[Jan 17, 2025 Valid Professional-Machine-Learning-Engineer Test Answers & Professional-Machine-Learning-Engineer Exam PDF [Q133-Q157

[Jan 17, 2025] Valid Professional-Machine-Learning-Engineer Test Answers & Professional-Machine-Learning-Engineer Exam PDF [Q133-Q157]

[Jan 17, 2025] Valid Professional-Machine-Learning-Engineer Test Answers & Professional-Machine-Learning-Engineer Exam PDF
Valid Google Cloud Certified Professional-Machine-Learning-Engineer Dumps Ensure Your Passing

Google Professional Machine Learning Engineer certification is a challenging yet rewarding exam that provides candidates with the opportunity to showcase their expertise in machine learning. Google Professional Machine Learning Engineer certification is ideal for individuals who are seeking to advance their careers in this field and want to gain recognition for their skills and knowledge. With this certification, candidates can demonstrate their proficiency in machine learning and position themselves as experts in this rapidly growing field.

NEW QUESTION 133

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?
* Store image files in Cloud Storage and access them directly.
* Store image files in Cloud Storage and access them by using serialized records.
* Store image files in Cloud Filestore, and access them by using serialized records.
* Store image files in Cloud Filestore and access them directly by using an NFS mount point.

NEW QUESTION 134

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?
* Set up Vertex Al Experiments to track metrics and parameters Configure Vertex Al TensorBoard for visualization.
* 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.
* Set up a Vertex Al Workbench notebook instance Use the instance to save metrics data in a Cloud Storage bucket and to host TensorBoard locally for visualization.
* 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.

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

NEW QUESTION 135

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?
* Create multiple models using AutoML Tables
* Automate multiple training runs using Cloud Composer
* Run multiple training jobs on Al Platform with similar job names
* Create an experiment in Kubeflow Pipelines to organize multiple runs

NEW QUESTION 136

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?
* 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
* 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
* 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
* 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
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 AI 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.

References:

* 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

NEW QUESTION 137

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?

* Ensure that training is reproducible
* Ensure that all hyperparameters are tuned
* Ensure that model performance is monitored
* Ensure that feature expectations are captured in the schema
* 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 theteam 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.

NEW QUESTION 138

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?
* Perform data validation to ensure that the input data to the pipeline is the same format as the input data to the endpoint.
* 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.
* 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.
* 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.
According to the official exam guide1, one of the skills assessed in the exam is to &#8220;design, build, and productionalize ML models to solve business challenges using Google Cloud technologies&#8221;. 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

NEW QUESTION 139

You developed an ML model with Al 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?
* Significantly increase the max_batch_size TensorFlow Serving parameter
* Switch to the tensorflow-model-server-universal version of TensorFlow Serving
* Significantly increase the max_enqueued_batches TensorFlow Serving parameter
* Recompile TensorFlow Serving using the source to support CPU-specific optimizations Instruct GKE to choose an appropriate baseline minimum CPU platform for serving nodes
https://www.tensorflow.org/tfx/serving/performance

NEW QUESTION 140

You are investigating the root cause of a misclassification error made by one of your models. You used Vertex Al 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 Al Training on that copy, and deploys the model to a Vertex Al 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&#8217;?
* Use Vertex Al 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.
* Use the lineage feature of Vertex Al 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.
* Use the logging features in the Vertex Al 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.
* Find the job ID in Vertex Al Training corresponding to the training for the model Search in the logs of that job for the data used for the training.

NEW QUESTION 141

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?
* Configure Kubeflow Pipelines to schedule your multi-step workflow from training to deploying your model.
* Use a model trained and deployed on BigQuery ML and trigger retraining with the scheduled query feature in BigQuery
* Write a Cloud Functions script that launches a training and deploying job on Ai Platform that is triggered by Cloud Scheduler
* Use Cloud Composer to programmatically schedule a Dataflow job that executes the workflow from training to deploying your model
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.

NEW QUESTION 142

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?
* 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.
* 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.
* Use a time series forecasting model to predict each item&#8217;s monthly sales. Give the results to the logistics team so they can base inventory on the amount predicted by the model.
* 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.
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&#8217;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.

References:

* 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

NEW QUESTION 143

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?
* 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
* 1 Create a Vertex Al pipeline with a custom model training job as part of the pipeline Configure the pipeline&#8217;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
* 1 Create a Vertex Al 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
* 1 Create an experiment in Vertex Al Experiments

2. Create a Vertex Al 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
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&#8217;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&#8217;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&#8217;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

NEW QUESTION 144

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?
* 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.
* 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.
* 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.
* 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.
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

NEW QUESTION 145

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&#8217;s data from the past 12 hours. How should you design the architecture?
* 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.
* 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.
* 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.
* 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.

NEW QUESTION 146

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

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.

References:

* Hyperparameter tuning overview

* Using custom containers

* PyTorch on AI Platform Training

NEW QUESTION 147

You work on the data science team for a multinational beverage company. You need to develop an ML model to predict the company&#8217;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?
* Use latitude, longitude, and product type as features. Use profit as model output.
* Use latitude, longitude, and product type as features. Use revenue and expenses as model outputs.
* Use product type and the feature cross of latitude with longitude, followed by binning, as features. Use profit as model output.
* Use product type and the feature cross of latitude with longitude, followed by binning, as features. Use revenue and expenses as model outputs.

* 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&#8217;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&#8217;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&#8217;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 ormore 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&#8217;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.

References:

* Feature cross

* Binning

* [Profitability]

* [Revenue and expenses]

* [Latitude and longitude]

* [Product type]

NEW QUESTION 148

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?
* Use Kubeflow Pipelines on Google Kubernetes Engine.

* Use Vertex AI Pipelines with TensorFlow Extended (TFX) SDK.
* Use Vertex AI Pipelines with Kubeflow Pipelines SDK.
* Use Cloud Composer for the orchestration.

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&#8217;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]

NEW QUESTION 149

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?
* Configure Kubeflow Pipelines to schedule your multi-step workflow from training to deploying your model.
* Use a model trained and deployed on BigQuery ML and trigger retraining with the scheduled query feature in BigQuery
* Write a Cloud Functions script that launches a training and deploying job on Ai Platform that is triggered by Cloud Scheduler
* Use Cloud Composer to programmatically schedule a Dataflow job that executes the workflow from training to deploying your model

NEW QUESTION 150

A Machine Learning Specialist at a company sensitive to security is preparing a dataset for model training. The dataset is stored in Amazon S3 and contains Personally Identifiable Information (PII).

The dataset:

* Must be accessible from a VPC only.

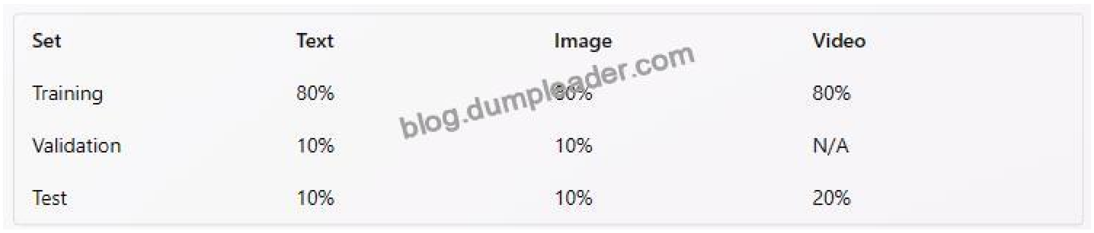
* Must not traverse the public internet.

How can these requirements be satisfied?
* Create a VPC endpoint and apply a bucket access policy that restricts access to the given VPC endpoint and the VPC.
* Create a VPC endpoint and apply a bucket access policy that allows access from the given VPC endpoint and an Amazon EC2 instance.
* Create a VPC endpoint and use Network Access Control Lists (NACLs) to allow traffic between only the given VPC endpoint and an Amazon EC2 instance.
* Create a VPC endpoint and use security groups to restrict access to the given VPC endpoint and an Amazon EC2 instance
Explanation/Reference: https://docs.aws.amazon.com/AmazonS3/latest/dev/example-bucket-policies-vpc-endpoint.html

NEW QUESTION 151

You work for a retail company. You have a managed tabular dataset in Vertex Al 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?
* Use Vertex Al manual split, using the store name feature to assign one store for each set.
* Use Vertex Al default data split.
* Use Vertex Al chronological split and specify the sales timestamp feature as the time vanable.
* Use Vertex Al random split assigning 70% of the rows to the training set, 10% to the validation set, and 20% to the test set.
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

NEW QUESTION 152

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?

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

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 pre-installed. Option C is the correct answer.

Option C: A Deep Learning VM with an n1-standard-2 machine and 1 GPU with all libraries pre-installed.

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.

References:

* Deep Learning VM Image documentation

* Compute Engine documentation

* Cloud TPU documentation

* Machine types documentation

* GPUs on Compute Engine documentation

NEW QUESTION 153

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?
* Convert each categorical value into an integer value.
* Convert the categorical string data to one-hot hash buckets.
* Map the categorical variables into a vector of boolean values.
* Convert each categorical value into a run-length encoded string.
* 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

&#8220;red&#8221;, &#8220;green&#8221;, and &#8220;blue&#8221; 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,

&#8220;AAAABBBCC&#8221; becomes &#8220;A4B3C2&#8221;. This method does not reduce the dimensionality of the input space, and does not preserve the semantic meaning of the categories4.

References:

* Encoding categorical features

* One-hot hash buckets

* Boolean vector

* Run-length encoding

NEW QUESTION 154

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?
* Train local surrogate models to explain individual predictions.
* Configure sampled Shapley explanations on Vertex Explainable AI.
* Configure integrated gradients explanations on Vertex Explainable AI.
* Measure the effect of each feature as the weight of the feature multiplied by the feature value.

NEW QUESTION 155

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

pre-installed. Option C is the correct answer.

Option C: A Deep Learning VM with an n1-standard-2 machine and 1 GPU with all libraries pre-installed.

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.

References:

* Deep Learning VM Image documentation

* Compute Engine documentation

* Cloud TPU documentation

* Machine types documentation

* GPUs on Compute Engine documentation

NEW 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?
* 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
* 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
* 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
* 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
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 AI 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.

References:

* 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

NEW QUESTION 157

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 Al 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?
* 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
* 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
* 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
* 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.
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. References:

* [Vertex AI Endpoint documentation]

* [Autoscaling documentation]

* [Monitoring documentation]

* [Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate]