Google Professional-Machine-Learning-Engineer Mock Exam

Shared by Perez on 17-06-2026

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

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?

Options:

A- Create a text dataset on Vertex AI 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 AI 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 AI 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

Question 2

Question Type: MultipleChoice

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?

Options:

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 3

Question Type: MultipleChoice

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?

Options:

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

Question 4

Question Type: MultipleChoice

You recently deployed a scikit-learn model to a Vertex AI 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?

Options:

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.

Configure compute resources for prediction | Vertex AI | Google Cloud

Deploy a model to an endpoint | Vertex AI | Google Cloud

Question 5

Question Type: MultipleChoice

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?

Options:

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 6

Question Type: MultipleChoice

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?

Options:

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 24hours.
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.

Vertex AI Pipelines documentation

Model monitoring documentation

[Prediction drift]

[TensorFlow Extended documentation]

[BigQuery documentation]

[Vertex AI documentation]

Question 7

Question Type: MultipleChoice

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?

Options:

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