Import the model into Vertex Al Model Registry. Use the Vertex Batch Prediction service to run batch inference jobs.

B. Save the model files in a Cloud Storage Bucket Create a Cloud Function to read the model files and make online inference requests on the Cloud Function.

C. Save the model files in a VM Load the model files each time there is a prediction request and run an inference job on the VM.

D. Import the model into Vertex Al Model Registry Create a Vertex Al endpoint that hosts the model and make online inference requests.

Answer: D

Explanation:

Online inference is a process where you send a single or a small number of prediction requests to a model and get immediate responses1. Online inference is suitable for scenarios where you need timely predictions, such as detecting cheating in online games. Online inference requires that the model is deployed to an endpoint, which is a resource that provides a service URL for prediction requests2.

Vertex AI Model Registry is a central repository where you can manage the lifecycle of your ML models3. You can import models from various sources, such as custom models or AutoML models, and assign them to different versions and aliases3. You can also deploy models to endpoints, which are resources that provide a service URL for online prediction2.

By importing the model into Vertex AI Model Registry, you can leverage the Vertex AI features to monitor and update the model3. You can use Vertex AI Experiments to track and compare the metrics of different model versions, such as accuracy, precision, recall, and AUC. You can also use Vertex AI Explainable AI to generate feature attributions that show how much each input feature contributed to the model's prediction.

By creating a Vertex AI endpoint that hosts the model, you can use the Vertex AI Prediction service to serve online inference requests2. Vertex AI Prediction provides various benefits, such as scalability, reliability, security, and logging2. You can use the Vertex AI API or the Google Cloud console to send online inference requests to the endpoint and get immediate classifications4.

Therefore, the best option for your scenario is to import the model into Vertex AI Model Registry, create a Vertex AI endpoint that hosts the model, and make online inference requests.

The other options are not suitable for your scenario, because they either do not provide immediate classifications, such as using batch prediction or loading the model files each time, or they do not use Vertex AI Prediction, which would require more development and maintenance effort, such as creating a Cloud Function or a VM.

Reference:

Online versus batch prediction | Vertex AI | Google Cloud

Deploy a model to an endpoint | Vertex AI | Google Cloud

Introduction to Vertex AI Model Registry | Google Cloud

Get online predictions | Vertex AI | Google Cloud

Question: 221

You have created a Vertex Al pipeline that automates custom model training You want to add a pipeline component

that enables your team to most easily collaborate when running different executions and comparing metrics both visually and programmatically. What should you do?

A. Add a component to the Vertex Al pipeline that logs metrics to a BigQuery table Query the table to compare different executions of the pipeline Connect BigQuery to Looker Studio to visualize metrics. B. Add a component to the Vertex Al pipeline that logs metrics to a BigQuery table Load the table into a pandas DataFrame to compare different executions of the pipeline Use Matplotlib to visualize metrics.

C. Add a component to the Vertex Al pipeline that logs metrics to Vertex ML Metadata Use Vertex Al Experiments to compare different executions of the pipeline Use Vertex Al TensorBoard to visualize metrics.

D. Add a component to the Vertex Al pipeline that logs metrics to Vertex ML Metadata Load the Vertex ML Metadata into a pandas DataFrame to compare different executions of the pipeline. Use Matplotlib to visualize metrics.

Answer: C

Explanation:
Vertex AI Experiments is a managed service that allows you to track, compare, and manage experiments with Vertex AI. You can use Vertex AI Experiments to record the parameters, metrics, and artifacts of each pipeline run, and compare them in a graphical interface. Vertex AI TensorBoard is a tool that lets you visualize the metrics of your models, such as accuracy, loss, and learning curves. By logging metrics to Vertex ML Metadata and using Vertex AI Experiments and TensorBoard, you can easily collaborate with your team and find the best model configuration for your problem. Reference: Vertex AI Pipelines: Metrics visualization and run comparison using the KFP SDK, Track, compare, manage experiments with Vertex AI Experiments, Vertex AI Pipelines

Question: 222

Your team is training a large number of ML models that use different algorithms, parameters and datasets. Some models are trained in Vertex Ai Pipelines, and some are trained on Vertex Al Workbench notebook instances. Your team wants to compare the performance of the models across both services. You want to minimize the effort required to store the parameters and metrics What should you do?

A. Implement an additional step for all the models running in pipelines and notebooks to export parameters and metrics to BigQuery.

B. Create a Vertex Al experiment Submit all the pipelines as experiment runs. For models trained on notebooks log parameters and metrics by using the Vertex Al SDK.

C. Implement all models in Vertex Al Pipelines Create a Vertex Al experiment, and associate all pipeline runs with that experiment.

D. Store all model parameters and metrics as mode! metadata by using the Vertex Al Metadata API.

Answer: B

Explanation:
Vertex AI Experiments is a service that allows you to track, compare, and manage experiments with Vertex AI. You can use Vertex AI Experiments to record the parameters, metrics, and artifacts of each model training run, and compare them in a graphical interface. Vertex AI Experiments supports models trained in Vertex AI Pipelines, Vertex AI Custom Training, and Vertex AI Workbench notebooks. To use Vertex AI Experiments, you need to create an experiment and submit your pipeline runs or custom training jobs as experiment runs. For models trained on notebooks, you need to use the Vertex AI SDK to log the parameters and metrics to the experiment. This way, you can minimize the effort required to store and compare the model performance across different services. Reference: Track, compare, manage

experiments with Vertex AI Experiments, Vertex AI

Pipelines: Metrics visualization and run comparison using the KFP SDK, [Vertex AI SDK for Python]

Question: 223

You work on a team that builds state-of-the-art deep learning models by using the TensorFlow framework. Your team runs multiple ML experiments each week which makes it difficult to track the experiment runs. You want a simple approach to effectively track, visualize and debug ML experiment runs on Google Cloud while minimizing any overhead code. How should you proceed?

A. Set up Vertex AI Experiments to track metrics and parameters Configure Vertex AI TensorBoard for visualization.
B. Set up a Cloud Function to write and save metrics files to a Cloud Storage Bucket Configure a Google Cloud VM to host TensorBoard locally for visualization.
C. Set up a Vertex AI Workbench notebook instance Use the instance to save metrics data in a Cloud Storage bucket and to host TensorBoard locally for visualization.
D. Set up a Cloud Function to write and save metrics files to a BigQuery table. Configure a Google Cloud VM to host TensorBoard locally for visualization.

Answer: A

Explanation:
Vertex AI Experiments is a service that allows you to track, compare, and optimize your ML experiments on Google Cloud. You can use Vertex AI Experiments to log metrics and parameters from your TensorFlow models, and then visualize them in Vertex AI TensorBoard. Vertex AI TensorBoard is a managed service that provides a web interface for viewing and debugging your ML experiments. You can use Vertex AI TensorBoard to compare different runs, inspect model graphs, analyze scalars, histograms, images, and more. By using Vertex AI Experiments and Vertex AI TensorBoard, you can simplify your ML experiment tracking and visualization workflow, and avoid the overhead of setting up and maintaining your own Cloud Functions, Cloud Storage buckets, or VMs. Reference:
[Vertex AI Experiments documentation]
[Vertex AI TensorBoard documentation]
Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate

Question: 224

Your work for a textile manufacturing company. Your company has hundreds of machines and each machine has many sensors. Your team used the sensory data to build hundreds of ML models that detect machine anomalies Models are retrained daily and you need to deploy these models in a costeffective way. The models must operate 24/7 without downtime and make sub millisecond predictions. What should you do?

A. Deploy a Dataflow batch pipeline and a Vertex AI Prediction endpoint.
B. Deploy a Dataflow batch pipeline with the RunInference API. and use model refresh.
C. Deploy a Dataflow streaming pipeline and a Vertex AI Prediction endpoint with autoscaling.
D. Deploy a Dataflow streaming pipeline with the RunInference API and use automatic model refresh.

Answer: D

Explanation:

A Dataflow streaming pipeline is a cost-effective way to process large volumes of real-time data from sensors. The RunInference API is a Dataflow transform that allows you to run online predictions on your streaming data using your ML models. By using the RunInference API, you can avoid the latency and cost of using a separate prediction service. The automatic model refresh feature enables you to update your models in the pipeline without redeploying the pipeline. This way, you can ensure that your models are always up-to-date and accurate. By deploying a Dataflow streaming pipeline with the RunInference API and using automatic model refresh, you can achieve sub-millisecond predictions, 24/7 availability, and low operational overhead for your ML models. Reference: Dataflow documentation

RunInference API documentation

Automatic model refresh documentation

Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate

Question: 225

You are developing an ML model that predicts the cost of used automobiles based on data such as location, condition model type color, and engine-'battery efficiency. The data is updated every night Car dealerships will use the model to determine appropriate car prices. You created a Vertex Al pipeline that reads the data splits the data into training/evaluation/test sets performs feature engineering trains the model by using the training dataset and validates the model by using the evaluation dataset. You need to configure a retraining workflow that minimizes cost What should you do?

A. Compare the training and evaluation losses of the current run If the losses are similar, deploy the model to a Vertex AI endpoint Configure a cron job to redeploy the pipeline every night.

B. Compare the training and evaluation losses of the current run If the losses are similar deploy the model to a Vertex Al endpoint with training/serving skew threshold model monitoring When the model monitoring threshold is tnggered redeploy the pipeline.

C. Compare the results to the evaluation results from a previous run If the performance improved deploy the model to a Vertex Al endpoint Configure a cron job to redeploy the pipeline every night. D. Compare the results to the evaluation results from a previous run If the performance improved deploy the model to a Vertex Al endpoint with training/serving skew threshold model monitoring. When the model monitoring threshold is triggered, redeploy the pipeline.

Answer: B

Explanation:

Comparing the training and evaluation losses of the current run is a good way to check if the model is overfitting or underfitting. If the losses are similar, it means that the model is generalizing well and can be deployed to a Vertex AI endpoint. Vertex AI endpoint is a service that allows you to serve your ML models online and scale them automatically. By using a training/serving skew threshold model monitoring, you can detect if there is a significant difference between the data used for training and

the data used for serving. This can indicate that the model is becoming stale or inaccurate over time. When the model monitoring threshold is triggered, it means that the model needs to be retrained with the latest data. By redeploying the pipeline, you can automate the retraining process and update the model with the new data. This way, you can minimize the cost of retraining and ensure that your model is always up-to-date and accurate. Reference:

Vertex AI documentation

Vertex AI endpoint documentation

Model monitoring documentation

Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate

www.atmicnetworks.com

Question: 226

You recently used BigQuery ML to train an AutoML regression model. You shared results with your team and received positive feedback. You need to deploy your model for online prediction as quickly as possible. What should you do?

A. Retrain the model by using BigQuery ML. and specify Vertex Al as the model registry Deploy the model from Vertex Al Model Registry to a Vertex Al endpoint.

B. Retrain the model by using Vertex Al Deploy the model from Vertex Al Model Registry to a Vertex Al endpoint.

C. Alter the model by using BigQuery ML and specify Vertex Al as the model registry Deploy the model from Vertex Al Model Registry to a Vertex Al endpoint.

D. Export the model from BigQuery ML to Cloud Storage Import the model into Vertex Al Model Registry Deploy the model to a Vertex Al endpoint.

Answer: A

Explanation:

BigQuery ML is a service that allows you to create and train ML models using SQL queries. You can use BigQuery ML to train an AutoML regression model, which is a type of model that automatically selects the best features and architecture for your data. You can also specify Vertex AI as the model registry, which is a service that allows you to store and manage your ML models. By using Vertex AI as the model registry, you can easily deploy your model to a Vertex AI endpoint, which is a service that allows you to serve your ML models online and scale them automatically. By using BigQuery ML, Vertex AI model registry, and Vertex AI endpoint, you can deploy your model for online prediction as quickly as possible, without having to export, import, or retrain your model. Reference:

BigQuery ML documentation

Vertex AI documentation

Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate

Question: 227

You built a deep learning-based image classification model by using on-premises dat
a. You want to use Vertex Al to deploy the model to production Due to security concerns you cannot move your data to the cloud. You are aware that the input data distribution might change over time You need to detect model performance changes in production. What should you do?

A. Use Vertex Explainable Al for model explainability Configure feature-based explanations.

B. Use Vertex Explainable Al for model explainability Configure example-based explanations.

C. Create a Vertex Al Model Monitoring job. Enable training-serving skew detection for your model.

D. Create a Vertex Al Model Monitoring job. Enable feature attribution skew and dnft detection for your model.

Answer: C

Explanation:

Vertex AI Model Monitoring is a service that allows you to monitor the performance and quality of your ML models in production. You can use Vertex AI Model Monitoring to detect changes in the input data distribution, the prediction

output distribution, or the model accuracy over time. Trainingserving skew detection is a feature of Vertex AI Model Monitoring that compares the statistics of the data used for training the model and the data used for serving the model. If there is a significant difference between the two data distributions, it indicates that the model might be outdated or inaccurate. By enabling training-serving skew detection for your model, you can detect model performance changes in production and trigger retraining or redeployment of your model as needed. This way, you can ensure that your model is always up-to-date and accurate, without moving your data to the cloud. Reference:

Vertex AI Model Monitoring documentation

Training-serving skew detection documentation

Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate

Question: 228

You trained a model, packaged it with a custom Docker container for serving, and deployed it to Vertex Al Model Registry. When you submit a batch prediction job, it fails with this error "Error model server never became ready Please validate that your model file or container configuration are valid. There are no additional errors in the logs What should you do?

A. Add a logging configuration to your application to emit logs to Cloud Logging.

B. Change the HTTP port in your model's configuration to the default value of 8080

C. Change the health Route value in your models configuration to /heal thcheck.

D. Pull the Docker image locally and use the decker run command to launch it locally. Use the docker logs command to explore the error logs.

Answer: B

Explanation:

When you deploy a custom container to Vertex AI Model Registry, you need to follow some requirements for the container configuration. One of these requirements is to use the HTTP port 8080 for serving predictions. If you use a different port, the model server might not be able to communicate with Vertex AI and cause the error “Error model server never became ready”. To fix this error, you need to change the HTTP port in your model’s configuration to the default value of 8080 and redeploy the container. Reference:

Custom container requirements documentation

Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate

Question: 229

You are developing an ML model to identify your company s products in images. You have access to over one million images in a Cloud Storage bucket. You plan to experiment with different TensorFlow models by using Vertex Al Training You need to read images at scale during training while minimizing data I/O bottlenecks What should you do?

A. Load the images directly into the Vertex Al compute nodes by using Cloud Storage FUSE Read the images by using

the tf .data.Dataset.from_tensor_slices function.

B. Create a Vertex Al managed dataset from your image data Access the aip_training_data_uri environment variable to read the images by using the tf. data. Dataset. list_flies function.

C. Convert the images to TFRecords and store them in a Cloud Storage bucket Read the TFRecords by using the tf. ciata.TFRecordDataset function.

D. Store the URLs of the images in a CSV file Read the file by using the tf.data.experomental.CsvDataset function.

Answer: C

Explanation:

TFRecords are a binary file format that can store large amounts of data efficiently. By converting the images to TFRecords and storing them in a Cloud Storage bucket, you can reduce the data size and improve the data transfer speed. You can then read the TFRecords by using the tf.data.TFRecordDataset function, which creates a dataset of tensors from the TFRecord files. This way, you can read images at scale during training while minimizing data I/O bottlenecks. Reference: TFRecord documentation
tf.data.TFRecordDataset documentation
Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate

Question: 230

You work at an ecommerce startup. You need to create a customer churn prediction model Your company's recent sales records are stored in a BigQuery table You want to understand how your initial model is making predictions. You also want to iterate on the model as quickly as possible while minimizing cost How should you build your first model?

A. Export the data to a Cloud Storage Bucket Load the data into a pandas DataFrame on Vertex Al Workbench and train a logistic regression model with scikit-learn.

B. Create a tf.data.Dataset by using the TensorFlow BigQueryChent Implement a deep neural network in TensorFlow.

C. Prepare the data in BigQuery and associate the data with a Vertex Al dataset Create an AutoMLTabuiarTrainmgJob to train a classification model.

D. Export the data to a Cloud Storage Bucket Create tf. data. Dataset to read the data from Cloud Storage Implement a deep neural network in TensorFlow.

Answer: C

Explanation:

BigQuery is a service that allows you to store and query large amounts of data in a scalable and costeffective way. You can use BigQuery to prepare the data for your customer churn prediction model, such as filtering, aggregating, and transforming the data. You can then associate the data with a Vertex AI dataset, which is a service that allows you to store and manage your ML data on Google Cloud. By using a Vertex AI dataset, you can easily access the data from other Vertex AI services, such as AutoML. AutoML is a service that allows you to create and train ML models without writing code. You can use AutoML to create an AutoMLTabularTrainingJob, which is a type of job that trains a

classification model for tabular data, such as customer churn. By using an AutoMLTabularTrainingJob, you can benefit from the automated feature engineering, model selection, and hyperparameter tuning that AutoML provides. You can also use Vertex Explainable AI to understand how your model is making predictions, such as which features are most important and how they affect the prediction outcome. By using BigQuery, Vertex AI dataset, and AutoMLTabularTrainingJob, you can build your first model as quickly as possible while minimizing cost and complexity.

Reference:

BigQuery documentation

Vertex AI dataset documentation

AutoMLTabularTrainingJob documentation

Vertex Explainable AI documentation

Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate

Question: 231

You are developing a training pipeline for a new XGBoost classification model based on tabular data The data is stored in a BigQuery table You need to complete the following steps

1 . Randomly split the data into training and evaluation datasets in a 65/35 ratio

2 . Conduct feature engineering

3 Obtain metrics for the evaluation dataset.

4 Compare models trained in different pipeline executions

How should you execute these steps'?

A. 1 Using Vertex Al Pipelines, add a component to divide the data into training and evaluation sets, and add another component for feature engineering

2 . Enable auto logging of metrics in the training component.

3 Compare pipeline runs in Vertex Al Experiments

B. 1 Using Vertex Al Pipelines, add a component to divide the data into training and evaluation sets, and add another component for feature engineering

2 Enable autologging of metrics in the training component

3 Compare models using the artifacts lineage in Vertex ML Metadata

C. 1 In BigQuery ML. use the create model statement with bocstzd_tree_classifier as the model type and use BigQuery to handle the data splits.

2 Use a SQL view to apply feature engineering and train the model using the data in that view

3 . Compare the evaluation metrics of the models by using a SQL query with the ml. training_infc statement.

D. 1 In BigQuery ML use the create model statement with boosted_tree_classifier as the model type, and use BigQuery to handle the data splits.

2 Use ml transform to specify the feature engineering transformations, and train the model using the data in the table

' 3. Compare the evaluation metrics of the models by using a SQL query with the ml. training_info statement.

Answer: B

Explanation:

Vertex AI Pipelines is a service that allows you to create and run scalable and portable ML pipelines on Google Cloud. You can use Vertex AI Pipelines to add a component to divide the data into training and evaluation sets, and add another component for feature engineering. A component is a selfcontained piece of code that performs a specific task in the pipeline. You can use the built-in components provided by Vertex AI Pipelines, or create your own custom components. By using Vertex AI Pipelines, you can orchestrate and automate your ML workflow, and track the provenance and lineage of your data and models. You can also enable autologging of metrics in the training component, which is a feature that automatically logs the metrics from your XGBoost model to Vertex AI Experiments. Vertex AI Experiments is a service that allows you to track, compare, and optimize your ML experiments on Google Cloud. You can use Vertex AI Experiments to monitor the training progress, visualize the metrics, and analyze the results of your model. You can also compare models using the artifacts lineage in Vertex ML Metadata. Vertex ML Metadata is a service that stores and manages the metadata of your ML artifacts, such as datasets, models, metrics, and executions. You can use Vertex ML Metadata to view the artifacts lineage, which is a graph that shows the relationships and dependencies among the artifacts. By using the artifacts lineage, you can compare the performance and quality of different models trained in different pipeline executions, and identify the best model for your use case. By using Vertex AI Pipelines, Vertex AI Experiments, and Vertex ML
Metadata, you can execute the steps required for developing a training pipeline for a new XGBoost classification model based on tabular data stored in a BigQuery table. Reference:

Vertex AI Pipelines documentation
Vertex AI Experiments documentation
Vertex ML Metadata documentation
Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate

Question: 232

You work for a company that sells corporate electronic products to thousands of businesses worldwide. Your company stores historical customer data in BigQuery. You need to build a model that predicts customer lifetime value over the next three years. You want to use the simplest approach to build the model and you want to have access to visualization tools. What should you do?

A. Create a Vertex Al Workbench notebook to perform exploratory data analysis. Use IPython magics to create a new BigQuery table with input features Use the BigQuery console to run the create model statement Validate the results by using the ml. evaluate and ml. predict statements.
B. Run the create model statement from the BigQuery console to create an AutoML model Validate the results by using the ml. evaluate and ml. predict statements.
C. Create a Vertex Al Workbench notebook to perform exploratory data analysis and create input features Save the features as a CSV file in Cloud Storage Import the CSV file as a new BigQuery table Use the BigQuery console to run the create model statement Validate the results by using the ml. evaluate and ml. predict statements.
D. Create a Vertex Al Workbench notebook to perform exploratory data analysis Use IPython magics to create a new BigQuery table with input features, create the model and validate the results by using the create model, ml. evaluates, and ml. predict statements.

Answer: B

Explanation:

BigQuery is a service that allows you to store and query large amounts of data in a scalable and costeffective way. You can use BigQuery to build a model that predicts customer lifetime value over the next three years, by using the create model statement. The create model statement is a SQL command that allows you to create and train an ML model using your data in BigQuery. You can use the create model statement to create an AutoML model, which is a type of model

that automatically selects the best features and architecture for your data. By using an AutoML model, you can use the simplest approach to build the model, without writing any code or performing any feature engineering. You can also use the ml.evaluate and ml.predict statements to validate the results of your model. The ml.evaluate statement is a SQL command that allows you to evaluate the performance and quality of your model using various metrics. The ml.predict statement is a SQL command that allows you to make predictions using your model and new data. You can also use the BigQuery console to access visualization tools, such as charts and graphs, to explore and analyze your data and model results. By using the BigQuery console, the create model statement, and the ml.evaluate and ml.predict statements, you can build and validate a model that predicts customer lifetime value over the next three years, and have access to visualization tools. Reference:

BigQuery documentation
create model statement documentation
ml.evaluate statement documentation
ml.predict statement documentation
Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate

Question: 233

You work at a large organization that recently decided to move their ML and data workloads to Google Cloud. The data engineering team has exported the structured data to a Cloud Storage bucket in Avro format. You need to propose a workflow that performs analytics, creates features, and hosts the features that your ML models use for online prediction How should you configure the pipeline?

A. Ingest the Avro files into Cloud Spanner to perform analytics Use a Dataflow pipeline to create the features and store them in BigQuery for online prediction.
B. Ingest the Avro files into BigQuery to perform analytics Use a Dataflow pipeline to create the features, and store them in Vertex Al Feature Store for online prediction.
C. Ingest the Avro files into BigQuery to perform analytics Use BigQuery SQL to create features and store them in a separate BigQuery table for online prediction.
D. Ingest the Avro files into Cloud Spanner to perform analytics. Use a Dataflow pipeline to create the features. and store them in Vertex Al Feature Store for online prediction.

Answer: B

Explanation:
BigQuery is a service that allows you to store and query large amounts of data in a scalable and costeffective way. You can use BigQuery to ingest the Avro files from the Cloud Storage bucket and perform analytics on the structured data. Avro is a binary file format that can store complex data types and schemas. You can use the bq load command or the BigQuery API to load the Avro files into a BigQuery table. You can then use SQL queries to analyze the data and generate insights. Dataflow is a service that allows you to create and run scalable and portable data processing pipelines on Google Cloud. You can use Dataflow to create the features for your ML models, such as transforming, aggregating, and encoding the data. You can use the Apache Beam SDK to write your Dataflow pipeline code in Python or Java. You can also use the built-in transforms or custom transforms to apply the feature engineering logic to your data. Vertex AI Feature Store is a service that allows you to store and manage your ML features on Google Cloud. You can use Vertex AI Feature Store to host the features that your ML models use for online prediction. Online prediction is a type of prediction that provides low-latency responses to individual or small batches of input data. You can use the Vertex AI Feature Store API to write the features from your Dataflow pipeline to a feature store entity type. You can then use the Vertex AI Feature Store online serving API to read the features from the feature store and pass them to your ML models for online prediction. By using BigQuery, Dataflow, and Vertex AI Feature Store, you can configure a pipeline that

performs analytics, creates features, and hosts the features that your ML models use for online prediction. Reference:
BigQuery documentation Dataflow documentation Vertex AI Feature Store documentation
Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate

Question: 234

You work for a delivery company. You need to design a system that stores and manages features such as parcels delivered and truck locations over time. The system must retrieve the features with low latency and feed those features into a model for online prediction. The data science team will retrieve historical data at a specific point in time for model training. You want to store the features with minimal effort. What should you do?

A. Store features in Bigtable as key/value data.
B. Store features in Vertex Al Feature Store.
C. Store features as a Vertex Al dataset and use those features to tram the models hosted in Vertex Al endpoints.
D. Store features in BigQuery timestamp partitioned tables, and use the BigQuery Storage Read API to serve the features.

Answer: B

Explanation:
Vertex AI Feature Store is a service that allows you to store and manage your ML features on Google Cloud. You can use Vertex AI Feature Store to store features such as parcels delivered and truck locations over time, and retrieve them with low latency for online prediction. Online prediction is a type of prediction that provides low-latency responses to individual or small batches of input data. You can also use Vertex AI Feature Store to retrieve historical data at a specific point in time for model training. Model training is a process of learning the parameters of a ML model from data. By using Vertex AI Feature Store, you can store the features with minimal effort, and avoid the complexity of managing your own data storage and serving system. Reference:
Vertex AI Feature Store documentation
Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate

Question: 235

You are working on a prototype of a text classification model in a managed Vertex AI Workbench notebook. You want to quickly experiment with tokenizing text by using a Natural Language Toolkit (NLTK) library. How should you add the library to your Jupyter kernel?

A. Install the NLTK library from a terminal by using the pip install nltk command.
B. Write a custom Dataflow job that uses NLTK to tokenize your text and saves the output to Cloud Storage.
C. Create a new Vertex Al Workbench notebook with a custom image that includes the NLTK library.
D. Install the NLTK library from a Jupyter cell by using the! pip install nltk —user command.

Answer: D

Explanation:
NLTK is a Python library that provides a set of tools for natural language processing, such as tokenization, stemming, tagging, parsing, and sentiment analysis. Tokenization is a process of
breaking a text into smaller units, such as words or sentences. You can use NLTK to quickly experiment with tokenizing text in a managed Vertex AI Workbench notebook. A Vertex AI Workbench notebook is a web-based interactive

environment that allows you to write and execute Python code on Google Cloud. You can install the NLTK library from a Jupyter cell by using the !pip install nltk --user command. This command uses the pip package manager to install the NLTK library for the current user. By installing the NLTK library from a Jupyter cell, you can avoid the hassle of opening a terminal or creating a custom image for your notebook. Reference:

NLTK documentation
Vertex AI Workbench documentation
Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate

Question: 236

You have recently used TensorFlow to train a classification model on tabular data You have created a Dataflow pipeline that can transform several terabytes of data into training or prediction datasets consisting of TFRecords. You now need to productionize the model, and you want the predictions to be automatically uploaded to a BigQuery table on a weekly schedule. What should you do?

A. Import the model into Vertex Al and deploy it to a Vertex Al endpoint On Vertex Al Pipelines create a pipeline that uses the Dataf lowPythonJobop and the Mcdei3archPredictoc components.
B. Import the model into Vertex Al and deploy it to a Vertex Al endpoint Create a Dataflow pipeline that reuses the data processing logic sends requests to the endpoint and then uploads predictions to a BigQuery table.
C. Import the model into Vertex Al On Vertex Al Pipelines, create a pipeline that uses the DataflowPythonJobOp and the ModelBatchPredictOp components.
D. Import the model into BigQuery Implement the data processing logic in a SQL query On Vertex Al Pipelines create a pipeline that uses the BigqueryQueryJobop and the EigqueryPredictModejobOp components.

Answer: C

Explanation:
Vertex AI is a service that allows you to create and train ML models using Google Cloud technologies. You can use Vertex AI to import the model that you trained with TensorFlow and store it in the Vertex AI Model Registry. The Vertex AI Model Registry is a service that allows you to store and manage your ML models on Google Cloud. You can then use Vertex AI Pipelines to create a pipeline that uses the DataflowPythonJobOp and the ModelBatchPredictOp components. The DataflowPythonJobOp component is a component that allows you to run a Dataflow job using a Python script. Dataflow is a service that allows you to create and run scalable and portable data processing pipelines on Google Cloud. You can use the DataflowPythonJobOp component to reuse the data processing logic that you created for transforming the data into TFRecords. The ModelBatchPredictOp component is a component that allows you to run a batch prediction job using a model from the Vertex AI Model Registry. Batch prediction is a type of prediction that provides high-throughput responses to large batches of input data. You can use the ModelBatchPredictOp component to make predictions using the TFRecords from the DataflowPythonJobOp component and the model from the Vertex AI Model Registry. You can also configure the ModelBatchPredictOp component to automatically upload the predictions to a BigQuery table. BigQuery is a service that
allows you to store and query large amounts of data in a scalable and cost-effective way. You can use BigQuery to store and analyze the predictions from your model. You can also schedule the pipeline to run on a weekly basis, so that the predictions are updated regularly. By using Vertex AI, Vertex AI Pipelines, Dataflow, and BigQuery, you can productionize the model and upload the predictions to a BigQuery table on a weekly schedule. Reference:

Vertex AI documentation
Vertex AI Pipelines documentation
Dataflow documentation
BigQuery documentation

Question: 237

You work for an online grocery store. You recently developed a custom ML model that recommends a recipe when a user arrives at the website. You chose the machine type on the Vertex AI endpoint to optimize costs by using the queries per second (QPS) that the model can serve, and you deployed it on a single machine with 8 vCPUs and no accelerators.

A holiday season is approaching and you anticipate four times more traffic during this time than the typical daily traffic You need to ensure that the model can scale efficiently to the increased demand. What should you do?

A. 1, Maintain the same machine type on the endpoint.
2 Set up a monitoring job and an alert for CPU usage
3 If you receive an alert add a compute node to the endpoint

B. 1 Change the machine type on the endpoint to have 32 vCPUs
2 . Set up a monitoring job and an alert for CPU usage
3 If you receive an alert, scale the vCPUs further as needed

C. 1 Maintain the same machine type on the endpoint Configure the endpoint to enable autoscalling based on vCPU usage.
2 Set up a monitoring job and an alert for CPU usage
3 If you receive an alert investigate the cause

D. 1 Change the machine type on the endpoint to have a GPU_ Configure the endpoint to enable autoscaling based on the GPU usage.
2 Set up a monitoring job and an alert for GPU usage.
3 If you receive an alert investigate the cause.

Answer: C

Explanation:

Vertex AI Endpoint is a service that allows you to serve your ML models online and scale them automatically. You can use Vertex AI Endpoint to deploy the custom ML model that you developed for recommending recipes to the users. You can maintain the same machine type on the endpoint, which is a single machine with 8 vCPUs and no accelerators. This machine type can optimize the costs by using the queries per second (QPS) that the model can serve. You can also configure the endpoint to enable autoscaling based on vCPU usage. Autoscaling is a feature that allows the endpoint to adjust the number of compute nodes based on the traffic demand. By enabling autoscaling based on vCPU usage, you can ensure that the endpoint can scale efficiently to the increased demand during the holiday season, without overprovisioning or underprovisioning the resources. You can also set up a monitoring job and an alert for CPU usage. Monitoring is a service that allows you to collect and analyze the metrics and logs from your Google Cloud resources. You can use Monitoring to monitor the CPU usage of your endpoint, which is an indicator of the load and performance of your model. You can also set up an alert for CPU usage, which is a feature that allows you to receive notifications when the CPU usage exceeds a certain threshold. By setting up a monitoring job and an alert for CPU usage, you can keep track of the health and status of your endpoint, and detect any issues or anomalies. If you receive an alert, you can investigate the cause by using the Monitoring dashboard, which provides a graphical interface for viewing and analyzing the metrics and logs from your endpoint. You can also use the Monitoring dashboard to troubleshoot and resolve the issues, such as adjusting the autoscaling parameters, optimizing the model, or updating the machine type. By using Vertex AI Endpoint, autoscaling, and Monitoring, you can ensure that the model can scale efficiently to the increased demand during the holiday season, and handle any issues or alerts

that might arise. Reference:

[Vertex AI Endpoint documentation]

[Autoscaling documentation] [Monitoring documentation]

[Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate]

Question: 238

You recently trained an XGBoost model on tabular data You plan to expose the model for internal use as an HTTP microservice After deployment you expect a small number of incoming requests. You want to productionize the model with the least amount of effort and latency. What should you do?

A. Deploy the model to BigQuery ML by using CREATE model with the BOOSTED-THREE-REGRESSOR statement and invoke the BigQuery API from the microservice.

B. Build a Flask-based app Package the app in a custom container on Vertex AI and deploy it to Vertex AI Endpoints.

C. Build a Flask-based app Package the app in a Docker image and deploy it to Google Kubernetes Engine in Autopilot mode.

D. Use a prebuilt XGBoost Vertex container to create a model and deploy it to Vertex AI Endpoints.

Answer: D

Explanation:

XGBoost is a popular open-source library that provides a scalable and efficient implementation of gradient boosted trees. You can use XGBoost to train a classification or regression model on tabular data. You can also use Vertex AI to productionize the model and expose it for internal use as an HTTP microservice. Vertex AI is a service that allows you to create and train ML models using Google Cloud technologies. You can use a prebuilt XGBoost Vertex container to create a model and deploy it to Vertex AI Endpoints. A prebuilt Vertex container is a container image that contains the dependencies and libraries needed to run a specific ML framework, such as XGBoost. You can use a prebuilt Vertex container to simplify the model creation and deployment process, without having to build your own custom container. Vertex AI Endpoints is a service that allows you to serve your ML models online and scale them automatically. You can use Vertex AI Endpoints to deploy the model from the prebuilt Vertex container and expose it as an HTTP microservice. You can also configure the endpoint to

handle a small number of incoming requests, and optimize the latency and cost of serving the model. By using a prebuilt XGBoost Vertex container and Vertex AI Endpoints, you can productionize the model with the least amount of effort and latency. Reference:

XGBoost documentation

Vertex AI documentation

Prebuilt Vertex container documentation

Vertex AI Endpoints documentation

Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate

Question: 239

You work for an international manufacturing organization that ships scientific products all over the world Instruction manuals for these products need to be translated to 15 different languages Your organization's leadership team wants to start using machine learning to reduce the cost of manual human translations and increase translation speed. You need to implement a scalable solution that maximizes accuracy and minimizes operational overhead. You also want to

include a process to evaluate and fix incorrect translations. What should you do?

A. Create a workflow using Cloud Function Triggers Configure a Cloud Function that is triggered when documents are uploaded to an input Cloud Storage bucket Configure another Cloud Function that translates the documents using the Cloud Translation API and saves the translations to an output Cloud Storage bucket Use human reviewers to evaluate the incorrect translations.

B. Create a Vertex Al pipeline that processes the documents1 launches an AutoML Translation training job evaluates the translations, and deploys the model to a Vertex Al endpoint with autoscaling and model monitoring When there is a predetermined skew between training and live data re-trigger the pipeline with the latest data.

C. Use AutoML Translation to tram a model Configure a Translation Hub project and use the trained model to translate the documents Use human reviewers to evaluate the incorrect translations D. Use Vertex Al custom training jobs to fine-tune a state-of-the-art open source pretrained model with your data Deploy the model to a Vertex Al endpoint with autoscaling and model monitoring When there is a predetermined skew between the training and live data, configure a trigger to run another training job with the latest data.

Answer: C

Explanation:

AutoML Translation is a service that allows you to create and train custom ML models for translating text between different languages. You can use AutoML Translation to train a model that can translate instruction manuals for scientific products to 15 different languages. You can also use Translation Hub to configure a project and use the trained model to translate the documents. Translation Hub is a service that allows you to manage and automate your translation workflows on Google Cloud. You can use Translation Hub to upload the documents to a Cloud Storage bucket, select the source and target languages, and apply the trained model to translate the documents. You can also use Translation Hub to download the translated documents or save them to another Cloud Storage bucket. You can also use human reviewers to evaluate the incorrect translations. Human reviewers are people who can review and correct the translations produced by the ML model. You can use human reviewers to improve the quality and accuracy of the translations, and provide feedback to

the ML model. You can use Translation Hub to integrate with third-party human review services, such as Google Translate Community or Appen. By using AutoML Translation, Translation Hub, and human reviewers, you can implement a scalable solution that maximizes accuracy and minimizes operational overhead. You can also include a process to evaluate and fix incorrect translations. Reference:

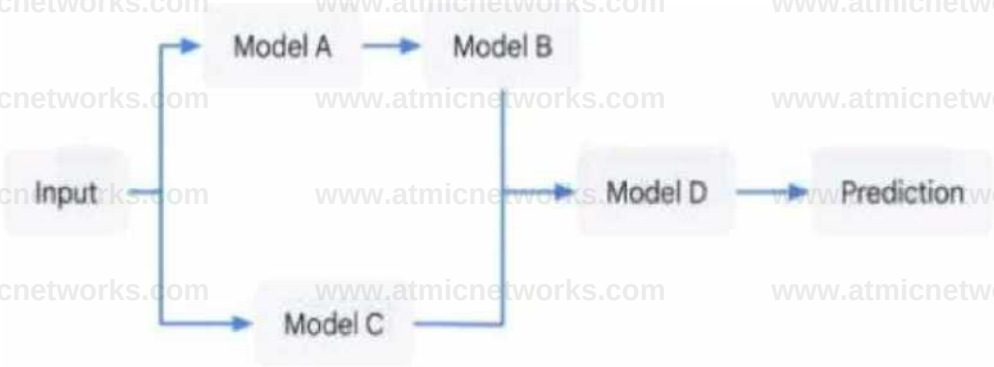
[AutoML Translation documentation]

[Translation Hub documentation]

[Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate]

Question: 240

You have developed an application that uses a chain of multiple scikit-learn models to predict the optimal price for your company's products. The workflow logic is shown in the diagram Members of your team use the individual models in other solution workflows. You want to deploy this workflow while ensuring version control for each individual model and the overall workflow Your application needs to be able to scale down to zero. You want to minimize the compute resource utilization and the manual effort required to manage this solution. What should you do?



A. Expose each individual model as an endpoint in Vertex Al Endpoints. Create a custom container endpoint to orchestrate the workflow.

B. Create a custom container endpoint for the workflow that loads each models individual files Track the versions of each individual model in BigQuery.

C. Expose each individual model as an endpoint in Vertex Al Endpoints. Use Cloud Run to orchestrate the workflow.

D. Load each model's individual files into Cloud Run Use Cloud Run to orchestrate the workflow Track the versions of each individual model in BigQuery.

Answer: C

Explanation:
The option C is the most efficient and scalable solution for deploying a machine learning workflow with multiple models while ensuring version control and minimizing compute resource utilization. By exposing each model as an endpoint in Vertex AI Endpoints, it allows for easy versioning and management of individual models. Using Cloud Run to orchestrate the workflow ensures that the application can scale down to zero, thus minimizing resource utilization when not in use. Cloud Run is a service that allows you to run stateless containers on a fully managed environment or on Google Kubernetes Engine. You can use Cloud Run to invoke the endpoints of each model in the workflow

and pass the data between them. You can also use Cloud Run to handle the input and output of the workflow and provide an HTTP interface for the application. Reference:
Vertex AI Endpoints documentation
Cloud Run documentation
Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate

Question: 241

You are developing a model to predict whether a failure will occur in a critical machine part. You have a dataset consisting of a multivariate time series and labels indicating whether the machine part failed You recently started experimenting with a few different preprocessing and modeling approaches in a Vertex Al Workbench notebook. You want to log data and track artifacts from each run. How should you set up your experiments?

A.

1 Use the Vertex Al SDK to create an experiment and set up Vertex ML Metadata
2 Use the log_time_series_nettics function to track the preprocessed data and use the logjnetrics function to log loss values

B.

1 Use the Vertex Al SDK to create an experiment and set up Vertex ML Metadata

2 Use the log_tiae_«eries_metrics function to track the preprocessed data and use the log_aetrics function to log loss values

C.

1 Create a Vertex Al TensorBoard instance and use the Vertex Al SDK to create an experiment and associate the TensorBoard instance

2 Use the assign_input_artifact method to track the preprocessed data and use the log X1IQ8 Series metrics function to log loss values

D.

1 Create a Vertex Al TensorBoard instance and use the Vertex Al SDK to create an experiment and associate the TensorBoard instance

2 Use the log_tine_series_nietrics function to track the preprocessed data, and use the log_n»trics function to log loss values

A. Option A

B. Option B C. Option C D. Option D

Answer: A

Explanation:

The option A is the most suitable solution for logging data and tracking artifacts from each run of a model development experiment in a Vertex AI Workbench notebook. Vertex AI Workbench is a service that allows you to create and run interactive notebooks on Google Cloud. You can use Vertex AI Workbench to experiment with different preprocessing and modeling approaches for your time series prediction problem. You can also use the Vertex AI TensorBoard instance and the Vertex AI SDK to create an experiment and associate the TensorBoard instance. TensorBoard is a tool that allows you to visualize and monitor the metrics and artifacts of your ML experiments. You can use the Vertex AI SDK to create an experiment object, which is a logical grouping of runs that share a common objective. You can also use the Vertex AI SDK to associate the experiment object with a TensorBoard instance, which is a managed service that hosts a TensorBoard web app. By using the Vertex AI TensorBoard instance and the Vertex AI SDK, you can easily set up and manage your experiments, and access the TensorBoard web app from the Vertex AI console. You can also use the log_time_series_metrics function and the log_metrics function to log data and track artifacts from each run. The log_time_series_metrics function is a function that allows you to log the time series data, such as the multivariate time series and the labels, to the TensorBoard instance. The log_metrics function is a function that allows you to log the scalar metrics, such as the loss values, to the TensorBoard instance. By using these functions, you can record the data and artifacts from each run of your experiment, and compare them in the TensorBoard web app. You can also use the TensorBoard web app to visualize the data and artifacts, such as the time series plots, the scalar charts, the histograms, and the distributions. By using the Vertex AI TensorBoard instance, the Vertex AI SDK, and the log functions, you can log data and track artifacts from each run of your experiment in a Vertex AI Workbench notebook. Reference:

Vertex AI Workbench documentation
Vertex AI TensorBoard documentation
Vertex AI SDK documentation
log_time_series_metrics function documentation log_metrics function documentation
[Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate]

Question: 242

You are developing a recommendation engine for an online clothing store. The historical customer transaction data is stored in BigQuery and Cloud Storage. You need to perform exploratory data analysis (EDA), preprocessing and model training. You plan to rerun these EDA, preprocessing, and training steps as you experiment with different types of algorithms. You want to minimize the cost and development effort of running these steps as you experiment. How should you configure the environment?

A. Create a Vertex AI Workbench user-managed notebook using the default VM instance, and use the %%bigquery magic commands in Jupyter to query the tables.

B. Create a Vertex AI Workbench managed notebook to browse and query the tables directly from the JupyterLab interface.

C. Create a Vertex AI Workbench user-managed notebook on a Dataproc Hub. and use the %%bigquery magic commands in Jupyter to query the tables.

D. Create a Vertex AI Workbench managed notebook on a Dataproc cluster, and use the spark-bigquery-connector to access the tables.

Answer: A

Explanation:
Cost-effectiveness: User-managed notebooks in Vertex AI Workbench allow you to leverage preconfigured virtual machines with reasonable resource allocation, keeping costs lower compared to options involving managed notebooks or Dataproc clusters.
Development flexibility: User-managed notebooks offer full control over the environment, allowing you to install additional libraries or dependencies needed for your specific EDA, preprocessing, and model training tasks. This flexibility is crucial while experimenting with different algorithms. BigQuery integration: The %%bigquery magic commands provide seamless integration with BigQuery within the Jupyter Notebook environment. This enables efficient querying and exploration of customer transaction data stored in BigQuery directly from the notebook, streamlining the workflow.
Other options and why they are not the best fit:
B . Managed notebook: While managed notebooks offer an easier setup, they might have limited customization options, potentially hindering your ability to install specific libraries or tools.
C . Dataproc Hub: Dataproc Hub focuses on running large-scale distributed workloads, and it might be overkill for your scenario involving exploratory analysis and experimentation with different algorithms. Additionally, it could incur higher costs compared to a user-managed notebook.
D . Dataproc cluster with spark-bigquery-connector: Similar to option C, using a Dataproc cluster with the spark-bigquery-connector would be more complex and potentially more expensive than using %%bigquery magic commands within a user-managed notebook for accessing BigQuery data.

Reference:
https://cloud.google.com/vertex-ai/docs/workbench/instances/bigquery https://cloud.google.com/vertex-ai-notebooks

www.atmicnetworks.com

Question: 243

You recently deployed a model lo a Vertex AI endpoint and set up online serving in Vertex AI Feature Store. You have configured a daily batch ingestion job to update your featurestore During the batch ingestion jobs you discover that CPU utilization is high in your featurestores online serving nodes and that feature retrieval latency is high. You need to improve online serving performance during the daily batch ingestion. What should you do?

A. Schedule an increase in the number of online serving nodes in your featurestore prior to the batch ingestion jobs.
B. Enable autoscaling of the online serving nodes in your featurestore
C. Enable autoscaling for the prediction nodes of your DeployedModel in the Vertex AI endpoint.
D. Increase the worker counts in the importFeaturevalues request of your batch ingestion job.

Answer: B

Explanation:
Vertex AI Feature Store provides two options for online serving: Bigtable and optimized online serving. Both options support autoscaling, which means that the number of online serving nodes can automatically adjust to the traffic demand. By enabling autoscaling, you can improve the online serving performance and reduce the feature retrieval latency during the daily batch ingestion.
Autoscaling also helps you optimize the cost and resource utilization of your featurestore. Reference: Online serving | Vertex AI | Google Cloud
New Vertex AI Feature Store: BigQuery-Powered, GenAI-Ready | Google Cloud Blog

Question: 244

You are developing a custom TensorFlow classification model based on tabular dat
a. Your raw data is stored in BigQuery contains hundreds of millions of rows, and includes both categorical and numerical features. You need to use a MaxMin scaler on some numerical features, and apply a one-hot encoding to some categorical features such as SKU names. Your model will be trained over multiple epochs. You want to minimize the effort and cost of your solution. What should you do?

A. 1 Write a SQL query to create a separate lookup table to scale the numerical features.
2. Deploy a TensorFlow-based model from Hugging Face to BigQuery to encode the text features.
3. Feed the resulting BigQuery view into Vertex AI Training.
B. 1 Use BigQuery to scale the numerical features.
2 . Feed the features into Vertex AI Training.
3 Allow TensorFlow to perform the one-hot text encoding.
C. 1 Use TFX components with Dataflow to encode the text features and scale the numerical features.
2 Export results to Cloud Storage as TFRecords.
3 Feed the data into Vertex AI Training.
D. 1 Write a SQL query to create a separate lookup table to scale the numerical features.
2 Perform the one-hot text encoding in BigQuery.
3 . Feed the resulting BigQuery view into Vertex AI Training.

Answer: C

Explanation:
TFX (TensorFlow Extended) is a platform for end-to-end machine learning pipelines. It provides components for data

ingestion, preprocessing, validation, model training, serving, and monitoring. Dataflow is a fully managed service for scalable data processing. By using TFX components with Dataflow, you can perform feature engineering on large-scale tabular data in a distributed and efficient way. You can use the Transform component to apply the MaxMin scaler and the one-hot encoding to the numerical and categorical features, respectively. You can also use the ExampleGen component to read data from BigQuery and the Trainer component to train your TensorFlow model. The output of the Transform component is a TFRecord file, which is a binary format for storing TensorFlow data. You can export the TFRecord file to Cloud Storage and feed it into Vertex AI Training, which is a managed service for training custom machine learning models on Google Cloud. Reference:

TFX | TensorFlow
Dataflow | Google Cloud
Vertex AI Training | Google Cloud

Question: 245

You are developing a custom image classification model in Python. You plan to run your training application on Vertex Al Your input dataset contains several hundred thousand small images You need to determine how to store and access the images for training. You want to maximize data throughput and minimize training time while reducing the amount of additional code. What should you do?

A. Store image files in Cloud Storage and access them directly.
B. Store image files in Cloud Storage and access them by using serialized records.
C. Store image files in Cloud Filestore, and access them by using serialized records.
D. Store image files in Cloud Filestore and access them directly by using an NFS mount point.

Answer: B

Explanation:
Cloud Storage is a scalable and cost-effective storage service for any type of data. By storing image files in Cloud Storage, you can access them from anywhere and avoid the overhead of managing your own storage infrastructure. However, accessing image files directly from Cloud Storage can be slow and inefficient, especially for large-scale training. A better option is to use serialized records, such as TFRecord or Apache Avro, which are binary formats that store multiple images and their labels in a single file. Serialized records can improve the data throughput and reduce the network latency, as well as enable data compression and sharding. You can use TensorFlow or Apache Beam APIs to create and read serialized records from Cloud Storage. This solution requires minimal code changes and can speed up your training time significantly. Reference:

Cloud Storage | Google Cloud
TFRecord and tf.Example | TensorFlow Core
Apache Avro 1.10.2 Specification
Using Apache Beam with Cloud Storage | Cloud Storage

Question: 246

You are developing an image recognition model using PyTorch based on ResNet50 architecture Your code is working fine on your local laptop on a small subsample. Your full dataset has 200k labeled images You want to quickly scale your training workload while minimizing cost. You plan to use 4 V100 GPUs What should you do?

A. Create a Google Kubernetes Engine cluster with a node pool that has 4 V100 GPUs Prepare and submit a TFJob

operator to this node pool.

B. Configure a Compute Engine VM with all the dependencies that launches the training Tram your model with Vertex AI using a custom tier that contains the required GPUs.

C. Create a Vertex AI Workbench user-managed notebooks instance with 4 V100 GPUs, and use it to tram your model.

D. Package your code with Setuptools and use a pre-built container. Train your model with Vertex AI using a custom tier that contains the required GPUs.

Answer: D

Explanation:
Vertex AI is a unified platform for building and managing machine learning solutions on Google Cloud. It provides a managed service for training custom models with various frameworks, such as TensorFlow, PyTorch, scikit-learn, and XGBoost. To train your PyTorch model with Vertex AI, you need to package your code with Setuptools, which is a Python tool for creating and distributing packages. You also need to use a pre-built container, which is a Docker image that contains the dependencies and libraries for your framework. You can choose from a list of pre-built containers provided by Google, or create your own custom container. By using a pre-built container, you can avoid the hassle of installing and configuring the environment for your model. You can also specify a custom tier for your training job, which allows you to select the number and type of GPUs you want to use. You can choose from various GPU options, such as V100, P100, K80, and T4. By using 4 V100 GPUs, you can leverage the high performance and memory capacity of these accelerators to train your model faster and cheaper than using CPUs. This solution requires minimal changes to your code and can scale your training workload efficiently. Reference:
Vertex AI | Google Cloud
Custom training with pre-built containers | Vertex AI [Using GPUs | Vertex AI]

Question: 247

You work for a retail company. You have been tasked with building a model to determine the probability of churn for each customer. You need the predictions to be interpretable so the results can be used to develop marketing campaigns that target at-risk customers. What should you do?

A. Build a random forest regression model in a Vertex AI Workbench notebook instance Configure the model to generate feature importance's after the model is trained.

B. Build an AutoML tabular regression model Configure the model to generate explanations when it makes predictions.

C. Build a custom TensorFlow neural network by using Vertex AI custom training Configure the model to generate explanations when it makes predictions.

D. Build a random forest classification model in a Vertex AI Workbench notebook instance Configure the model to generate feature importance's after the model is trained.

Answer: D

Explanation:
A random forest is an ensemble learning method that consists of many decision trees. It can be used for both regression and classification tasks. A random forest classification model can predict the probability of churn for each customer by assigning them to different classes, such as high-risk,

medium-risk, or low-risk. A random forest model can also generate feature importances, which measure how much each feature contributes to the prediction. Feature importances can help interpret the model and understand what factors influence customer churn. Vertex AI Workbench is an integrated development environment (IDE) that allows you to create and run Jupyter notebooks on Google Cloud. You can use Vertex AI Workbench to build a random forest classification model in Python, using libraries such as scikit-learn or TensorFlow. You can also configure the model to generate feature importances after the model is trained, and visualize them using plots or tables. This solution can help you build an interpretable model for customer churn prediction, and use the results to design marketing campaigns that target at-risk customers. Reference: Random Forests | scikit-learn
Vertex AI Workbench | Google Cloud
Interpreting Random Forests | Towards Data Science

Question: 248

You work for a company that is developing an application to help users with meal planning You want to use machine learning to scan a corpus of recipes and extract each ingredient (e g carrot, rice pasta) and each kitchen cookware (e.g. bowl, pot spoon) mentioned Each recipe is saved in an unstructured text file What should you do?

A. Create a text dataset on Vertex Al for entity extraction Create two entities called ingredient" and cookware" and label at least 200 examples of each entity Train an AutoML entity extraction model to extract occurrences of these entity types Evaluate performance on a holdout dataset.
B. Create a multi-label text classification dataset on Vertex Al Create a test dataset and label each recipe that corresponds to its ingredients and cookware Train a multi-class classification model Evaluate the model's performance on a holdout dataset.
C. Use the Entity Analysis method of the Natural Language API to extract the ingredients and cookware from each recipe Evaluate the model's performance on a prelabeled dataset.
D. Create a text dataset on Vertex Al for entity extraction Create as many entities as there are different ingredients and cookware Train an AutoML entity extraction model to extract those entities Evaluate the models performance on a holdout dataset.

Answer: A

Explanation:
Entity extraction is a natural language processing (NLP) task that involves identifying and extracting specific types of information from text, such as names, dates, locations, etc. Entity extraction can help you analyze a corpus of recipes and extract each ingredient and cookware mentioned in them. Vertex AI is a unified platform for building and managing machine learning solutions on Google Cloud. It provides a service for AutoML entity extraction, which allows you to create and train custom entity extraction models without writing any code. You can use Vertex AI to create a text dataset for entity extraction, and label your data with two entities: "ingredient" and "cookware". You need to label at least 200 examples of each entity type to train an AutoML entity extraction model. You can also use a holdout dataset to evaluate the performance of your model, such as precision, recall, and F1-score. This solution can help you build a machine learning model to scan a corpus of recipes and extract each ingredient and cookware mentioned in them, and use the results to help users with meal planning. Reference:

AutoML Entity Extraction | Vertex AI
Preparing data for AutoML Entity Extraction | Vertex AI

www.atmicnetworks.com

Question: 249

You work for an organization that operates a streaming music service. You have a custom production model that is serving a "next song" recommendation based on a user's recent listening history. Your model is deployed on a Vertex Al endpoint. You recently retrained the same model by using fresh dat a. The model received positive test results offline. You now want to test the new model in production while minimizing complexity. What should you do?

A. Create a new Vertex Al endpoint for the new model and deploy the new model to that new endpoint Build a service to randomly send 5% of production traffic to the new endpoint Monitor enduser metrics such as listening time If end-user metrics improve between models over time gradually increase the percentage of production traffic sent to the new endpoint.

B. Capture incoming prediction requests in BigQuery Create an experiment in Vertex Al Experiments Run batch predictions for both models using the captured data Use the user's selected song to compare the models performance side by side If the new models performance metrics are better than the previous model deploy the new model to production.

C. Deploy the new model to the existing Vertex Al endpoint Use traffic splitting to send 5% of production traffic to the new model Monitor end-user metrics, such as listening time If end-user metrics improve between models over time, gradually increase the percentage of production traffic sent to the new model.

D. Configure a model monitoring job for the existing Vertex Al endpoint. Configure the monitoring job to detect prediction drift, and set a threshold for alerts Update the model on the endpoint from the previous model to the new model If you receive an alert of prediction drift, revert to the previous model.

Answer: C

Explanation:

Traffic splitting is a feature of Vertex AI that allows you to distribute the prediction requests among multiple models or model versions within the same endpoint. You can specify the percentage of traffic that each model or model version receives, and change it at any time. Traffic splitting can help you test the new model in production without creating a new endpoint or a separate service. You can deploy the new model to the existing Vertex AI endpoint, and use traffic splitting to send 5% of production traffic to the new model. You can monitor the end-user metrics, such as listening time, to compare the performance of the new model and the previous model. If the end-user metrics improve between models over time, you can gradually increase the percentage of production traffic sent to the new model. This solution can help you test the new model in production while minimizing complexity and cost. Reference:

Traffic splitting | Vertex AI

Deploying models to endpoints | Vertex AI

Question: 250

You created a model that uses BigQuery ML to perform linear regression. You need to retrain the model on the cumulative data collected every week. You want to minimize the development effort and the scheduling cost. What should you do?

A. Use BigQuerys scheduling service to run the model retraining query periodically.

B. Create a pipeline in Vertex Al Pipelines that executes the retraining query and use the Cloud Scheduler API to run the query weekly.

C. Use Cloud Scheduler to trigger a Cloud Function every week that runs the query for retraining the model.

D. Use the BigQuery API Connector and Cloud Scheduler to trigger. Workflows every week that retrains the model.

Answer: B

Explanation:
BigQuery is a serverless data warehouse that allows you to perform SQL queries on large-scale data. BigQuery ML is a feature of BigQuery that enables you to create and execute machine learning models using standard SQL queries. You can use BigQuery ML to perform linear regression on your data and create a model. BigQuery also provides a scheduling service that allows you to create and manage recurring SQL queries. You can use BigQuery's scheduling service to run the model retraining query periodically, such as every week. You can specify the destination table for the query results, and the schedule options, such as start date, end date, frequency, and time zone. You can also monitor the status and history of your scheduled queries. This solution can help you retrain the model on the cumulative data collected every week, while minimizing the development effort and the scheduling cost. Reference:

BigQuery ML | Google Cloud
Scheduling queries | BigQuery

Question: 251

You want to migrate a scikrt-learn classifier model to TensorFlow. You plan to train the TensorFlow classifier model using the same training set that was used to train the scikit-learn model and then compare the performances using a common test set. You want to use the Vertex Al Python SDK to manually log the evaluation metrics of each model and compare them based on their F1 scores and confusion matrices. How should you log the metrics?

A.
Use the aiplatfom. log_classification_Mtrics function to log the F1 score and use the aiplatform.log_metrics function to log the confusion matrix

B.
Use the aiplatform. log_claasificationjneti^cs function to log the Fl score and the confusion matrix

C.
Use the aiplatform. log_metri.es function to log the F1 score and the confusion matrix

D.
Use the aiplatform. lcg_metrics function to log the F1 score and use the aiplatform. log_classification_metries function to log the confusion matrix

A. Option A B. Option B C. Option C D. Option D

Answer: D

Explanation:
To log the metrics of a machine learning model in TensorFlow using the Vertex AI Python SDK, you should utilize the aiplatform.log_metrics function to log the F1 score
and aiplatform.log_classification_metrics function to log the confusion matrix. These functions allow users to manually

record and store evaluation metrics for each model, facilitating an efficient comparison based on specific performance indicators like F1 scores and confusion matrices. Reference: The answer can be verified from official Google Cloud documentation and resources related to Vertex AI and TensorFlow.

Vertex AI Python SDK reference | Google Cloud

Logging custom metrics | Vertex AI

Migrating from scikit-learn to TensorFlow | TensorFlow

Question: 252

You work at a gaming startup that has several terabytes of structured data in Cloud Storage. This data includes gameplay time data user metadata and game metadat

a. You want to build a model that recommends new games to users that requires the least amount of coding. What should you do?

A. Load the data in BigQuery Use BigQuery ML to tram an Autoencoder model.

B. Load the data in BigQuery Use BigQuery ML to train a matrix factorization model.

C. Read data to a Vertex Al Workbench notebook Use TensorFlow to train a two-tower model.

D. Read data to a Vertex AI Workbench notebook Use TensorFlow to train a matrix factorization model.

Answer: B

Explanation:

BigQuery is a serverless data warehouse that allows you to perform SQL queries on large-scale data. BigQuery ML is a feature of BigQuery that enables you to create and execute machine learning models using standard SQL queries. You can use BigQuery ML to train a matrix factorization model, which is a common technique for recommender systems. Matrix factorization models learn the latent factors that represent the preferences of users and the characteristics of items, and use them to predict the ratings or interactions between users and items. You can use the CREATE MODEL statement to create a matrix factorization model in BigQuery ML, and specify the matrix_factorization option as the model type. You can also use the ML.RECOMMEND function to generate recommendations for new games based on the trained model. This solution requires the least amount of coding, as you only need to write SQL queries to train and use the model. Reference: The answer can be verified from official Google Cloud documentation and resources related to BigQuery and BigQuery ML.

BigQuery ML | Google Cloud

Using matrix factorization | BigQuery ML

ML.RECOMMEND function | BigQuery ML

Question: 253

You are developing a model to help your company create more targeted online advertising campaigns. You need to create a dataset that you will use to train the model. You want to avoid creating or reinforcing unfair bias in the model. What should you do?

Choose 2 answers

A. Include a comprehensive set of demographic features.

B. include only the demographic groups that most frequently interact with advertisements.

C. Collect a random sample of production traffic to build the training dataset.

D. Collect a stratified sample of production traffic to build the training dataset.

E. Conduct fairness tests across sensitive categories and demographics on the trained model.

Answer: C, E

Explanation:

To avoid creating or reinforcing unfair bias in the model, you should collect a representative sample of production traffic to build the training dataset, and conduct fairness tests across sensitive categories and demographics on the trained model. A representative sample is one that reflects the true distribution of the population, and does not over- or under-represent any group. A random sample is a simple way to obtain a representative sample, as it ensures that every data point has an equal chance of being selected. A stratified sample is another way to obtain a representative sample, as it ensures that every subgroup has a proportional representation in the sample. However, a stratified sample requires prior knowledge of the subgroups and their sizes, which may not be available or easy to obtain. Therefore, a random sample is a more feasible option in this case. A fairness test is a way to measure and evaluate the potential bias and discrimination of the model, based on different categories and demographics, such as age, gender, race, etc. A fairness test can help you identify and mitigate any unfair outcomes or impacts of the model, and ensure that the model treats all groups fairly and equitably. A fairness test can be conducted using various methods and tools, such as confusion matrices, ROC curves, fairness indicators, etc. Reference: The answer can be verified from official Google Cloud documentation and resources related to data sampling and fairness testing.

Sampling data | BigQuery
Fairness Indicators | TensorFlow
What-if Tool | TensorFlow

Question: 254

You are developing an ML model in a Vertex Al Workbench notebook. You want to track artifacts and compare models during experimentation using different approaches. You need to rapidly and easily transition successful experiments to production as you iterate on your model implementation. What should you do?

A. 1 Initialize the Vertex SDK with the name of your experiment Log parameters and metrics for each experiment, and attach dataset and model artifacts as inputs and outputs to each execution.
2 After a successful experiment create a Vertex Al pipeline.

B. 1. Initialize the Vertex SDK with the name of your experiment Log parameters and metrics for each experiment, save your dataset to a Cloud Storage bucket and upload the models to Vertex Al Model Registry.
2 After a successful experiment create a Vertex Al pipeline.

C. 1 Create a Vertex Al pipeline with parameters you want to track as arguments to your Pipeline Job Use the Metrics. Model, and Dataset artifact types from the Kubeflow Pipelines DSL as the inputs and outputs of the components in your pipeline.
2. Associate the pipeline with your experiment when you submit the job.

D. 1 Create a Vertex Al pipeline Use the Dataset and Model artifact types from the Kubeflow Pipelines. DSL as the inputs and outputs of the components in your pipeline.
2. In your training component use the Vertex Al SDK to create an experiment run Configure the log_params and log_metrics functions to track parameters and metrics of your experiment.

Answer: A

Explanation:

Vertex AI is a unified platform for building and managing machine learning solutions on Google Cloud. It provides

various services and tools for different stages of the machine learning lifecycle, such as data preparation, model training, deployment, monitoring, and experimentation. Vertex AI Workbench is an integrated development environment (IDE) that allows you to create and run Jupyter notebooks on Google Cloud. You can use Vertex AI Workbench to develop your ML model in Python, using libraries such as TensorFlow, PyTorch, scikit-learn, etc. You can also use the Vertex SDK, which is a Python client library for Vertex AI, to track artifacts and compare models during experimentation. You can use the aiplatform.init function to initialize the Vertex SDK with the name of your experiment. You can use the aiplatform.start_run and aiplatform.end_run functions to create and close an experiment run. You can use

the aiplatform.log_params and aiplatform.log_metrics functions to log the parameters and metrics for each experiment run. You can also use

the aiplatform.log_datasets and aiplatform.log_model functions to attach the dataset and model artifacts as inputs and outputs to each experiment run. These functions allow you to record and store the metadata and artifacts of your experiments, and compare them using the Vertex AI Experiments UI. After a successful experiment, you can create a Vertex AI pipeline, which is a way to automate and orchestrate your ML workflows. You can use the aiplatform.PipelineJob class to create a pipeline job, and specify the components and dependencies of your pipeline. You can also use

the aiplatform.CustomContainerTrainingJob class to create a custom container training job, and use the run method to run the job as a pipeline component. You can use

the aiplatform.Model.deploy method to deploy your model as a pipeline component. You can also use the aiplatform.Model.monitor method to monitor your model as a pipeline component. By creating a Vertex AI pipeline, you can rapidly and easily transition successful experiments to production, and reuse and share your ML workflows. This solution requires minimal changes to your code, and leverages the Vertex AI services and tools to streamline your ML development process. Reference: The answer can be verified from official Google Cloud documentation and resources related to Vertex AI, Vertex AI Workbench, Vertex SDK, and Vertex AI pipelines.

Vertex AI | Google Cloud
Vertex AI Workbench | Google Cloud
Vertex SDK for Python | Google Cloud
Vertex AI pipelines | Google Cloud

Question: 255

You recently created a new Google Cloud Project After testing that you can submit a Vertex Al Pipeline job from the Cloud Shell, you want to use a Vertex Al Workbench user-managed notebook instance to run your code from that instance You created the instance and ran the code but this time the job fails with an insufficient permissions error. What should you do?

A. Ensure that the Workbench instance that you created is in the same region of the Vertex Al Pipelines resources you will use.
B. Ensure that the Vertex Al Workbench instance is on the same subnetwork of the Vertex Al Pipeline resources that you will use.
C. Ensure that the Vertex Al Workbench instance is assigned the Identity and Access Management (1AM) Vertex Al User rote.
D. Ensure that the Vertex Al Workbench instance is assigned the Identity and Access Management (1AM) Notebooks Runner role.

Answer: C

Explanation:
Vertex AI Workbench is an integrated development environment (IDE) that allows you to create and run Jupyter notebooks on Google Cloud. Vertex AI Pipelines is a service that allows you to create and manage machine learning workflows using Vertex AI components. To submit a Vertex AI Pipeline job from a Vertex AI Workbench instance, you need to have the appropriate permissions to access the Vertex AI resources. The Identity and Access Management (IAM) Vertex AI User role is a predefined role that grants the minimum permissions required to use Vertex AI services, such as creating and deploying models, endpoints, and pipelines. By assigning the Vertex AI User role to the Vertex AI Workbench instance, you can ensure that the instance has sufficient permissions to submit a Vertex AI Pipeline job. You can assign the role to the instance by using the Cloud Console, the gcloud command-line tool, or the Cloud IAM API.
Reference: The answer can be verified from official Google Cloud documentation and resources related to Vertex AI Workbench, Vertex AI Pipelines, and IAM. Vertex AI Workbench | Google Cloud
Vertex AI Pipelines | Google Cloud
Vertex AI roles | Google Cloud
Granting, changing, and revoking access to resources | Google Cloud

Question: 256

You work for a semiconductor manufacturing company. You need to create a real-time application that automates the quality control process High-definition images of each semiconductor are taken at the end of the assembly line in real time. The photos are uploaded to a Cloud Storage bucket along with tabular data that includes each semiconductor's batch number serial number dimensions, and weight You need to configure model training and serving while maximizing model accuracy. What should you do?

A. Use Vertex Al Data Labeling Service to label the images and train an AutoML image classification model. Deploy the model and configure Pub/Sub to publish a message when an image is categorized into the failing class.
B. Use Vertex Al Data Labeling Service to label the images and train an AutoML image classification model. Schedule a daily batch prediction job that publishes a Pub/Sub message when the job completes.
C. Convert the images into an embedding representation Import this data into BigQuery, and train a BigQuery. ML K-means clustenng model with two clusters Deploy the model and configure Pub/Sub to publish a message when a semiconductor's data is categorized into the failing cluster.
D. Import the tabular data into BigQuery use Vertex Al Data Labeling Service to label the data and train an AutoML tabular classification model Deploy the model and configure Pub/Sub to publish a message when a semiconductor's data is categorized into the failing class.

Answer: A

Explanation:
Vertex AI is a unified platform for building and managing machine learning solutions on Google Cloud. It provides various services and tools for different stages of the machine learning lifecycle, such as data preparation, model training, deployment, monitoring, and experimentation. Vertex AI Data Labeling Service is a service that allows you to create and manage human-labeled datasets for machine learning. You can use Vertex AI Data Labeling Service to label the images of semiconductors with binary labels, such as “pass” or “fail”, based on the quality criteria. You can also use Vertex AI AutoML Image Classification, which is a service that allows you to create and train custom image classification models without writing any code. You can use Vertex AI AutoML Image Classification to train an image classification model on the labeled images of semiconductors, and optimize the model for accuracy. You can also use Vertex AI to deploy the model to an endpoint, which is a service that allows you to serve online predictions from your model. You

can configure Pub/Sub, which is a service that allows you to publish and subscribe to messages, to publish a message when an image is categorized into the failing class by the model. You can use the message to trigger an action, such as alerting the quality control team or stopping the production line. This solution can help you create a real-time application that automates the quality control process of semiconductors, and maximizes the model accuracy.
Reference: The answer can be verified from official Google Cloud documentation and resources related to Vertex AI, Vertex AI Data Labeling Service, Vertex AI AutoML Image Classification, and Pub/Sub.

Vertex AI | Google Cloud

Vertex AI Data Labeling Service | Google Cloud

Vertex AI AutoML Image Classification | Google Cloud

Pub/Sub | Google Cloud

Question: 257

You work for a rapidly growing social media company. Your team builds TensorFlow recommender models in an on-premises CPU cluster. The data contains billions of historical user events and 100 000 categorical features. You notice that as the data increases the model training time increases. You plan to move the models to Google Cloud You want to use the most scalable approach that also minimizes training time. What should you do?

A. Deploy the training jobs by using TPU VMs with TPUv3 Pod slices, and use the TPUEmbedding API. B. Deploy the training jobs in an autoscaling Google Kubernetes Engine cluster with CPUs C. Deploy a matrix factorization model training job by using BigQuery ML.

D. Deploy the training jobs by using Compute Engine instances with A100 GPUs and use the t f. nn. embedding_lookup API.

Answer: A

Explanation:
TPU VMs with TPUv3 Pod slices are the most scalable and performant option for training large-scale recommender models on Google Cloud. TPUv3 Pods can provide up to 2048 cores and 32 TB of memory, and can process billions of examples and features in minutes. The TPUEmbedding API is designed to efficiently handle large-scale categorical features and embeddings, and can reduce the memory footprint and communication overhead of the model. The other options are either less scalable (B and C) or less efficient (D) for this use case.

Question: 258

You are training and deploying updated versions of a regression model with tabular data by using Vertex Al Pipelines. Vertex Al Training Vertex Al Experiments and Vertex Al Endpoints. The model is deployed in a Vertex Al endpoint and your users call the model by using the Vertex Al endpoint. You want to receive an email when the feature data distribution changes significantly, so you can retrigger the training pipeline and deploy an updated version of your model What should you do?

A. Use Vertex Al Model Monitoring Enable prediction drift monitoring on the endpoint. and specify a notification email.

B. In Cloud Logging, create a logs-based alert using the logs in the Vertex Al endpoint. Configure Cloud Logging to send an email when the alert is triggered.

C. In Cloud Monitoring create a logs-based metric and a threshold alert for the metric. Configure Cloud Monitoring to send an email when the alert is triggered.

D. Export the container logs of the endpoint to BigQuery Create a Cloud Function to run a SQL query over the exported logs and send an email. Use Cloud Scheduler to trigger the Cloud Function.

Answer: A

Explanation:

Prediction drift is the change in the distribution of feature values or labels over time. It can affect the performance and accuracy of the model, and may require retraining or redeploying the model.

Vertex AI Model Monitoring allows you to monitor prediction drift on your deployed models and endpoints, and set up alerts and notifications when the drift exceeds a certain threshold. You can specify an email address to receive the notifications, and use the information to retrigger the training pipeline and deploy an updated version of your model. This is the most direct and convenient way to achieve your goal. Reference:

Vertex AI Model Monitoring

Monitoring prediction drift

Setting up alerts and notifications

Question: 259

You have trained an XGBoost model that you plan to deploy on Vertex AI for online prediction. You are now uploading your model to Vertex AI Model Registry, and you need to configure the explanation method that will serve online prediction requests to be returned with minimal latency. You also want to be alerted when feature attributions of the model meaningfully change over time. What should you do?

A. 1 Specify sampled Shapley as the explanation method with a path count of 5.

2 Deploy the model to Vertex AI Endpoints.

3 . Create a Model Monitoring job that uses prediction drift as the monitoring objective.

B. 1 Specify Integrated Gradients as the explanation method with a path count of 5.

2 Deploy the model to Vertex AI Endpoints.

3. Create a Model Monitoring job that uses prediction drift as the monitoring objective.

C. 1. Specify sampled Shapley as the explanation method with a path count of 50.

2. Deploy the model to Vertex AI Endpoints.

3. Create a Model Monitoring job that uses training-serving skew as the monitoring objective.

D. 1 Specify Integrated Gradients as the explanation method with a path count of 50.

2 . Deploy the model to Vertex AI Endpoints.

3 Create a Model Monitoring job that uses training-serving skew as the monitoring objective.

Answer: A

Explanation:

Sampled Shapley is a fast and scalable approximation of the Shapley value, which is a game-theoretic concept that measures the contribution of each feature to the model prediction. Sampled Shapley is suitable for online prediction requests, as it can return feature attributions with minimal latency. The path count parameter controls the number of samples used to estimate the Shapley value, and a lower value means faster computation. Integrated Gradients is another explanation method that computes the average gradient along the path from a baseline input to the actual input. Integrated Gradients is more accurate than Sampled Shapley, but also more computationally intensive. Therefore, it is not recommended for online prediction requests, especially with a high path count. Prediction drift is the change in the distribution of feature values or labels over time. It can affect the performance and accuracy of the model, and may require retraining or redeploying the model.

Vertex AI Model Monitoring allows you to monitor prediction drift on your deployed models and endpoints, and set up alerts and notifications when the drift exceeds a certain threshold. You can specify an email address to receive the notifications, and use the information to retrigger the training pipeline and deploy an updated version of your model. This is the most direct and convenient way to achieve your goal. Training-serving skew is the difference between the data used for training the model and the data used for serving the model. It can also affect the performance and accuracy of the model, and may indicate data quality issues or model staleness. Vertex AI Model Monitoring allows you to monitor training-serving skew on your deployed models and endpoints, and set up alerts and notifications when the skew exceeds a certain threshold. However, this is not relevant to the question, as the question is about the feature attributions of the model, not the data distribution. Reference:

Vertex AI: Explanation methods
Vertex AI: Configuring explanations
Vertex AI: Monitoring prediction drift
Vertex AI: Monitoring training-serving skew

Question: 260

You have recently developed a new ML model in a Jupyter notebook. You want to establish a reliable and repeatable model training process that tracks the versions and lineage of your model artifacts. You plan to retrain your model weekly. How should you operationalize your training process?

A. 1. Create an instance of the CustomTrainingJob class with the Vertex AI SDK to train your model. 2. Using the Notebooks API, create a scheduled execution to run the training code weekly.
B. 1. Create an instance of the CustomJob class with the Vertex AI SDK to train your model.
2. Use the Metadata API to register your model as a model artifact.
3. Using the Notebooks API, create a scheduled execution to run the training code weekly.
C. 1. Create a managed pipeline in Vertex Al Pipelines to train your model by using a Vertex Al CustomTrainingJoOp component.
2. Use the ModelUploadOp component to upload your model to Vertex Al Model Registry.
3. Use Cloud Scheduler and Cloud Functions to run the Vertex Al pipeline weekly.
D. 1. Create a managed pipeline in Vertex Al Pipelines to train your model using a Vertex Al HyperParameterTuningJobRunOp component.
2. Use the ModelUploadOp component to upload your model to Vertex Al Model Registry.
3. Use Cloud Scheduler and Cloud Functions to run the Vertex Al pipeline weekly.

Answer: C

Explanation:

The best way to operationalize your training process is to use Vertex AI Pipelines, which allows you to create and run scalable, portable, and reproducible workflows for your ML models. Vertex AI Pipelines also integrates with Vertex AI Metadata, which tracks the provenance, lineage, and artifacts of your ML models. By using a Vertex AI CustomTrainingJobOp component, you can train your model using the same code as in your Jupyter notebook. By using a ModelUploadOp component, you can upload your trained model to Vertex AI Model Registry, which manages the versions and endpoints of your models. By using Cloud Scheduler and Cloud Functions, you can trigger your Vertex AI pipeline to run weekly, according to your plan.

Reference:

Vertex AI Pipelines documentation

Vertex AI Metadata documentation

Vertex AI CustomTrainingJobOp documentation

ModelUploadOp documentation

Cloud Scheduler documentation

[Cloud Functions documentation]

Question: 261

You work at a gaming startup that has several terabytes of structured data in Cloud Storage. This data includes gameplay time data, user metadata, and game metadat

a. You want to build a model that recommends new games to users that requires the least amount of coding. What should you do?

A. Load the data in BigQuery. Use BigQuery ML to train an Autoencoder model.

B. Load the data in BigQuery. Use BigQuery ML to train a matrix factorization model.

C. Read data to a Vertex AI Workbench notebook. Use TensorFlow to train a two-tower model.

D. Read data to a Vertex AI Workbench notebook. Use TensorFlow to train a matrix factorization model.

Answer: B

Explanation:

The best option to build a game recommendation model with the least amount of coding is to use BigQuery ML, which allows you to create and execute machine learning models using standard SQL queries. BigQuery ML supports several types of models, including matrix factorization, which is a common technique for collaborative filtering-based recommendation systems. Matrix factorization models learn latent factors for users and items from the observed ratings, and then use them to predict the ratings for new user-item pairs. BigQuery ML provides a built-in function called ML.RECOMMEND that can generate recommendations for a given user based on a trained matrix factorization model. To use BigQuery ML, you need to load the data in BigQuery, which is a serverless, scalable, and cost-effective data warehouse. You can use the bq command-line tool, the BigQuery API, or the Cloud Console to load data from Cloud Storage to BigQuery. Alternatively, you can use federated queries to query data directly from Cloud Storage without loading it to BigQuery, but this may incur additional costs and performance overhead. Option A is incorrect because BigQuery ML does not support Autoencoder models, which are a type of neural network that can learn compressed representations of the input data. Autoencoder models are not suitable for recommendation systems, as they do not capture the interactions between users and items. Option C is incorrect because using TensorFlow to train a two-tower model requires more coding than using BigQuery ML. A two-tower model is a type of neural network that learns embeddings for users and items separately, and then combines them with a dot product or a cosine similarity to compute the rating. TensorFlow is a low-level framework that requires you to define the model architecture, the loss function, the optimizer, the training loop, and the evaluation metrics. Moreover, you need to read the data from Cloud Storage to a Vertex AI Workbench notebook, which is an instance of JupyterLab that runs on a Google Cloud virtual machine. This may involve additional steps such as authentication, authorization, and data preprocessing. Option D is incorrect because using TensorFlow to train a matrix factorization model also requires more coding than using BigQuery ML. Although TensorFlow provides some high-level APIs such as Keras and TensorFlow Recommenders that can simplify the model development, you still need to handle the data loading and the model training and evaluation

yourself. Furthermore, you need to read the data from Cloud Storage to a Vertex AI Workbench notebook, which may incur additional complexity and costs. Reference: BigQuery ML documentation
Using matrix factorization with BigQuery ML Recommendations AI documentation Loading data into BigQuery
Querying data in Cloud Storage from BigQuery
Vertex AI Workbench documentation
TensorFlow documentation
TensorFlow Recommenders documentation

Question: 262

While running a model training pipeline on Vertex AI, you discover that the evaluation step is failing because of an out-of-memory error. You are currently using TensorFlow Model Analysis (TFMA) with a standard Evaluator TensorFlow Extended (TFX) pipeline component for the evaluation step. You want to stabilize the pipeline without downgrading the evaluation quality while minimizing infrastructure overhead. What should you do?

A. Add tfma.MetricsSpec () to limit the number of metrics in the evaluation step.
B. Migrate your pipeline to Kubeflow hosted on Google Kubernetes Engine, and specify the appropriate node parameters for the evaluation step.
C. Include the flag -runnerDataflowRunner in beam_pipeline_args to run the evaluation step on Dataflow.
D. Move the evaluation step out of your pipeline and run it on custom Compute Engine VMs with sufficient memory.

Answer: C

Explanation:
The best option to stabilize the pipeline without downgrading the evaluation quality while minimizing infrastructure overhead is to use Dataflow as the runner for the evaluation step. Dataflow is a fully managed service for executing Apache Beam pipelines that can scale up and down according to the workload. Dataflow can handle large-scale, distributed data processing tasks such as model evaluation, and it can also integrate with Vertex AI Pipelines and TensorFlow Extended (TFX). By using the flag -runnerDataflowRunner in beam_pipeline_args, you can instruct the Evaluator component to run the evaluation step on Dataflow, instead of using the default DirectRunner, which runs locally and may cause out-of-memory errors. Option A is incorrect because adding tfma.MetricsSpec() to limit the number of metrics in the evaluation step may downgrade the evaluation quality, as some important metrics may be omitted. Moreover, reducing the number of metrics may not solve the out-of-memory error, as the evaluation step may still consume a lot of memory depending on the size and complexity of the data and the model. Option B is incorrect because migrating the pipeline to Kubeflow hosted on Google Kubernetes Engine (GKE) may increase the infrastructure overhead, as you need to provision, manage, and monitor the GKE cluster yourself. Moreover, you need to specify the appropriate node parameters for the evaluation step, which may require trial and error to find the optimal configuration. Option D is incorrect because moving the evaluation step out of the pipeline and running it on custom Compute Engine VMs may also increase the infrastructure overhead, as you need to create, configure, and delete the VMs yourself.
Moreover, you need to ensure that the VMs have sufficient memory for the evaluation step, which may require trial and error to find the optimal machine type. Reference:
Dataflow documentation

Using DataflowRunner
Evaluator component documentation Configuring the Evaluator component

Question: 263

You developed a BigQuery ML linear regressor model by using a training dataset stored in a BigQuery table. New data is added to the table every minute. You are using Cloud Scheduler and Vertex AI Pipelines to automate hourly model training, and use the model for direct inference. The feature preprocessing logic includes quantile bucketization and MinMax scaling on data received in the last hour. You want to minimize storage and computational overhead. What should you do?

A. Create a component in the Vertex AI Pipelines directed acyclic graph (DAG) to calculate the required statistics, and pass the statistics on to subsequent components.
B. Preprocess and stage the data in BigQuery prior to feeding it to the model during training and inference.
C. Create SQL queries to calculate and store the required statistics in separate BigQuery tables that are referenced in the CREATE MODEL statement.
D. Use the TRANSFORM clause in the CREATE MODEL statement in the SQL query to calculate the required statistics.

Answer: D

Explanation:
The best option to minimize storage and computational overhead is to use the TRANSFORM clause in the CREATE MODEL statement in the SQL query to calculate the required statistics. The TRANSFORM clause allows you to specify feature preprocessing logic that applies to both training and prediction.
The preprocessing logic is executed in the same query as the model creation, which avoids the need to create and store intermediate tables. The TRANSFORM clause also supports quantile bucketization and MinMax scaling, which are the preprocessing steps required for this scenario. Option A is incorrect because creating a component in the Vertex AI Pipelines DAG to calculate the required statistics may increase the computational overhead, as the component needs to run separately from the model creation. Moreover, the component needs to pass the statistics to subsequent components, which may increase the storage overhead. Option B is incorrect because preprocessing and staging the data in BigQuery prior to feeding it to the model may also increase the storage and computational overhead, as you need to create and maintain additional tables for the preprocessed data. Moreover, you need to ensure that the preprocessing logic is consistent for both training and inference. Option C is incorrect because creating SQL queries to calculate and store the required statistics in separate BigQuery tables may also increase the storage and computational overhead, as you need to create and maintain additional tables for the statistics. Moreover, you need to ensure that the statistics are updated regularly to reflect the new data. Reference: BigQuery ML documentation Using the TRANSFORM clause
Feature preprocessing with BigQuery ML

Question: 264

You are creating a social media app where pet owners can post images of their pets. You have one million user uploaded images with hashtags. You want to build a comprehensive system that recommends images to users that are similar in appearance to their own uploaded images. What should you do?

A. Download a pretrained convolutional neural network, and fine-tune the model to predict hashtags based on the input images. Use the predicted hashtags to make recommendations. B. Retrieve image labels and dominant colors from the input images using the Vision API. Use these properties and the hashtags to make recommendations.

C. Use the provided hashtags to create a collaborative filtering algorithm to make recommendations. D. Download a pretrained convolutional neural network, and use the model to generate embeddings of the input images. Measure similarity between embeddings to make recommendations.

Answer: D

Explanation:

The best option to build a comprehensive system that recommends images to users that are similar in appearance to their own uploaded images is to download a pretrained convolutional neural network (CNN), and use the model to generate embeddings of the input images. Embeddings are low-dimensional representations of high-dimensional data that capture the essential features and semantics of the data. By using a pretrained CNN, you can leverage the knowledge learned from large-scale image datasets, such as ImageNet, and apply it to your own domain. A pretrained CNN can be used as a feature extractor, where the output of the last hidden layer (or any intermediate layer) is taken as the embedding vector for the input image. You can then measure the similarity between embeddings using a distance metric, such as cosine similarity or Euclidean distance, and

recommend images that have the highest similarity scores to the user's uploaded image. Option A is incorrect because downloading a pretrained CNN and fine-tuning the model to predict hashtags based on the input images may not capture the visual similarity of the images, as hashtags may not reflect the appearance of the images accurately. For example, two images of different breeds of dogs may have the same hashtag #dog, but they may not look similar to each other. Moreover, fine-tuning the model may require additional data and computational resources, and it may not generalize well to new images that have different or missing hashtags. Option B is incorrect because retrieving image labels and dominant colors from the input images using the Vision API may not capture the visual similarity of the images, as labels and colors may not reflect the fine-grained details of the images. For example, two images of the same breed of dog may have different labels and colors depending on the background, lighting, and angle of the image. Moreover, using the Vision API may incur additional costs and latency, and it may not be able to handle custom or domain-specific labels. Option C is incorrect because using the provided hashtags to create a collaborative filtering algorithm may not capture the visual similarity of the images, as collaborative filtering relies on the ratings or preferences of users, not the features of the images. For example, two images of different animals may have similar ratings or preferences from users, but they may not look similar to each other. Moreover, collaborative filtering may suffer from the cold start problem, where new images or users that have no ratings or preferences cannot be recommended. Reference:

Image similarity search with TensorFlow

Image embeddings documentation Pretrained models documentation Similarity metrics documentation

Question: 265

You are using Kubeflow Pipelines to develop an end-to-end PyTorch-based MLOps pipeline. The pipeline reads data from BigQuery, processes the data, conducts feature engineering, model training, model evaluation, and deploys the model as a binary file to Cloud Storage. You are writing code for several different versions of the feature engineering and model training steps, and running each new version in Vertex Al Pipelines.
Each pipeline run is taking over an hour to complete. You want to speed up the pipeline execution to reduce your development time, and you want to avoid additional costs. What should you do?

A. Delegate feature engineering to BigQuery and remove it from the pipeline.
B. Add a GPU to the model training step.
C. Enable caching in all the steps of the Kubeflow pipeline.
D. Comment out the part of the pipeline that you are not currently updating.

Answer: C

Explanation:
Kubeflow Pipelines allows for efficient use of compute resources through parallel task execution and caching, which eliminates redundant executions1. By enabling caching in all the steps of the Kubeflow pipeline, you can avoid re-running the same steps when you execute the pipeline multiple times. This can significantly speed up the pipeline execution and reduce your development time without incurring additional costs

Question: 266

You have developed an AutoML tabular classification model that identifies high-value customers who interact with your organization's website.
You plan to deploy the model to a new Vertex Al endpoint that will integrate with your website application. You expect higher traffic to the website during
nights and weekends. You need to configure the model endpoint's deployment settings to minimize latency and cost. What should you do?

A. Configure the model deployment settings to use an n1-standard-32 machine type.
B. Configure the model deployment settings to use an n1-standard-4 machine type. Set the minReplicaCount value to 1 and the maxReplicaCount value to 8.
C. Configure the model deployment settings to use an n1-standard-4 machine type and a GPU accelerator. Set the minReplicaCount value to 1 and the maxReplicaCount value to 4.
D. Configure the model deployment settings to use an n1-standard-8 machine type and a GPU accelerator.

Answer: B

Explanation:

Deploying a model to an endpoint in Vertex AI associates physical resources with the model so it can serve online predictions with low latency1. By configuring the model deployment settings to use an n1-standard-4 machine type and setting the minReplicaCount value to 1 and the maxReplicaCount value to 8, you can ensure that the model scales according to the traffic, thereby minimizing latency and cost1. The n1-standard-4 machine type provides a balance between computing power and cost, and the dynamic scaling allows the model to handle higher traffic during nights and weekends without incurring unnecessary costs during off-peak times

Question: 267

You developed a Python module by using Keras to train a regression model. You developed two model architectures, linear regression and deep neural network (DNN). within the same module. You are using the – raining_method argument to select one of the two methods, and you are using the Learning_rate-and num_hidden_layers arguments in the DNN. You plan to use Vertex Al's hypertuning service with a Budget to perform 100 trials. You want to identify the model architecture and hyperparameter values that minimize training loss and maximize model performance What should you do?

A. Run one hypertuning job for 100 trials. Set num hidden_layers as a conditional hypetparameter based on its parent hyperparameter training_mothod. and set learning rate as a non-conditional hyperparameter

B. Run two separate hypertuning jobs. a linear regression job for 50 trials, and a DNN job for 50 trials Compare their final performance on a common validation set. and select the set of hyperparameters with the least training loss

C. Run one hypertuning job for 100 trials Set num_hidden_layers and learning_rate as conditional hyperparameters based on their parent hyperparameter training method.

D. Run one hypertuning job with training_method as the hyperparameter for 50 trials Select the architecture with the lowest training loss. and further hypertune It and its corresponding hyperparameters for 50 trials

Answer: C

Explanation:

Question: 268

You have a custom job that runs on Vertex Al on a weekly basis The job is Implemented using a proprietary ML workflow that produces the datasets. models, and custom artifacts, and sends them to a Cloud Storage bucket Many different versions of the datasets and models were created Due to compliance requirements, your company needs to track which model was used for making a particular prediction, and needs access to the artifacts for each model. How should you configure your workflows to meet these requirement?

A. Configure a TensorFlow Extended (TFX) ML Metadata database, and use the ML Metadata API.

B. Create a Vertex Al experiment, and enable autologging inside the custom job

C. Use the Vertex Al Metadata API inside the custom Job to create context, execution, and artifacts for each model, and use events to link them together.

D. Register each model in Vertex Al Model Registry, and use model labels to store the related dataset and model information.

Answer: D

Explanation:

Question: 269

You work at an organization that maintains a cloud-based communication platform that integrates conventional chat, voice, and video conferencing into one platform. The audio recordings are stored in Cloud Storage. All recordings have an 8 kHz sample rate and are more than one minute long. You need to implement a new feature in the platform that will automatically transcribe voice call recordings into a text for future applications, such as call summarization and sentiment analysis. How

should you implement the voice call transcription feature following Google-recommended best practices?

A. Use the original audio sampling rate, and transcribe the audio by using the Speech-to-Text API with synchronous recognition.

B. Use the original audio sampling rate, and transcribe the audio by using the Speech-to-Text API with asynchronous recognition.

C. Upsample the audio recordings to 16 kHz. and transcribe the audio by using the Speech-to-Text API with synchronous recognition.

D. Upsample the audio recordings to 16 kHz. and transcribe the audio by using the Speech-to-Text API with asynchronous recognition.

Answer: D

Explanation:

Question: 270

You have recently developed a custom model for image classification by using a neural network. You need to automatically identify the values for learning rate, number of layers, and kernel size. To do this, you plan to run multiple jobs in parallel to identify the parameters that optimize performance. You want to minimize custom code development and infrastructure management. What should you do?

A. Create a Vertex Al pipeline that runs different model training jobs in parallel.

B. Train an AutoML image classification model.

C. Create a custom training job that uses the Vertex Al Vizier SDK for parameter optimization.

D. Create a Vertex Al hyperparameter tuning job.

Answer: D

Explanation:

www.atmicnetworks.com

Question: 271

You work for a company that sells corporate electronic products to thousands of businesses worldwide. Your company stores historical customer data in BigQuery. You need to build a model that predicts customer lifetime value over the next three years. You want to use the simplest approach to build the model. What should you do?

A. Access BigQuery Studio in the Google Cloud console. Run the create model statement in the SQL editor to create an ARIMA model.

B. Create a Vertex AI Workbench notebook. Use IPython magic to run the create model statement to create an ARIMA model.

C. Access BigQuery Studio in the Google Cloud console. Run the create model statement in the SQL editor to create an AutoML regression model.

D. Create a Vertex AI Workbench notebook. Use IPython magic to run the create model statement to create an AutoML regression model.

Answer: C

Explanation:

BigQuery ML allows you to build and run machine learning models using SQL queries directly within BigQuery, which is one of the simplest approaches because it doesn't require setting up an external environment like Vertex AI or managing infrastructure.

AutoML regression is more appropriate for predicting customer lifetime value (CLV) compared to ARIMA, which is typically used for time series forecasting (e.g., sales over time, stock prices, etc.). CLV prediction involves understanding complex relationships between customer behavior and value, which is best captured by a regression model.

Using BigQuery Studio and running a CREATE MODEL statement to build an AutoML regression model offers the simplicity you're looking for because it automates much of the feature engineering, model selection, and hyperparameter tuning.

The other options involving ARIMA models (A and B) are not appropriate for CLV, and setting up a Vertex AI Workbench notebook (D) introduces unnecessary complexity for this task.

Question: 272

You are an AI architect at a popular photo-sharing social media platform. Your organization's content moderation team currently scans images uploaded by users and removes explicit images manually. You want to implement an AI service to automatically prevent users from uploading explicit images. What should you do?

A. Develop a custom TensorFlow model in a Vertex AI Workbench instance. Train the model on a dataset of manually labeled images. Deploy the model to a Vertex AI endpoint. Run periodic batch inference to identify inappropriate uploads and report them to the content moderation team.

B. Train an image clustering model using TensorFlow in a Vertex AI Workbench instance. Deploy this model to a Vertex AI endpoint and configure it for online inference. Run this model each time a new image is uploaded to identify and

block inappropriate uploads.

C. Create a dataset using manually labeled images. Ingest this dataset into AutoML. Train an image classification model and deploy it to a Vertex AI endpoint. Integrate this endpoint with the image upload process to identify and block inappropriate uploads. Monitor predictions and periodically retrain the model.

D. Send a copy of every user-uploaded image to a Cloud Storage bucket. Configure a Cloud Run function that triggers the Cloud Vision API to detect explicit content each time a new image is uploaded. Report the classifications to the content moderation team for review.

Answer: D

Explanation:

Cloud Vision API offers pre-trained models specialized in identifying explicit or inappropriate content. By sending a copy of each image to a Cloud Storage bucket and triggering Cloud Vision through Cloud Run, the detection of explicit content is automated with minimal development time. Vertex AI custom models require more training data and infrastructure management, while AutoML-based solutions add more complexity. Cloud Vision's existing capabilities meet the requirement effectively and are highly scalable for real-time moderation needs.

Question: 273

You are an AI engineer working for a popular video streaming platform. You built a classification model using PyTorch to predict customer churn. Each week, the customer retention team plans to contact customers identified as at-risk for churning with personalized offers. You want to deploy the model while minimizing maintenance effort. What should you do?

A. Use Vertex AI's prebuilt containers for prediction. Deploy the container on Cloud Run to generate online predictions.

B. Use Vertex AI's prebuilt containers for prediction. Deploy the model on Google Kubernetes Engine (GKE), and configure the model for batch prediction.

C. Deploy the model to a Vertex AI endpoint, and configure the model for batch prediction. Schedule the batch prediction to run weekly.

D. Deploy the model to a Vertex AI endpoint, and configure the model for online prediction. Schedule a job to query this endpoint weekly.

Answer: C

Explanation:

Deploying the model on Vertex AI with a batch prediction configuration is ideal for weekly inference jobs since the retention team needs predictions once per week. Scheduling batch predictions minimizes computational costs, and Vertex AI's endpoint management simplifies infrastructure setup without needing additional maintenance. Using Vertex AI's prebuilt containers also provides a flexible deployment pipeline for any future model updates. Options A and D do not suit batch needs, and GKE (Option B) requires more manual maintenance.

Question: 274

Your organization's marketing team is building a customer recommendation chatbot that uses a generative AI large language model (LLM) to provide personalized product suggestions in real time. The chatbot needs to access data from millions of customers, including purchase history, browsing behavior, and preferences. The data is stored in a Cloud SQL

for PostgreSQL database. You need the chatbot response time to be less than 100ms. How should you design the system?

A. Use BigQuery ML to fine-tune the LLM with the data in the Cloud SQL for PostgreSQL database, and access the model from BigQuery.

B. Replicate the Cloud SQL for PostgreSQL database to AlloyDB. Configure the chatbot server to query AlloyDB.

C. Transform relevant customer data into vector embeddings and store them in Vertex AI Search for retrieval by the LLM.

D. Create a caching layer between the chatbot and the Cloud SQL for PostgreSQL database to store frequently accessed customer data. Configure the chatbot server to query the cache.

Answer: D

Explanation:
A caching layer is essential to reduce database access time, meeting the <100ms requirement. Caches store high-frequency, low-latency queries in memory, minimizing access delays caused by database lookups. While AlloyDB (Option B) provides performance benefits, a caching layer is more efficient and cost-effective for this purpose. BigQuery ML (Option A) is less ideal for real-time personalized responses due to access speed, and vector embeddings (Option C) are not needed unless semantic search is a requirement.

Question: 275

You need to train a ControlNet model with Stable Diffusion XL for an image editing use case. You want to train this model as quickly as possible. Which hardware configuration should you choose to train your model?

A. Configure one a2-highgpu-1g instance with an NVIDIA A100 GPU with 80 GB of RAM. Use float32 precision during model training.

B. Configure one a2-highgpu-1g instance with an NVIDIA A100 GPU with 80 GB of RAM. Use bfloat16 quantization during model training.

C. Configure four n1-standard-16 instances, each with one NVIDIA Tesla T4 GPU with 16 GB of RAM. Use float32 precision during model training.

D. Configure four n1-standard-16 instances, each with one NVIDIA Tesla T4 GPU with 16 GB of RAM. Use float16 quantization during model training.

Answer: A

Explanation:
NVIDIA A100 GPUs are optimized for training complex models like Stable Diffusion XL. Using float32 precision ensures high model accuracy during training, whereas float16 or bfloat16 may cause lower precision in gradients, especially important for image editing. Distributing across multiple instances with T4 GPUs (Options C and D) would not speed up the process effectively due to lower power and more complex setup requirements.

Question: 276

You trained a model on data stored in a Cloud Storage bucket. The model needs to be retrained

frequently in Vertex AI Training using the latest data in the bucket. Data preprocessing is required prior to retraining. You want to build a simple and efficient near-real-time ML pipeline in Vertex AI that will preprocess the data when new data arrives in the bucket. What should you do?

A. Create a pipeline using the Vertex AI SDK. Schedule the pipeline with Cloud Scheduler to preprocess the new data in the bucket. Store the processed features in Vertex AI Feature Store.
B. Create a Cloud Run function that is triggered when new data arrives in the bucket. The function initiates a Vertex AI Pipeline to preprocess the new data and store the processed features in Vertex AI Feature Store.
C. Build a Dataflow pipeline to preprocess the new data in the bucket and store the processed features in BigQuery. Configure a cron job to trigger the pipeline execution.
D. Use the Vertex AI SDK to preprocess the new data in the bucket prior to each model retraining. Store the processed features in BigQuery.

Answer: B

Explanation:
Cloud Run can be triggered on new data arrivals, which makes it ideal for near-real-time processing. The function then initiates the Vertex AI Pipeline for preprocessing and storing features in Vertex AI Feature Store, aligning with the retraining needs. Cloud Scheduler (Option A) is suitable for scheduled jobs, not event-driven triggers. Dataflow (Option C) is better suited for batch processing or ETL rather than ML preprocessing pipelines.

Question: 277

You have developed a fraud detection model for a large financial institution using Vertex AI. The model achieves high accuracy, but stakeholders are concerned about potential bias based on customer demographics. You have been asked to provide insights into the model's decision-making process and identify any fairness issues. What should you do?

A. Enable Vertex AI Model Monitoring to detect training-serving skew. Configure an alert to send an email when the skew or drift for a model's feature exceeds a predefined threshold. Retrain the model by appending new data to existing training data.
B. Compile a dataset of unfair predictions. Use Vertex AI Vector Search to identify similar data points in the model's predictions. Report these data points to the stakeholders.
C. Use feature attribution in Vertex AI to analyze model predictions and the impact of each feature on the model's predictions.
D. Create feature groups using Vertex AI Feature Store to segregate customer demographic features and non-demographic features. Retrain the model using only non-demographic features.

Answer: C

Explanation:
Feature attribution helps to determine how each feature influences predictions, essential for identifying bias. Vertex AI's built-in explainability tools provide insights without altering the model's feature space. Model monitoring (Option A) detects distributional drift rather than feature influence.

Options B and D do not directly address the request to explain model decisions or provide fairness insights.

www.atmicnetworks.com

Question: 278

You have created multiple versions of an ML model and have imported them to Vertex AI Model Registry. You want to perform A/B testing to identify the best-performing model using the simplest approach. What should you do?

A. Split incoming traffic among separate Cloud Run instances of deployed models. Monitor the performance of each version using Cloud Monitoring.
B. Split incoming traffic to distribute prediction requests among the versions. Monitor the performance of each version using Looker Studio dashboards that compare logged data for each version.
C. Split incoming traffic among Google Kubernetes Engine (GKE) clusters and use Traffic Director to distribute prediction requests to different versions. Monitor the performance of each version using Cloud Monitoring.
D. Split incoming traffic to distribute prediction requests among the versions. Monitor the performance of each version using Vertex AI's built-in monitoring tools.

Answer: D

Explanation:
Vertex AI Model Registry supports traffic splitting and built-in monitoring, making A/B testing seamless. This approach eliminates the need for additional monitoring tools and infrastructure overhead. Cloud Run and GKE solutions (Options A and C) add unnecessary complexity, while Looker Studio (Option B) requires additional configuration for monitoring.

Question: 279

You are the lead ML engineer on a mission-critical project that involves analyzing massive datasets using Apache Spark. You need to establish a robust environment that allows your team to rapidly prototype Spark models using Jupyter notebooks. What is the fastest way to achieve this?

A. Configure a Compute Engine instance with Spark and use Jupyter notebooks.
B. Set up a Dataproc cluster with Spark and use Jupyter notebooks.
C. Set up a Vertex AI Workbench instance with a Spark kernel.
D. Use Colab Enterprise with a Spark kernel.

Answer: B

Explanation:
Dataproc provides a managed Spark environment and integrates with Jupyter notebooks, ideal for large datasets and rapid prototyping. It reduces setup time compared to manual Spark configurations on Compute Engine or Vertex AI. Colab Enterprise is more suitable for small-scale prototyping rather

than extensive Spark-based analysis.

Question: 280

You work as an ML researcher at an investment bank and are experimenting with the Gemini large language model (LLM). You plan to deploy the model for an internal use case and need full control of the model's underlying infrastructure while minimizing inference time. Which serving configuration should you use for this task?

A. Deploy the model on a Vertex AI endpoint using one-click deployment in Model Garden.
B. Deploy the model on a Google Kubernetes Engine (GKE) cluster manually by creating a custom YAML manifest.
C. Deploy the model on a Vertex AI endpoint manually by creating a custom inference container.
D. Deploy the model on a Google Kubernetes Engine (GKE) cluster using the deployment options in Model Garden.

Answer: B

Explanation:
Deploying the model on GKE with a custom YAML manifest allows maximum control over infrastructure and latency, aligning with the need for low inference time and internal model use. Vertex AI's one-click deployment (Option A) limits control, and deploying on Vertex AI (Option C) doesn't allow for as much customization as a GKE setup.

Question: 281

Your company needs to generate product summaries for vendors. You evaluated a foundation model from Model Garden for text summarization but found that the summaries do not align with your company's brand voice. How should you improve this LLM-based summarization model to better meet your business objectives?

A. Increase the model's temperature parameter.
B. Fine-tune the model using a company-specific dataset.
C. Tune the token output limit in the response.
D. Replace the pre-trained model with another model in Model Garden.

Answer: B

Explanation:
Fine-tuning the model with a company-specific dataset aligns the model outputs with the brand voice, making it better suited for the company's objectives. Adjusting the temperature (Option A) affects randomness rather than content style, and changing token limits (Option C) does not impact tone. Replacing the model (Option D) is inefficient without guarantees of better alignment.

Question: 282

You are training a deep learning model for semantic image segmentation with reduced training time. While using a Deep Learning VM Image, you receive the following error: The resource 'projects/deeplearning-platforn/zones/europe-west4-c/acceleratorTypes/nvidia-tesla-k80' was not found. What should you do?

A. Ensure that you have GPU quota in the selected region.
B. Ensure that the required GPU is available in the selected region.
C. Ensure that you have preemptible GPU quota in the selected region.
D. Ensure that the selected GPU has enough GPU memory for the workload.

Answer: B

Explanation:
The error message indicates that the selected GPU type (nvidia-tesla-k80) is not available in the selected region

(europe-west4-c). This can happen when the GPU type is not supported in the region, or when the GPU quota is exhausted in the region. To avoid this error, you should ensure that the required GPU is available in the selected region before creating a Deep Learning VM Image. You can use the following steps to check the GPU availability and quota:

To check the GPU availability, you can use the gcloud compute accelerator-types list command with the --filter flag to specify the GPU type and the region. For example, to check the availability of nvidia-tesla-k80 in europe-west4-c, you can run:

gcloud compute accelerator-types list --filter"namenvidia-tesla-k80 AND zone:europe-west4-c"

If the command returns an empty result, it means that the GPU type is not supported in the region. You can either choose a different GPU type or a different region that supports the GPU type. You can use the same command without the --filter flag to list all the available GPU types and regions. For example, to list all the available GPU types in europe-west4-c, you can run: gcloud compute accelerator-types list --filter"zone:europe-west4-c"

To check the GPU quota, you can use the gcloud compute regions describe command with the -format flag to specify the region and the quota metric. For example, to check the quota for nvidia- tesla-k80 in europe-west4-c, you can run:

gcloud compute regions describe europe-west4-c --format"value(quotas.NVIDIA_K80_GPUS)"

If the command returns a value of 0, it means that the GPU quota is exhausted in the region. You can either request more quota from Google Cloud or choose a different region that has enough quota for the GPU type.

Reference:

Troubleshooting | Deep Learning VM Images | Google Cloud

Checking GPU availability

Checking GPU quota

Question: 283

You are implementing a batch inference ML pipeline in Google Cloud. The model was developed by using TensorFlow and is stored in SavedModel format in Cloud Storage. You need to apply the model to a historical dataset that is stored in a BigQuery table. You want to perform inference with minimal effort. What should you do?

A. Import the TensorFlow model by using the create model statement in BigQuery ML. Apply the historical data to the TensorFlow model.

B. Export the historical data to Cloud Storage in Avro format. Configure a Vertex AI batch prediction job to generate predictions for the exported data.

C. Export the historical data to Cloud Storage in CSV format. Configure a Vertex AI batch prediction job to generate predictions for the exported data.

D. Configure and deploy a Vertex AI endpoint. Use the endpoint to get predictions from the historical data in BigQuery.

Answer: B

Explanation:

Vertex AI batch prediction is the most appropriate and efficient way to apply a pre-trained model like TensorFlow's SavedModel to a large dataset, especially for batch processing.

The Vertex AI batch prediction job works by exporting your dataset (in this case, historical data from BigQuery) to a suitable format (like Avro or CSV) and then processing it in Cloud Storage where the model is stored.

Avro format is recommended for large datasets as it is highly efficient for data storage and is optimized for read/write operations in Google Cloud, which is why option B is correct.

Option A suggests using BigQuery ML for inference, but it does not support running arbitrary TensorFlow models

directly within BigQuery ML. Hence, BigQuery ML is not a valid option for this particular task.

Option C (exporting to CSV) is a valid alternative but is less efficient compared to Avro in terms of performance.

Option D suggests deploying a Vertex AI endpoint, which is better suited for real-time inference rather than batch inference. Since the question asks for batch inference, B is the best answer.