

Official Google Cloud Certified Professional Data Engineer Study Guide

PREV

Chapter 8 Understanding Data Operations for Flexibilit

Next

Choosing Training and Serving Infrastructure

## Chapter 9

### Deploying Machine Learning Pipelines

Google Cloud Professional Data Engineer Exam objectives covered in this chapter include the following:

- 3. Operationalizing machine learning models
- ✓ 3.2 Deploying an ML pipeline. Considerations include:
  - Ingesting appropriate data
  - Retraining of machine learning models (Cloud AI, BigQuery ML, Kubeflow, Spark ML)
  - Continuous evaluation



Data engineers are increasingly working with machine learning (ML) pipelines. In this chapter, we will review the structure of ML pipelines and describe several ways to implement those pipelines in GCP.

ML pipelines include several stages, beginning with data ingestion and preparation, then data segregation, followed by model training and evaluation. As in other software development processes, models, like other software, should be deployed and monitored in a structured manner.

GCP provides multiple ways to implement ML pipelines. General-purpose computing resources, such as Compute Engine and Kubernetes Engine, can be used to implement ML pipelines. Managed services, such as Cloud Dataflow and Cloud Dataproc, are also available as well as specialized ML services, such as Cloud ML.

#### Structure of ML Pipelines

Machine learning projects begin with a problem definition. This could involve how to improve sales revenue by making product recommendations to customers, how to evaluate medical images to detect tumors, or how to identify fraudulent financial transactions. These are three distinct types of problems, but they can all be solved using machine learning techniques deployed via ML pipelines.

The stages of a machine learning pipeline are as follows:

- Data ingestion
- Data preparation
- Data segregation
- Model training
- Model evaluation
- Model deployment
- Model monitoring

Although the stages are listed in a linear manner, ML pipelines are more cyclic than linear, as shown in Figure 9.1. This is a difference with data-flow pipelines, like those used to ingest, transform, and store data, which are predominantly linear.



Figure 9.1 ML pipelines are usually executed in cycles.

## DATA INGESTION

Data ingestion for machine learning can be either batch or streaming.

### Batch Data Ingestion

*Batch data ingestion* should use a dedicated process for ingesting each distinct data source. For example, one process may ingest sales transactions from an e-commerce site, whereas another process ingests data about customers from another source. Batch ingestion is often done on a relatively fixed schedule, much like many data warehouse extraction, load, and transformation (ELT) processes. It is important to be able to track which batch data comes from, so you must include a batch identifier with each record that is ingested. This is considered a best practice, and it allows you to compare results across datasets more easily.

Batch data can be stored in several ways. Cloud Storage is a good option since it can store unstructured data as well as file-based, semi-structured, and structured data, such as database extracts. The lifecycle management features of Cloud Storage are also helpful. You can define policies to migrate batches of data to Nearline or Coldline storage after it has been processed by later stages of the data ingestion pipeline. Object storage is also a good option for *data lakes*, which are repositories of a wide variety of data from different sources and in different formats, which are then stored with minimal processing. If your organization is already using a well-designed data lake, you can take advantage of its services, like metadata cataloging, for storing machine learning data well.

### Streaming Data Ingestion

Cloud Pub/Sub is designed for scalable messaging, including *streaming data ingestion*. There are several advantages to using Cloud Pub/Sub for streaming data ingestion. Cloud Pub/Sub is a fully managed, serverless service that is available globally. Clients can publish up to 1000 MB/second, and subscribers can consume up to 2000 MB/second for pull subscriptions and 100 MB/second for push subscriptions. Message attributes are made up of key-value pairs. Keys are restricted to 256 bytes, but values can be as large as 1024 bytes.

Cloud Pub/Sub is a good option for ingesting streaming data that will be stored in a database, such as Bigtable or Cloud Firestore, or immediately consumed by machine learning processes running in Cloud Dataflow, Cloud Dataproc, Kubernetes Engine, or Compute Engine.

When using BigQuery, you have the option of using streaming inserts. Data is usually available for use within a few seconds of ingesting, but it may be up to 90 minutes before data is available for export or copy operations. Streaming inserts support deduplication. To deduplicate, you will need to pass in an insert identifier, named `insertID`, with each record. BigQuery caches that identifier for at least one minute and will perform deduplication on a best-effort basis. If you need to maximize the rate of ingestion and can tolerate duplicates, then not passing in a value for `insertID` will allow for higher ingestion rates. Data can be ingested into ingestion time partitioned tables as well as tables partitioned on a date or timestamp column.

Once data has been ingested into GCP, the next step is preparing that data for use with machine learning algorithms.

## DATA PREPARATION

*Data preparation* is the process of transforming data from its raw form into a structure and format that is amenable to analysis by machine learning algorithms. There are three steps to data preparation:

- Data exploration
- Data transformation
- Feature engineering

### Data Exploration

*Data exploration* is the first step to working with a new data source or a data source that has had significant changes. The goal of this stage is to understand the distribution of data and the overall quality of data.

To understand the distribution of data, we can look at basic descriptive statistics of numeric attributes, such as minimum, maximum, mean, and mode. Histograms are also useful. When working with non-numeric data, it can help to understand the number of distinct values in an attribute, which is known as the *cardinality* of the attribute, and the frequency of each of those values.

Overall quality of data is assessed by determining the number of missing or invalid values in a dataset. If some attributes of a record are missing, you may want to discard the data or use a default value, depending on the use case. Similarly, if there is an invalid value of an attribute, such as an invalid country code, you may be able to determine the correct value by looking at other attributes, such as the name of a city and province or state.

As noted in Chapter 8, "Understanding Data Operations for Flexibility and Portability," Cloud Dataprep is designed to support this kind of exploration, as well as data transformation. Cloud Data Fusion, a fully managed, code-free data integration service in GCP, is also recommended for this kind of data exploration.

### Data Transformation

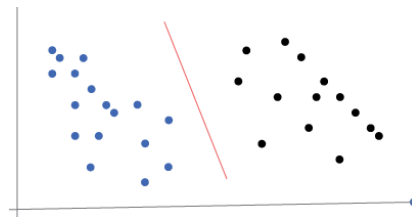
*Data transformation* is the process of mapping data from its raw form into data structures and formats that allow for machine learning. Transformations can include the following:

- Replacing missing values with a default value
- Replacing missing values with an inferred value based on other attributes in a record
- Replacing missing values with an inferred value based on attributes in other records
- Changing the format of numeric values, such as truncating or rounding real numbers to integers
- Removing or correcting attribute values that violate business logic, such as an invalid product identifier
- Deduplicating records
- Joining records from different data sets
- Aggregating data, such as summing values of metrics into hour or minute totals

Cloud Dataprep can be used for interactive data transformation, and it is especially useful when you're working with new datasets. For large volumes of data or cases in which the set of needed transformations is well defined, Cloud Dataflow is a good option for implementing transformations. Cloud Dataflow supports both batch and stream processing.

#### Feature Engineering

*Feature engineering* is the process of adding or modifying the representation of features to make implicit patterns more explicit. For example, if a ratio of two numeric features is important to classifying an instance, then calculating that ratio and including it as a feature may improve the model quality. Feature engineering is especially important when you are using models that employ some kind of linear separation to make predictions. For example, consider datasets with two features that are labeled as positive or negative. The dataset could be plotted on an x-y axis. If a binary classifier can find a line that separates positive from negative examples, this is called a *linear problem*. Figure 9.2 shows an example of a linear-separable set of instances.



**Figure 9.2** An example of a dataset that can be classified with a linear model

Now consider Figure 9.3. There is no straight line that you can draw to separate the two classes of instances.



**Figure 9.3** An example of a dataset that cannot be classified with a linear model

In cases such as the one shown in Figure 9.3, you can create new features that allow you to represent each instance in a higher dimensional space where a linear model can separate the instances. One way to do this is to use a *cross product*, which is created by multiplying two or more numeric features. For example, if we have two numeric features,  $x$  and  $y$ , we can multiply them to create a new feature  $z$ ; that is,  $z = x * y$ . Now, instead of representing each instance in a two-dimensional space, it is represented in a three-dimensional space. Cross products encode information about nonlinearity, so linear models can be used to build a classifier.

This characteristic is also common to *cross-categorical features*. For example, an online retailer building a model to predict how much a customer will spend might use a combination of a postal code and a credit rating to create a new feature. Assuming that there are 1,000 postal codes and four categories of credit ratings, a cross of these features would create a new feature with 4,000 possible values.

Another common feature engineering technique is *one-hot encoding*. This process maps a list of categorical values to a series of binary numbers that have a single value set to 1 and all other values set to 0. For example, refer to Table 9.1.

Table 9.1 One-hot encoding of a categorical feature

Category	Numeric ID
Good Credit	[1,0,0,0]
Fair Credit	[0,1,0,0]
Poor Credit	[0,0,1,0]
No Credit History	[0,0,0,1]

One-hot encoding is often used with neural networks and other algorithms that do not work directly with categorical values. Decision trees and random forests work directly with categorical values, so one-hot encoding is not necessary.

DATA SEGREGATION

Data segregation is the process splitting a data set into three segments: training, validation, and test data.

Training Data

Machine learning algorithms create models, which are functions that map from some input to a predicted output. The mappings are generated based on data rather than by manually programming them. The data used to generate these mappings is known as the *training data*.

Validation Data

Although machine learning algorithms can learn some values, known as *parameters*, from training data, there are some values that are specified by a machine learning engineer. These values are known as hyperparameters. *Hyperparameters* are values that configure a model, and they can include the number of layers in a neural network or the maximum depth of trees in a random forest model. Hyperparameters vary by machine learning algorithm. See Figure 9.4 and Figure 9.5 for examples of hyperparameters.

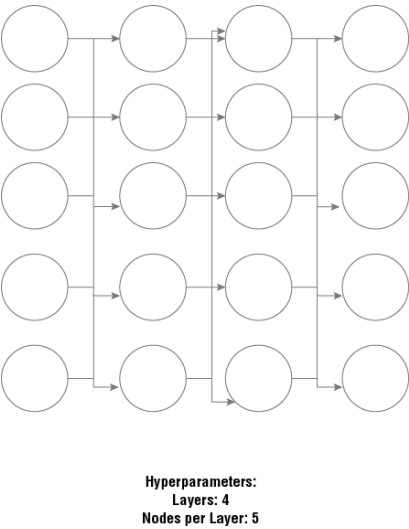


Figure 9.4 Neural networks have hyperparameters to specify the number of layers and the number of nodes in each layer.

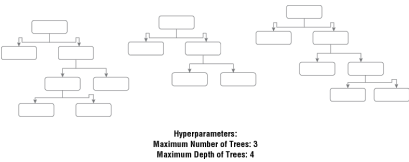


Figure 9.5 Random forest models have hyperparameters specifying the maximum number of trees to build and the maximum depth of trees.

The *validation data* set is used to tune hyperparameters. This is often done by experimenting with different values for hyperparameters. For example, in neural networks, you can specify the number of layers and number of nodes in each layer of neural networks, as well as a learning rate.

The number of layers and number of nodes in each layer of a neural network determine both how well a model will function and how long it will take to train. There is no calculation available to determine the optimal

number of layers and nodes—you have to experiment with different combinations of layers and nodes per layer to find the best configuration.

The *learning rate* determines how much neural network weights are adjusted when training. A large learning rate will make larger changes to weights, so the model may learn faster. However, you also risk missing an optimal set of weights because large changes in weights can overshoot the optimal weights. Smaller learning rates will learn more slowly but are more likely to find the optimal set of weights.

Each time you try another combination of hyperparameter values, you will need to train the model and evaluate the results. You should not use the same data to assess your hyperparameter choices as you use to train the model. The data that is used for evaluating hyperparameter choices is known as *validation data*.

#### Test Data

The third data segment is *test data*. This is data that is not used for either training or hyperparameter tuning. By using data to which the model has never been exposed and that has never been used to tune hyperparameters, you can get an estimate of how well the model will perform with other previously unseen data.

The main criteria for deciding how to split data are as follows:

- Ensuring that the test and validation datasets are large enough to produce statistically meaningful results
- Ensuring that the test and validation datasets are representative of the data as a whole
- Ensuring that the training dataset is large enough for the model to learn from in order to make accurate predictions with reasonable precision and recall

After data is segmented, the next step in ML pipelines is model training.

### MODEL TRAINING

*Model training* is the process of using training data to create a model that can be used to make predictions. This is sometimes called *fitting a model to the data*. Part of the model training phase is performing hyperparameter tuning, which we discussed earlier.

#### Feature Selection

*Feature selection* is another part of model training. This is the process of evaluating how a particular attribute or feature contributes to the predictiveness of a model. The goal is to have features of a dataset that allow a model to learn to make accurate predictions. Features are selected to do the following:

- Reduce the time and amount of data needed to train a model
- Make models easier to understand
- Reduce the number of features that have to be considered; this is known as the *curse of dimensionality*
- Reduce the risk of overfitting the model to the training data, which in turn would reduce the model accuracy on data not encountered during training

There are a number of different algorithms for selecting features. The simplest approach is to train a model with each subset of features and see which subset has the best performance. Although it is simple and easy to implement, this naive approach is not scalable.

Another approach is to start with a subset of features, measure its performance, and then make an incremental modification in the subset of features. If the performance of the modified subset is better, the modification is retained, and the process is repeated. Otherwise, the modification is discarded, and another one is tried. This is technique is known as *greedy hill climbing*.

There are many other approaches to feature selection based on information theory, correlations among features, as well as heuristics.

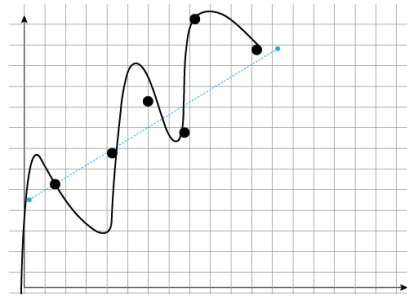
#### Underfitting, Overfitting, and Regularization

Two problems that can occur during model training are underfitting and overfitting.

*Underfitting* creates a model that is not able to predict values of training data correctly or new data that was not used during training. If the model performs poorly across multiple algorithms, and when evaluating the model using the same data that was used to train it, then that is underfitting, and it is likely caused by insufficient training data. The problem of underfitting may be corrected by increasing the amount of training data, using a different machine learning algorithm, or modifying hyperparameters. In the case of changing hyperparameters, this could include using more nodes or layers in a neural network, which is a collection of interconnected artificial neurons used to make a calculation, or increasing the number and depth of trees in a random forest—in other words, a collection decision trees, which are an ordered conjunction of features that determine a label.

*Overfitting* occurs when a model fits the training data too well. This happens when there is noise in the data and the model fits the noise as well as the correct data points. *Noise* is a term borrowed from signal processing to describe data points that are in the dataset but that are not generated by the underlying processes being modeled. A problem in the network, for

instance, could corrupt a measurement sent from a sensor, in which case that data point would be noise. For an example, see Figure 9.6. Overfitting can occur when training data and validation data are not distinct or when training datasets are too small.



**Figure 9.6** The dashed line is a linear model that does not overfit, and the solid line fits the model to an outlier data point that causes the model to overfit.

One way to compensate for the impact of noise in the data and reduce the risk of overfitting is by introducing a penalty for data points that make the model more complicated. This process is called *regularization*. Two kinds of regularization are L1 regularization, which is also known as *Lasso Regularization*, for Least Absolute Shrinkage and Selection Operator, and L2, or *Ridge Regression*.

Both L1 and L2 regularization add a penalty to the function that computes errors in a prediction, which is called the *cost function*. Machine learning algorithms use these cost functions to improve the quality of prediction. In L1 regularization, the penalty is based on magnitude or absolute value of coefficients in a model. In L2 regularization, the penalty is based on the square of the magnitude of the coefficient.

L1 regularization can shrink a feature's coefficient to zero, effectively eliminating the feature from consideration.

#### MODEL EVALUATION

There are a variety of ways to understand and evaluate the quality of a machine learning model, including

- Individual evaluation metrics
- K-fold cross validation
- Confusion matrices
- Bias and variance

##### Individual Evaluation Metrics

Machine learning models can be evaluated based on a number of metrics, including the following:

**Accuracy** This is the measure of how often a machine learning algorithm makes a correct prediction, specifically

$$\text{Accuracy} = (\text{True Positive} + \text{True Negative}) / \text{Total Predicted}$$

**Precision** This is the proportion of positive cases that were correctly identified. This measure is also known as the *positive predictive value*. The formula is as follows:

$$\text{Precision} = \text{True Positive} / (\text{True Positive} + \text{False Positive})$$

**Recall** This is the number of actual positive cases that were correctly identified as opposed to negative cases identified as positive. The formula for recall is as follows:

$$\text{Recall} = \text{True Positive} / (\text{True Positive} + \text{False Negative})$$

**F1 Score** This is a measure that combines precision and recall. The formula for calculating the F measure is as follows:

$$2 * ((\text{precision} * \text{recall}) / (\text{precision} + \text{recall}))$$

There are trade-offs to optimizing precision or recall. The F1 score combines both precision and recall, so it is often used to evaluate models.

##### K-Fold Cross Validation

*K-fold cross validation* is a technique for evaluating model performance by splitting a data set into  $k$  segments, where  $k$  is an integer. For example, if a dataset were split into five equal-sized subsets, that would be a five-fold cross-validation dataset.

K-fold cross-validation datasets are used as follows. The dataset is shuffled or randomly sorted and split into  $k$  groups. One group is held out for evaluation purposes, and the remaining groups are used to train the model. When the model is trained, it is evaluated using the dataset that was held out. The evaluation results are stored, and the process is repeated, holding out another group and training with the remaining groups until all groups have been used for evaluation.

The value of *k* should be chosen so that each segment is representative of the dataset at large. In practice, setting *k* to 10 seems to work well in many cases.

When *k* is set equal to the number of records in a dataset, that is called *leave-one-out-cross-validation*.

Confusion Matrices

*Confusion matrices* are used with classification models to show the relative performance of a model. In the case of a binary classifier, a confusion matrix would be 2×2, with one column and one row for each value. **Figure 9.7** shows an example confusion matrix for a binary classifier. The columns indicate the predicted value, and the rows indicate the actual value.

		Predicted	
		Yes	No
Actual	Yes	65	5
	No	4	26

**Figure 9.7** A confusion matrix for a classifier making 100 predictions (n = 100)

In this example, there are a total of 100 predictions. There were a total of 70 Yes instances, with 65 of those correctly predicted as Yes and 5 predicted as No. There were a total of 30 No instances. Twenty-six No instances were correctly predicted, and the prediction of the 4 others was incorrect.

One advantage of using a confusion matrix is that it helps to see quickly the accuracy of a model.

Bias and Variance

The errors made by predictive models can be understood in terms of bias and variance. Data scientists typically run many cycles of training models with different sets and sizes of training datasets. This allows them to understand bias and variance better.

*Bias* is the difference between the average prediction of a model and the correct prediction of a model. Models with high bias tend to have oversimplified representations of the process that generates the training data; this is underfitting the model. When data scientists run training models with different datasets, bias is measured by the difference of the average predicted value from a target value.

*Variance* is the variability in model predictions. It helps to understand how model prediction differs across multiple training datasets. It does not measure how accurate a prediction is from a target value, but rather the variability of a model to predict. Models with high variance tend to overfit training data so that the model works well when making predictions on the training data, but it does not generalize to data that the model has not seen before.

Ideally, models should have both low bias and low variance, and this is achieved by lowering the mean squared error (MSE) of the model by working through multiple training datasets.

MODEL DEPLOYMENT

Machine learning models are programs not unlike other programs. When they are deployed, they should be managed using the same best practices used for manually coded applications, including version control, testing, and continuous integration and deployment.

MODEL MONITORING

Machine learning models should be monitored like other applications. *Performance monitoring* should be used to understand how well the model is using resources. Stackdriver Monitoring can be used to collect metrics on resource utilization, and Stackdriver Logging can be used to capture information on events that occur while the model is running.

In addition to performance monitoring, models should be monitored for accuracy. This can be done using the model to predict the value of new instances on which the model was not trained. For example, if a model was deployed one week ago and new data has become available since then, and the actual values of the predicted feature are known, then that new data can be used to assess the accuracy of the model.

Keep in mind, however, that even if the model does not change, the accuracy of the model can change if there are changes in the process or the entity being modeled. For example, consider that a model is developed to predict how much a customer will spend on a particular type of product. The model is trained with data collected at some point in time. Now imagine that the company changes its pricing strategy. The data used to train

the model does not reflect how customers respond to the new pricing strategy and therefore will make predictions based on the prior strategy. In this case, the model should be updated by training it on the data collected since the new strategy was adopted.

ML pipelines are abstractions that can be implemented in several ways in GCP.

### GCP Options for Deploying Machine Learning Pipeline

Data engineers have multiple options for deploying machine learning workflows in GCP, from running custom programs in Compute Engine to using a fully managed machine learning service. In this section, we'll look at four options that take advantage of the machine learning capabilities of several different GCP services. They are as follows:

- Cloud AutoML
- BigQuery ML
- Kubeflow
- Spark Machine Learning

Each option is particularly well suited to a specific workload, as you'll see.

#### CLOUD AUTOML

*Cloud AutoML* is a machine learning service designed for developers who want to incorporate machine learning into their applications without having to learn many of the details of ML. The service uses a GUI to train and evaluate models, which reduces the level of effort required to get started building models. There are several AutoML products, including the following:

- AutoML Vision
- AutoML Video Intelligence (in beta as of this writing)
- AutoML Natural Language
- AutoML Translation
- AutoML Tables (in beta as of this writing)

AutoML Vision has several capabilities:

**AutoML Vision Classification** This service enables users to train their own machine learning models to classify images.

**AutoML Vision Edge - Image Classification** This capability enables users to build custom image classification models that can be deployed to edge devices. This is useful if an application requires local classification and real-time responses to the results of the classification.

**AutoML Vision Object Detection** This feature is used to train models to detect objects in images and provide information about those objects, such as their location in the image.

**AutoML Vision Edge - Object Detection** This feature enables object detection capabilities at the edge of a distributed system that includes cloud and remote or edge processing.

AutoML Video Intelligence Classification can be used to train machine learning models to classify segments of video using a custom set of labels. AutoML Video Intelligence Object Tracking supports the training of machine learning models that can detect and track multiple objects through video segments.

AutoML Natural Language enables developers to deploy machine learning applications that can analyze documents and classify them, identify entities in the text, and determine sentiment or attitudes from text.

AutoML Translation provides developers with the ability to create custom translation models. This is particularly useful if you are developing a translation application for a domain with its own nomenclature, such a field of science or engineering.

AutoML Tables builds machine learning models based on structured data. AutoML provides tools to clean and analyze datasets, including the ability to detect missing data and determine the distribution of data for each feature. It also performs common feature engineering tasks such as normalizing numeric values, creating buckets for continuous value features, and extracting date and time features from timestamps. AutoML builds models using multiple machine learning algorithms, including linear regression, deep neural network, gradient-boosted decision trees, AdaNet, and ensembles of models generated by a variety of algorithms. This approach allows AutoML Tables to determine the best algorithm for each use case.

Finally, AutoML provides comparable capabilities as BigQueryML.

#### BIGQUERY ML

*BigQuery ML* enables users of the analytical database to build machine learning models using SQL and data in BigQuery datasets. Making machine learning methods available through SQL functions, combined with not having to move data, is a key advantage to using BigQuery ML over other options.

BigQuery ML can be accessed through the following:



- BigQuery web user interface
- bq command-line tool
- BigQuery REST API
- External tools, including Jupyter Notebooks

BigQuery ML supports several types of machine learning algorithms, including

- Linear regression for forecasting
- Binary logistic regression for classification with two labels
- Multiple logistic regression for classification with more than two labels
- K-means clustering for segmenting datasets
- TensorFlow model importing to allow BigQuery users access to custom TensorFlow models

Sometimes, a use case could make use of either AutoML Tables or BigQueryML. AutoML Tables may be a better option when you want to optimize your model, without a lot of experimentation, with different algorithms and feature engineering. If you have many features, AutoML maybe a better option since it automates common feature engineering tasks. AutoML typically takes longer to return a model since it tests a variety of models, so if you need to minimize model generation time, then consider BigQueryML.

#### KUBEFLOW

*Kubeflow* is an open source project for developing, orchestrating, and deploying scalable and portable machine learning workloads. Kubeflow is designed for the Kubernetes platform. Kubeflow originally began life as a tool to help run TensorFlow jobs on Kubernetes, but it expanded to a multicloud framework for running ML pipelines. Kubeflow can be used to run machine learning workloads in multiple clouds or in a hybrid cloud environment.

Kubeflow includes the following components:

- Support for training TensorFlow models
- TensorFlow Serving, which is used to deploy trained models and make them available to other services
- A JupyterHub installation, which is a platform for spawning and managing multiple instances of a single-user Jupyter Notebook server
- Kubeflow Pipelines, which are used to define machine learning workflows

*Kubeflow Pipelines* are descriptions of machine learning workflows, including all the components needed to execute a workflow. Pipelines are packaged as Docker images.

Kubeflow is a good choice for running machine learning workloads if you are already working with Kubernetes Engine and have some experience with building machine learning models. Kubeflow supports the scalable use of machine learning models but does not provide some of the features of AutoML or the simplicity of BigQuery ML.

#### SPARK MACHINE LEARNING

*Cloud Dataproc* is a managed Spark and Hadoop service. Included with Spark is a machine learning library called *MLib*. If you are already using Cloud Dataproc or a self-managed Spark cluster, then using Spark MLlib may be a good choice for running machine learning workloads.

*Spark MLlib* contains several tools, including

- Machine learning algorithms for classification, regression, clustering, and collaborative filtering, which are used with recommendation engines
- Support for feature engineering, data transformation, and dimensionality reduction
- ML pipelines for executing multistep workflows
- Other utilities, such as math libraries and other data management tools

Spark MLlib's API supports the chaining together of multiple steps in a machine learning pipeline. Pipelines are constructed from several types of components, including

**Data Frames** Tabular structures for storing data in memory

**Transformers** Applies functions to data frames, including applying a machine learning model to generate predictions

**Estimators** Algorithms that are used to apply machine learning algorithms to create models

**Parameters** Used by transformers and estimators

Spark MLlib has a wide variety of machine learning algorithms. If you need an algorithm not available in other GCP machine learning services, consider using Spark MLlib. The available algorithms include the following:

- Support vector machines for classification
- Linear regression for forecasting
- Decision trees, random forests, and gradient-boosted trees for classification

- Naive Bayes for classification
- K-means clustering and streaming k-means for segmenting
- Latent Dirichlet allocation for segmenting
- Singular value decomposition and principal component analysis for dimensionality reduction
- Frequent Pattern (FP)–growth and association rules for frequent pattern mining

Data engineers have a variety of options for running their machine learning workloads in GCP. Cloud AutoML is designed for developers who want to build on existing machine learning models and tools that automate some machine learning tasks, like feature engineering. BigQuery ML allows SQL users to build models within BigQuery and avoid having to export data and develop models using Python or Java. Kubeflow supports deploying scalable ML pipelines in Kubernetes. Spark MLlib is a comprehensive set of machine learning tools that can be used when deploying Cloud Dataproc clusters.

Cloud AutoML and Cloud BigQuery are readily used by developers and analysts with limited or no experience building machine learning models. Both Kubeflow and Spark MLlib require some knowledge of machine learning techniques and practices.

### Exam Essentials

**Know the stages of ML pipelines.** Data ingestion, data preparation, data segregation, model training, model evaluation, model deployment, and model monitoring are the stages of ML pipelines. Although the stages are listed in a linear manner, ML pipelines are more cyclic than linear, especially relating to training and evaluation.

**Understand batch and streaming ingestion.** Batch data ingestion should use a dedicated process for ingesting each distinct data source. Batch ingestion often occurs on a relatively fixed schedule, much like many data warehouse ETL processes. It is important to be able to track which batch data comes from, so include a batch identifier with each record that is ingested. Cloud Pub/Sub is designed for scalable messaging, including ingesting streaming data. Cloud Pub/Sub is a good option for ingesting streaming data that will be stored in a database, such as Bigtable or Cloud Firestore, or immediately consumed by machine learning processes running in Cloud Dataflow, Cloud Dataproc, Kubernetes Engine, or Compute Engine. When using BigQuery, you have the option of using streaming inserts.

**Know the three kinds of data preparation.** The three kinds of data preparation are data exploration, data transformation, and feature engineering. Data exploration is the first step in working with a new data source or a data source that has had significant changes. The goal of this stage is to understand the distribution of data and the overall quality of data. Data transformation is the process of mapping data from its raw form into data structures and formats that allow for machine learning. Transformations can include replacing missing values with a default value, changing the format of numeric values, and deduplicating records. Feature engineering is the process of adding or modifying the representation of features to make implicit patterns more explicit. For example, if a ratio of two numeric features is important to classifying an instance, then calculating that ratio and including it as a feature may improve the model quality. Feature engineering includes the understanding of key attributes (features) that are meaningful for machine learning objectives at hand. This includes dimensional reduction.

**Know that data segregation is the process splitting a dataset into three segments: training, validation, and test data.** Training data is used to build machine learning models. Validation data is used during hyperparameter tuning. Test data is used to evaluate model performance. The main criteria for deciding how to split data are to ensure that the test and validation datasets are large enough to produce statistically meaningful results, that test and validation datasets are representative of the data as a whole, and that the training dataset is large enough for the model to learn to make accurate predictions with reasonable precision and recall.

**Understand the process of training a model.** Know that feature selection is the process of evaluating how a particular attribute or feature contributes to the predictiveness of a model. The goal is to have features of a dataset that allow a model to learn to make accurate predictions. Know that underfitting creates a model that is not able to predict values of training data correctly or new data that was not used during training.

**Understand underfitting, overfitting, and regularization.** The problem of underfitting may be corrected by increasing the amount of training data, using a different machine learning algorithm, or modifying hyperparameters. Understand that overfitting occurs when a model fits the training data too well. One way to compensate for the impact of noise in the data and reduce the risk of overfitting is by introducing a penalty for data points, which makes the model more complicated. This process is called regularization. Two kinds of regularization are L1 regularization, which is also known as Lasso Regularization, for Least Absolute Shrinkage and Selection Operator, and L2 or Ridge Regression.

**Know ways to evaluate a model.** Methods for evaluation a model include individual evaluation metrics, such as accuracy, precision, recall, and the F measure; k-fold cross-validation; confusion matrices; and bias

and variance. K-fold cross-validation is a technique for evaluating model performance by splitting a data set into  $k$  segments, where  $k$  is an integer. Confusion matrices are used with classification models to show the relative performance of a model. In the case of a binary classifier, a confusion matrix would be  $2 \times 2$ , with one column and one row for each value.

**Understand bias and variance.** Bias is the difference between the average prediction of a model and the correct prediction of a model. Models with high bias tend to have oversimplified models; this is underfitting the model. Variance is the variability in model predictions. Models with high variance tend to overfit training data so that the model works well when making predictions on the training data but does not generalize to data that the model has not seen before.

**Know options for deploying machine learning workloads on GCP.** These options include Cloud AutoML, BigQuery ML, Kubeflow, and Spark MLlib. Cloud AutoML is a machine learning service designed for developers who want to incorporate machine learning in their applications without having to learn many of the details of ML. BigQuery ML enables users of the analytical database to build machine learning models using SQL and data in BigQuery datasets. Kubeflow is an open source project for developing, orchestrating, and deploying scalable and portable machine learning workloads. Kubeflow is designed for the Kubernetes platform. Cloud Dataproc is a managed Spark and Hadoop service. Included with Spark is a machine learning library called MLlib, and it is a good option for machine learning workloads if you are already using Spark or need one of the more specialized algorithms included in Spark MLlib.

### Review Questions

You can find the answers in the appendix.

1. You have been tasked with helping to establish ML pipelines for your department. The models will be trained using data from several sources, including several enterprise transaction processing systems and third-party data provider datasets. Data will arrive in batches. Although you know the structure of the data now, you expect that it will change, and you will not know about the changes until the data arrives. You want to ensure that your ingestion process can store the data in whatever structure it arrives in. After the data is ingested, you will transform it as needed. What storage system would you use for batch ingestion?
  1. Cloud Storage
  2. Cloud Spanner
  3. Cloud Dataprep
  4. Cloud Pub/Sub
2. A startup company is building an anomaly detection service for manufacturing equipment. IoT sensors on manufacturing equipment transmit data on machine performance every 30 seconds. The service will initially support up to 5,000 sensors, but it will eventually grow to millions of sensors. The data will be stored in Cloud Bigtable after it is preprocessed and transformed by a Cloud Dataflow workflow. What service should be used to ingest the IoT data?
  1. Cloud Storage
  2. Cloud Bigtable
  3. BigQuery Streaming Insert
  4. Cloud Pub/Sub
3. A machine learning engineer has just started a new project. The engineer will be building a recommendation engine for an e-commerce site. Data from several services will be used, including data about products, customers, and inventory. The data is currently available in a data lake and stored in its raw, unprocessed form. What is the first thing you would recommend the machine learning engineer do to start work on the project?
  1. Ingest the data into Cloud Storage
  2. Explore the data with Cloud Dataprep
  3. Transform the data with Cloud Dataflow
  4. Transform the data with BigQuery
4. A machine learning engineer is in the process of building a model for classifying fraudulent transactions. They are using a neural network and need to decide how many nodes and layers to use in the model. They are experimenting with several different combinations of number of nodes and number of layers. What data should they use to evaluate the quality of models being developed with each combination of settings?
  1. Training data
  2. Validation data
  3. Test data
  4. Hyperparameter data

5. A developer with limited knowledge of machine learning is attempting to build a machine learning model. The developer is using data collected from a data lake with minimal data preparation. After models are built, they are evaluated. Model performance is poor. The developer has asked for your help to reduce the time needed to train the model and increase the quality of model predictions. What would you do first with the developer?

1. Explore the data with the goal of feature engineering
2. Create visualizations of accuracy, precision, recall, and F measures
3. Use tenfold cross-validation
4. Tune hyperparameters

6. A developer has built a machine learning model to predict the category of new stories. The possible values are politics, economics, business, health, science, and local news. The developer has tried several algorithms, but the model accuracy is poor even when evaluating the model on using the training data. This is an example of what kind of potential problem with a machine learning model?

1. Overfitting
2. Underfitting
3. Too much training data
4. Using tenfold cross-validation for evaluation

7. A developer has built a machine learning model to predict the category of new stories. The possible values are politics, economics, business, health, science, and local news. The developer has tried several algorithms, but the model accuracy is quite high when evaluating the model using the training data but quite low when evaluating using test data. What would you recommend to correct this problem?

1. Use confusion matrices for evaluation
2. Use L1 or L2 regularization when evaluating
3. Use L1 or L2 regularization when training
4. Tune the hyperparameters more

8. Your e-commerce company deployed a product recommendation system six months ago. The system uses a machine learning model trained using historical sales data from the previous year. The model performed well initially. When customers were shown product recommendations, the average sale value increased by 14 percent. In the past month, the model has generated an average increase of only 2 percent. The model has not changed since it was deployed six months ago. What could be the cause of the decrease in effectiveness, and what would you recommend to correct it?

1. The model is overfitting—use regularization.
2. The data used to train the model is no longer representative of current sales data, and the model should be retrained with more recent data.
3. The model should be monitored to collect performance metrics to identify the root cause of the decreasing effectiveness of the model.
4. The model is underfitting—train with more data.

9. A startup company is developing software to analyze images of traffic in order to understand congestion patterns better and how to avoid them. The software will analyze images that are taken every minute from hundreds of locations in a city. The software will need to identify cars, trucks, cyclists, pedestrians, and buildings. The data on object identities will be used by analysis algorithms to detect daily patterns, which will then be used by traffic engineers to devise new traffic flow patterns. What GCP service would you use for this?

1. AutoML Vision Object Detection
2. AutoML Vision Edge - Object Detection
3. AutoML Video Intelligence Classification
4. Auto ML Video Intelligence Object Tracking

10. An analyst would like to build a machine learning model to classify rows of data in a dataset. There are two categories into which the rows can be grouped: Type A and Type B. The dataset has over 1 million rows, and each row has 32 attributes or features. The analyst does not know which features are important. A labeled training set is available with a sufficient number of rows to train a model. The analyst would like the most accurate model possible with the least amount of effort on the analyst's part. What would you recommend?

1. KubeFlow
2. Spark MLlib
3. AutoML Tables
4. AutoML Natural Language

11. The chief financial officer of your company would like to build a program to predict which customers will likely be late paying their bills. The company has an enterprise data warehouse in BigQuery containing all the data related to customers, billing, and payments. The company does not have anyone with machine learning experience, but it does have analysts and data scientists experienced in SQL, Python, and Java. The analysts and data scientists will generate and test a large number of models, so they prefer fast model building. What service would you recommend using to build the model?
1. Kubeflow

2. Spark MLlib

3. BigQuery ML

4. AutoML Tables
12. A team of researchers is analyzing buying patterns of customers of a national grocery store chain. They are especially interested in sets of products that customers frequently buy together. The researchers plan to use association rules for this frequent pattern mining. What machine learning option in GCP would you recommend?
1. Cloud Dataflow

2. Spark MLlib

3. BigQuery ML


4. AutoML Tables

[Support / Sign Out](#)

PREV

Chapter 8 Understanding Data Operations for Flexibility and ...

Chapter 10 Choosing Training and Serving Infrastructure

NEXT