Chapter 11 Measuring, Monitoring, and Troubleshooting Machine Learning Models

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

- **3. Operationalizing machine learning models**
- **✓ 3.4 Measuring, monitoring, and troubleshooting machine learning models. Considerations include:**
- Machine learning terminology (e.g., features, labels, models, regression, classification, recommendation, supervised and unsupervised learning, evaluation metrics)
- Impact of dependencies of machine learning models
- Common sources of error (e.g., assumptions about data)

Machine learning has spawned a fundamentally new way to develop services. Rather than manually code logic, machine learning algorithms use data to identify patterns that can be used to predict values, classify objects, and recommend products to customers. The Google Cloud Professional Data Engineer exam expects data engineers to have some familiarity with machine learning. You do not have to be a seasoned machine learning engineer or data scientist, but you should be able to have a technical discussion with one. To that end, this chapter focuses on key concepts in machine learning. The chapter is organized into the following topics:

- Types of machine learning algorithms
- Deep learning
- Engineering machine learning models
- Common sources of error in machine learning models

Machine learning is a broad discipline with many areas of specialization. This chapter will provide a high-level overview to help you pass the Professional Data Engineer exam, but it is not a substitute for studying machine learning using resources designed for that purpose. Additional learning resources are included at the end of the chapter.

Three Types of Machine Learning Algorithms

Machine learning algorithms have traditionally been categorized into supervised, unsupervised, and reinforcement learning algorithms. We will review supervised and unsupervised algorithms and then briefly describe reinforcement learning.

SUPERVISED LEARNING

Supervised learning algorithms are used to make predictions. When an ML model is used to predict a discrete value, it is known as a *classification model*. When a model is used to predict a continuous value, it is known as a *regression model*.

Classification

Supervised algorithms learn from examples. Consider the data in Table 11.1, which contains information about different kinds of animals. If you wanted to build an ML model to predict whether or not an animal is a mammal, you could use data such as that shown in the table. Supervised learning algorithms use one of the columns of data to represent the value to be predicted. For example, you would use the Mammal column for the predicted value. In ML terminology, this column is called a *label*.

Table 11.1 Animal classification data

Animal	Habitat	Live birth	Hair/follicles/fur	Backbone	Mammal
Whale	Aquatic	Yes	Yes	Yes	Yes
Cow	Terrestrial	Yes	Yes	Yes	Yes
Platypus	Semi-aquatic	No	Yes	Yes	Yes
Skates	Aquatic	No	No	No	No
Ray	Aquatic	Yes	No	No	No
Crow	Terrestrial	No	No	Yes	No

The simplest form of a classifier is called a *binary classifier*. A classifier that predicts if an animal is a mammal or not a mammal is an example of a binary classifier. More realistic examples of binary classifiers are those used to predict if a credit card transaction is fraudulent or legitimate, and a machine vision ML model that predicts if a medical image contains or does not contain evidence of a malignant tumor.

A *multiclass classification model* assigns more than two values. In the animal data example, a model that predicts a habitat from the other features is an example of a multiclass classifier. Another example is a classifier that categorizes news stories into one of several categories, such as politics, business, health, and technology.

Some commonly used supervised algorithms are support vector machines (SVMs), decision trees, and logistic regression.

Support vector machines (SVMs) represent instances in a multidimensional space. A simple example is a set of points in a three-dimensional grid. SVMs find a boundary between classes that maximize the distance between those classes. Figure 11.1 shows an example of a plane separating two groups of instances. If a new point were added to the space, it could be categorized based on which side of the plane it is located.

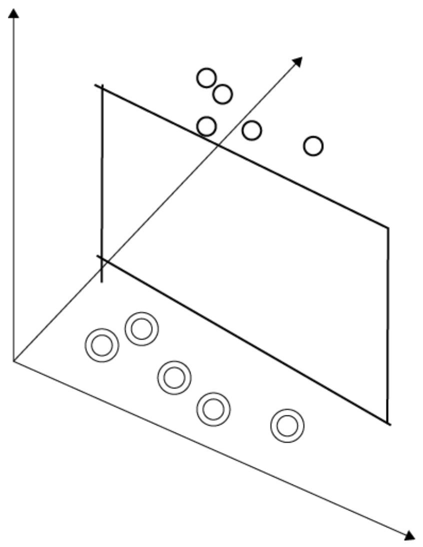

Figure 11.1 An example of how SVMs can be used to predict discrete values

In general, when features are numerical, you can imagine that each instance is a point in an *n*-dimensional space, where *n* is the number of numerical features. Thinking of instances in terms of points located in space makes some useful concepts possible for thinking about machine learning algorithms. A *decision boundary* is a line, plane, or hyperplane that separates instances into categories.

A *decision tree* is a type of supervised learning algorithm that builds a model as a series of decisions about features, such as the amount of a transaction or the frequency of a particular word in a news story. For example, Figure 11.2 shows how a decision tree can be used to predict the type of an animal.

Logistic regression is a statistical model based on the logistic function, which produces an S-shaped curve known as a *sigmoid* and is used for binary classification. Figure 11.3 shows an example of a logistic regression model that can perform binary classification.

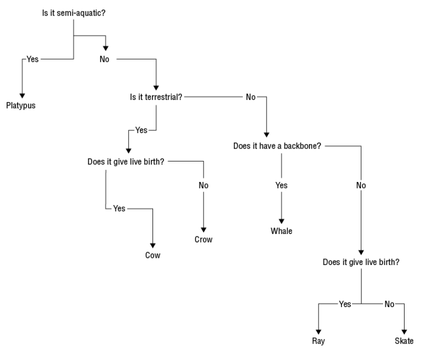

Figure 11.2 An example decision tree to predict the type of an animal

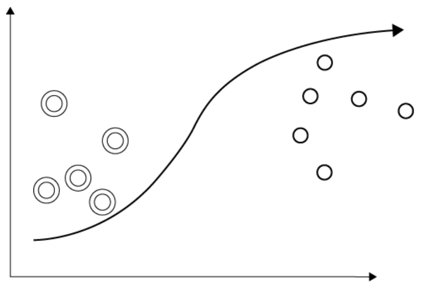

Figure 11.3 An example logistic regression function using a sigmoid curve

Regression

Supervised learning algorithms can also be used to predict a numeric value. These are called *regression algorithms*. Consider the data in Table 11.2, which shows the average number of months that it takes someone to find a job. Regression algorithms could be used with data such as this to create a model for predicting the number of months it will take a person to find a new job.

Table 11.2 Regression algorithms, which are used to predict continuous numeric values

Age	Profession	Years of experience	Average number of months to find a new job
37	Medical	10	5
22	Information Technology	1	3
25	Medical	5	10
41	Finance	12	6
52	Finance	21	9
48	Information Technology	15	7

Regression algorithms map one set of variables to other continuous variables. They are used, for example, to predict the lifetime value of a customer, to estimate the time before a piece of equipment fails given a set of operating parameters, or to predict the value of a stock at some point in the future.

Simple linear regression models use one feature to predict a value, such as using a person's age to predict their height.

Multiple linear regression models use two or more features to predict a value, such as age and gender, to predict a person's height. Figure 11.4 illustrates a simple linear regression problem.

Simple nonlinear regression models use a single feature but fit a non-straight line to data. For example, a nonlinear regression model may fit a quadratic curve (one with one bend in the curve). *Multiple nonlinear regression models* use two or more features and fit a curved line to the data.

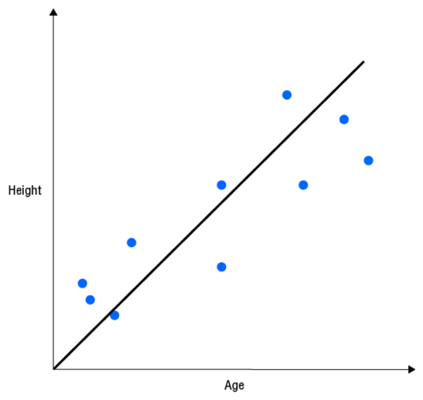

Figure 11.4 A simple linear regression example

UNSUPERVISED LEARNING

Unsupervised learning algorithms find patterns in data without using predefined labels. Two commonly used forms of unsupervised learning are *clustering* and *anomaly detection*.

Unsupervised algorithms learn by starting with unlabeled data and then identifying salient features, such as groups or clusters, and anomalies in a data stream. The animal data in Table 11.3 could be used to create clusters of similar animals. Unlike with supervised algorithms, there is no one feature or label that is being predicted. In this example, a clustering algorithm configured to find three groups within the Table 11.3 data may group the records based on habitat—aquatic, semi-aquatic, or terrestrial.

Table 11.3 Clustering algorithms, which group similar records together

Animal	Habitat	Live birth	Hair/follicles/fur	Backbone	Mammal	Cluster
Whale	Aquatic	Yes	Yes	Yes	Yes	1
Cow	Terrestrial	Yes	Yes	Yes	Yes	3
Platypus	Semi-aquatic	No	Yes	Yes	Yes	2
Skates	Aquatic	No	No	No	No	1
Ray	Aquatic	Yes	No	No	No	2
Crow	Terrestrial	No	No	Yes	No	3

Clustering, or cluster analysis, is the process of grouping instances together based on common features.

K-means clustering is a technique used for partitioning a dataset into *k* partitions and assigning each instance to one of the partitions. It works by minimizing the variance between the instances in a cluster.

Another clustering algorithm is the *K-nearest neighbors algorithm*, which takes as input the *k* closest instances and assigns a class to an instance based on the most common class of the nearest neighbors.

ANOMALY DETECTION

Anomaly detection is the process of identifying unexpected patterns in data. Anomalies come in a variety of forms. *Point anomalies* are outliers, such as a credit card transaction with an unusually large amount. *Contextual anomalies* are unusual patterns given other features of an instance. For example, purchasing a large number of umbrellas in a desert area is contextually anomalous. *Collective anomalies* are sets of instances that together create an anomaly, such as a large number of credit card transactions from different geographic areas in a short period of time.

Anomaly detection algorithms use density-based techniques, such as k-nearest neighbors, cluster analysis, and outlier detection approaches.

REINFORCEMENT LEARNING

Reinforcement learning is an approach to learning that uses agents interacting with an environment and adapting behavior based on rewards from the environment. This form of learning does not depend on labels.

Reinforcement learning is modeled as an environment, a set of agents, a set of actions, and a set of probabilities of transitioning from one state to another after a particular action is taken. A reward is given after the transition from one state to another following an action.

Reinforcement learning is useful when a problem requires a balance between short-term and long-term trade-offs. It is especially useful in robotics and game playing.

Deep Learning

If you are new to machine learning, you may be surprised not to see deep learning as a type of machine learning. The categorization used here is based on the kinds of problems solved, and that leads to the supervised, unsupervised, and reinforcement learning types. *Deep learning* is not a kind of problem but a way to solve ML problems. It can be used with any of the three types of ML problems.

Deep learning uses the concept of an artificial neuron as a building block. These neurons or nodes have one or more inputs and one output. The inputs and outputs are numeric values. The output of one neuron is used as the input of one or more other neurons. A simple, single neuron model is known as a perceptron and is trained using the perceptron algorithm. By adding layers of neurons on top of each other, you can build more complex, deep networks, such as the one shown in Figure 11.5.

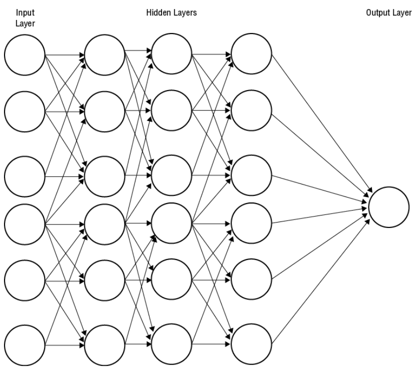

Figure 11.5 An example deep learning network

The first layer is the *input layer*. This is where feature values are input to the network. The middle layers are known as *hidden layers*. The final layer is the *output layer*. The output of that node is the output of the model.

When deep learning models are trained, parameters in each node are adjusted so that as values or signals are sent through the network, the correct value is produced by the output node. One of the most important features of neural networks is that they can learn virtually any mapping function, from inputs to outputs.

There are a few neural network and deep learning–specific terms with which you should be familiar. Each neuron in a neural network has a set of inputs and produces an output. The function that maps from inputs to outputs is known as an *activation function*. A commonly used activation function is the *rectified linear unit (ReLU)* function. When a weighted sum of inputs is equal to or less than zero, the ReLU outputs a 0. When the weighted sum of inputs is greater than zero, the ReLU outputs the weighted sum.

Other activation functions include the hyperbolic tangent (TanH) function and the sigmoid function. Both of these are nonlinear functions. If a neural network used linear activation functions, it would be possible to combine all of the linear activation functions into a single linear function, but that is not possible with nonlinear functions. Since neural networks use nonlinear activation functions, they are able to create networks that model nonlinear relationships, and that is one of the reasons why they are so successfully used on problems that do not lend themselves to solutions with linear models. Figure 11.6 shows graphs of the three activation functions described.

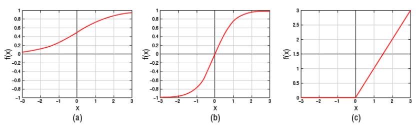

Figure 11.6 Examples of a (a) sigmoid function, (b) hyperbolic tangent function, and (c) rectilinear unit function

Since deep learning networks can learn virtually any mapping function, it is possible for networks essentially to memorize the training data and the correct responses. This is an extreme example of overfitting. Even if a network does not memorize the training data and labels, they can still overfit the training data. Deep learning models can make use of *dropout*, which is a form of regularization. During training, a random number of nodes are ignored when calculating weighted sums. This simulates removing a node from a network and, in effect, reduces the network's ability to learn and tends to produce models less likely to overfit the training data.

During training, the error of the prediction is propagated through the layers, and the size of the error propagated decreases with each layer. In deep networks, this can lead the *vanishing gradient problem*, which is that early layers in the network take much longer to train. Machine learning researchers have investigated many ways to tackle the vanishing gradient problem, and the most common solution is to use ReLU as an activation function.

Engineering Machine Learning Models

Using machine learning algorithms to build models is only one part of the ML process. Several steps are required to create a model that is ready to go into production. Once it is ready for production, it must be operationalized, which takes into consideration serving models at scale, monitoring model performance, and retraining models.

MODEL TRAINING AND EVALUATION

Building an ML model is a multistep process that includes the following:

- Data collection and preparation
- Feature engineering
- Training models
- Evaluating models

You should be familiar with each of these steps for the Professional Data Engineer exam.

Data Collection and Preparation

Data collection is similar to data collection in other pipelines, but preparation is slightly different because the goal of this stage is to map data from its original representation to a representation that is suitable for the machine learning algorithm or algorithms to be used to train ML models.

Machine learning algorithms can have particular requirements. For example, a deep learning network needs to have all features represented numerically. This is not a problem when features are continuous values, such as the total value of a sale or the number of days since the customer last made a purchase. When you are working with categorical values, such as the categories of news stories or the names of sales regions, the categories will need to be mapped to a numeric representation.

One way to map categorical values to numeric values is to assign a distinct number to each categorical value. For example, if you have four sales regions—Northeast, Southeast, Northwest, and Southwest—then you could replace those names with a 1, 2, 3, and 4, respectively. This mapping should be used only when the difference in magnitude between the numbers will not affect the model training. When magnitude between values is important, you should use one-hot encoding.

One-hot encoding maps from a category to a binary representation. Each distinct value is assigned a position in a bit vector. Using the sales regions example, you could assign Northeast to [1,0,0,0], Southeast to [0,1,0,0], Northwest to [0,0,1,0], and Southwest to [0,0,0,1]. This technique is commonly used with neural networks, including deep learning networks.

Another data preparation technique is *bucketing*, which is also called *binning*. Bucketing is the process of converting a continuous feature into ranges values, and it is used to reduce the impact of minor observation errors. For example, an IoT device reporting relative humidity may generate values from 1 to 100. For your purposes, however, you may want to indicate into which of 10 buckets a measurement falls, such as 1 to 10, 11 to 20, 21 to 30, and so forth. A related concept is *quantile bucketing*, which is the process of assigning instances to buckets so that each bucket has approximately the same number of instances. This can be useful when there are many instances in a small range and many instances spread over a large range.

You may also need to address the problem of missing or anomalous data. If data is missing, you might do one of the following:

- Discard the record
- Use a default value, such as 0
- Use an inferred value, such as the average of the previous three values for that attribute in an ordered list of records

You can use similar techniques for values that are out of range, such as values greater than 10 when all values should be between 0 and 10.

Feature Engineering

Feature engineering is the process of identifying which features are useful for building models. It also includes adding new features derived from existing features. For example, a data model that predicts if a customer is going to stop buying from you and start buying from a competitor may use as features the total amount spent in the last 30 days, last 60 days, and last 90 days. Those features by themselves may not be predictive, but the ratio of the amount spent in the last 30 days to the amount spent in the last 60 and last 90 days are two additional features that could be more informative.

Recall from the earlier section "Three Types of Machine Learning Algorithms" that some algorithms find lines, planes, or hyperplanes that separate instances of a particular category. Sometimes it is not possible to find a straight line, plane, or hyperplane that separates all instances using only the original set of features. In that case, you can use cross products of features. There are two types of cross products: cross products of numeric values and cross products that use one-hot encodings.

The cross product of numeric values is the multiple of those values. For example, if a customer purchases $100 worth of goods and it has been 12 days since the previous purchase, the cross product would be 100 × 12, or

1,200. This kind of cross product can represent nonlinear features of a dataset.

A more common form of cross product uses one-hot encodings. For example, a customer may be located in the Northeast sales region, encoded as [1,0,0,0], and their preferred language is English, encoded as [0,0,0,1]. The cross product would be a 16-bit vector with a single 1 and fifteen 0s.

Another part of feature engineering is eliminating features that you know will not be predictive, such as sequentially generated identifiers. You may also want to remove some features that highly correlate with another feature.

You can also use unsupervised learning techniques to help with feature engineering. Clustering and dimension reduction are two such techniques.

Thinking of instances as points in space is also useful when thinking about clustering or grouping of related instances. A *centroid* is the mean or center of a cluster. When you use K-means clustering, you specify *k*, which is the number of clusters. If *k* is set to 5, the algorithm will find 5 centroids, one for each cluster. Those cluster centroids could be used as a feature.

Continuing with the spatial reasoning approach, when we think of instances as points in space, we typically think of the space as having one dimension per feature. This can lead to a large number of dimensions, which can then lead to inefficient training. In those cases, it is common to use a *dimension reduction technique* to produce a set of representations of the features with fewer dimensions. *Principal component analysis (PCA)* is a dimension reduction technique.

Sometimes, there are not enough instances to train a model. *Data augmentation* is the practice of introducing artificial instances by transforming existing instances. For example, you might apply transformations to images to create new images for training an object detection model.

Training Models

When a model is trained, it is presented with a training set of data. The algorithm analyzes one instance at a time and then updates the parameters of the model. In some cases, it is more efficient to analyze multiple examples before updating parameters. This is done using batches. *Batches* are instances that are grouped together and processed in a single training iteration. The number of instances in a batch is known as the *batch size*.

Some algorithms can build a model by analyzing a training set once, whereas others, such as neural networks, can benefit from multiple passes over the training data. An *epoch* is the term used to describe one full pass over a training dataset by the training algorithm. With some ML algorithms, you will specify the number of epochs to use. This is an example of a *hyperparameter*, which is a parameter that uses the number of epochs specified instead of having the model learn it.

A simple way to think of neural network training is that the networks make predictions by taking inputs, multiplying each input by a value, called a *weight*, summing those products, and using that result as the input to a function called an *activation function*. The output of the activation function becomes the input to the next layer of the network. The process continues until the output layer emits its output. That output is compared to the expected value. If the output and expected value are different, then the difference is noted. This is known as the error. For example, if a network outputs the value of 0.5 and the expected value was 0.9, then the error is 0.4.

In this example, the network produced a number that was too large, so the weights in the network should be adjusted so that they collectively produce a smaller number. There are many ways that you could adjust weights; the most common method is to adjust weights based on the size of the error. *Backpropagation* is the most widely used algorithm for training neural networks, and it uses the error and the rate of change in the error to calculate weight adjustments. The size of adjustments is partially determined by a hyperparameter called the *learning rate*. Small learning rates can lead to longer training times, but learning rates that are too large can overshoot optimal weights.

Model training often starts with a training dataset and an algorithm to produce a model, but it is also possible to start with a trained model and build on it. This is known as *transfer learning*. This technique is often used in vision and language understanding, where a model built for one purpose, such as recognizing cats in images or identifying the names of persons in text, can be used to build other models, such as a dog recognition model or identifying the names of geographic places in text.

Regularization is a set of techniques for avoiding overgeneralization by penalizing complex models, so training tends to favor less complex, and therefore less likely to overfit, models. Two kinds of regularization are L1 and L2 regularization. *L1 regularization* calculates a penalty based on the absolute value of the weights. *L2 regularization* calculates a penalty based on the sum-of-the-squares of the weights. L1 regularization should be used when you want less relevant features to have weights close to zero. L2 regularization should be used when you want outlier instances to have weights close to zero, which is especially useful when working with linear models.

Evaluating Models

Once you have built a model, you need to assess how well it performs. This process is called *evaluation*. Just as a software engineer should not deploy code that has not been tested, an ML engineer should not deploy models that have not been evaluated.

There are several metrics that are commonly used to evaluate machine learning models.

Accuracy

Accuracy is a measure of how many predictions a model correctly makes. It is usually expressed as a percentage. The formula for calculating accuracy is the total number of correct predictions divided by the total number of predictions. When accuracy is applied to a binary classifier, accuracy is expressed in terms of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), formally written as follows:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Precision

Precision is a measure of the true positives divided by the sum of true positives and false positives. This is a measure of how well the model correctly predicts true positives and how often it classifies something as a true positive when it is not, formally written as follows:

Precision = TP / (TP + FP)

Recall

Recall is a measure of the number of true positives divided by the sum of true positives and false negatives. This is a measure of how many of actual positive examples were identified, formally written as follows:

Recall = TP / (TP + FN)

Recall is also known as sensitivity.

F1 Score

F1 score is a measure that takes into account both precision and recall. Formally, the F1 score is the harmonic mean of precision and recall:

F1 Score = 2 * [(Precision * Recall) / (Precision + Recall)]

F1 scores are useful when you want to optimize a model to have balanced precision and recall. It is also a good metric to use when evaluating models built with imbalanced training sets.

The measures can be combined to produce graphical representations of a model's performance. Two common ones are the *receiver operator characteristic (ROC)* and the *area under the curve (AUC)*.

The ROC curve is plotted as the true positive rate versus that false positive rate at varying threshold settings. Figure 11.7 shows an example of an ROC. The AUC is a measure of the area under the ROC curve.

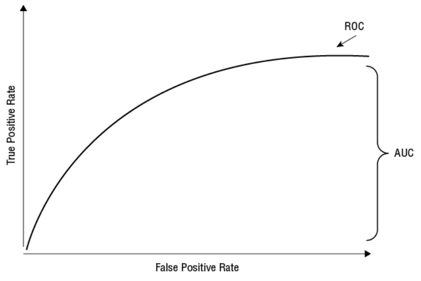

Figure 11.7 The ROC is the curved line, and the AUC is the area under that curve line.

In addition to using these metrics to measure the quality of a model, it is a common practice to compare more complex models with simpler models. This gives some idea of the trade-off between improvement in the model with increased complexity. *Baseline models* are simple models used as a reference point for evaluating more complex models. For example, a linear model may be used as a baseline for a deep learning network. *Linear models* assign a weight to each feature to make predictions and include an offset from the intercept to the origin. For example, the following equation uses four features, four weights, and a bias, which is *b*:

$$y = f1 * w1 + f2 * w2 + f3 * w3 + f4 * w4 + b$$

Baseline models also demonstrate the minimal performance that you should expect from a more complex model.

OPERATIONALIZING ML MODELS

Once models are trained and pass evaluation, they are released into production. The process of releasing and maintaining models is known as *operationalization*. It consists of the following:

- Deploying models
- Serving models

- Monitoring models
- Retraining models

Some of the considerations are the same as you would find when operationalizing any software program, and some are specific to ML models.

Deploying Models

Deploying models is a form of software deployment. They may be deployed using continuous integration/continuous deployment pipelines. Models should be deployed using *canary deployments*. These are deployments to a limited number of users to test the model under real-world conditions before making it generally available. When problems are discovered, a model should be rolled back and replaced with a previous stable version.

Model Serving

Model serving is making the model available to other services.

Models may be used in batch mode when the results of the model evaluation are not needed immediately. For example, an insurance company may have a batch process for paying insurance claims that includes a model to predict fraudulent claims. Since the company already has a batch process for this operation, model serving can be added to that process.

When a model evaluation is needed immediately—for example, when evaluating a credit card transaction—then the model should be available through an API.

You should also consider other factors common to deploying software, including scalability, reliability, and availability. If a model is running in Compute Engine, for example, you could take advantage of managed instance groups to deploy clusters of servers and load-balance across them. If you are deploying containerized models in Kubernetes Engine, you can configure your ML services to scale as needed and within the limits of business requirements. These requirements could include limits on the size of Kubernetes clusters or on the amount of resources allocated to particular ML models.

Monitoring

ML models have the same monitoring requirements as other software but have some specific issues that need to be addressed as well.

Monitoring should include monitoring source systems that provide data to train models. If those source systems are unavailable, it could prevent the updating of models. You should also monitor for changes in the schema of input sources.

Models should be tested in the training and production environments using the same test dataset. Both tests should produce the same results.

Retraining

If the conditions that were in place when a model is trained change, the quality of predictions can degrade. For example, a model used to predict which additional clothing products a customer will buy may need to be updated as the seasons change. Models should be kept up to date to prevent the quality of predictions from degrading.

Monitoring the last update of models can help detect which among them may be at risk of becoming stale.

Common Sources of Error in Machine Learning Models

Machine learning engineers face a number of challenges when building effective models. Problems like overfitting, underfitting, and vanishing gradient can be addressed by adjusting the way that a model is trained. In other cases, the data used to train can be a source of error. Three common problems are as follows:

- Data quality
- Unbalanced training sets
- Bias in training data

These problems can occur together, making it especially important to address them before beginning model training.

DATA QUALITY

Poor-quality data leads to poor models. The same can be said for anything derived from the data, unless steps are taken to compensate for it. Some common data quality problems are as follows:

- Missing data
- Invalid values
- Inconsistent use of codes and categories
- Data that is not representative of the population at large

It is important to analyze a dataset before using it for machine learning. Cloud Dataprep is useful for this stage of machine learning.

UNBALANCED TRAINING SETS

Some classification problems have trouble with *unbalanced datasets*, which are datasets that have significantly more instances of some categories than of others. For example, most credit card transactions are legitimate, but a small percentage are fraudulent. Let's assume that 2 percent of credit card transactions are fraudulent, and 98 percent are legitimate. If we trained a model that optimized for accuracy, then the model could predict that all transactions are legitimate, and it would have a 98 percent accuracy rate. However, it would essentially be useless.

You can deal with unbalanced datasets in several ways. You could undersample the majority class, which means that you would use fewer instances of the majority class than you would get with random sampling. Alternatively, you could oversample the minority class, in which case you would get more instances of the minority class than if you had randomly sampled. Another technique is *cost-sensitive learning*, which takes misclassification costs into account. For example, when building a decision tree, the training algorithm could take misclassification cost into account when determining a splitting factor. Hybrid algorithms use multiple strategies to compensate for unbalanced datasets.

In addition to altering how you train with unbalanced datasets, you should be careful to avoid using accuracy as an evaluation metric. A better metric for unbalanced datasets is the F1 score, which is described in the previous section, "Model Training and Evaluation."

TYPES OF BIAS

Bias is an overused word in machine learning. Sometimes, bias refers to an offset in a linear equation that adjusts the y-intercept. In other cases, it refers to a prejudice that favors something, but with merit. It is that type of bias that is addressed in this section.

Automation bias is a human bias in favor of decisions made by machines over those made by humans. These can occur as omission errors, which are when humans do not detect something missing from the machine decision. Automation bias can occur as a commission error in which humans do not detect an incorrect decision made by a machine.

Reporting bias occurs when a dataset does not accurately reflect the state of the world. Additional data should be collected until the distribution of data in a dataset reflects the distribution in the population as a whole. A related concept is *selection bias*, which is the process of choosing instances to include in ways that do not mirror the population. Selection bias by a human is a process that can result in reporting bias in a dataset.

Group attribution bias occurs when one generalizes a characteristic of an individual to a group as a whole. *In-group bias* occurs when someone favors a group to which they belong. *Out-group bias* occurs when invalid assumptions are made about a group of which the person is not a member.

When working to avoid bias, you can use tools such as the What If Tool for exploring datasets and detecting possible bias (`https://pair-code.github.io/what-if-tool/`).

NOTE Machine learning is a broad domain, spanning theoretical algorithm design to operationalizing production systems. Here are some additional resources that can help you prepare for the Professional Data Engineer exam:

- See Andrew Ng's "Machine Learning" course on Cousera; as of this writing, it had been viewed by 2.8 million students: `www.coursera.org/learn/machine-learning` (`http://www.coursera.org/learn/machine-learning`).
- See Eric Beck, Shanqing Cai, Eric Nielsen, Michael Salib, and D. Sculley, "The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction," at `https://research.google/pubs/pub46555/`.
- For more on supervised ML, see `https://en.wikipedia.org/wiki/Supervised_learning`.
- For more on unsupervised ML, see `https://en.wikipedia.org/wiki/Unsupervised_learning`.
- For more on reinforcement learning, see `https://en.wikipedia.org/wiki/Reinforcement_learning`.
- For more on deep learning, see Jianqing Fan, Cong Ma, and Yiqiao Zhong, see "Selective Overview of Deep Learning," at `https://arxiv.org/pdf/1904.05526.pdf`.

Exam Essentials

Know the three types of machine learning algorithms: supervised, unsupervised, and reinforcement learning. Supervised algorithms learn from labeled examples. Unsupervised learning starts with unlabeled data and identifies salient features, such as groups or clusters, and anomalies in a data stream. Reinforcement learning is a third type of machine learning algorithm that is distinct from supervised and unsupervised learning. It trains a model by interacting with its environment and receiving feedback on the decisions that it makes.

Know that supervised learning is used for classification and regression. Classification models assign discrete values to instances.

The simplest form is a binary classifier that assigns one of two values, such as fraudulent/not fraudulent, or has malignant tumor/does not have malignant tumor. Multiclass classification models assign more than two values. Regression models map continuous variables to other continuous variables.

Understand how unsupervised learning differs from supervised learning. Unsupervised learning algorithms find patterns in data without using predefined labels. Three types of unsupervised learning are clustering, anomaly detection, and collaborative filtering. Clustering, or cluster analysis, is the process of grouping instances together based on common features. Anomaly detection is the process of identifying unexpected patterns in data.

Understand how reinforcement learning differs from supervised and unsupervised techniques. Reinforcement learning is an approach to learning that uses agents interacting with an environment and adapting behavior based on rewards from the environment. This form of learning does not depend on labels. Reinforcement learning is modeled as an environment, a set of agents, a set of actions, and a set of probabilities of transitioning from one state to another after a particular action is taken. A reward is given after the transition from one state to another following an action.

Understand the structure of neural networks, particularly deep learning networks. Neural networks are systems roughly modeled after neurons in animal brains and consist of sets of connected artificial neurons or nodes. The network is composed of artificial neurons that are linked together into a network. The links between artificial neurons are called connections. A single neuron is limited in what it can learn. A multilayer network, however, is able to learn more functions. A multilayer neural network consists of a set of input nodes, hidden nodes, and an output layer.

Know machine learning terminology. This includes general machine learning terminology, such as baseline and batches; feature terminology, such as feature engineering and bucketing; training terminology, such as gradient descent and backpropagation; and neural network and deep learning terms, such as activation function and dropout. Finally, know model evaluation terminology such as precision and recall.

Know common sources of errors, including data-quality errors, unbalanced training sets, and bias. Poor-quality data leads to poor models. Some common data-quality problems are missing data, invalid values, inconsistent use of codes and categories, and data that is not representative of the population at large. Unbalanced datasets are ones that have significantly more instances of some categories than of others. There are several forms of bias, including automation bias, reporting bias, and group attribution.

Review Questions

You can find the answers in the appendix.

1. You are building a machine learning model to predict the sales price of houses. You have 7 years of historical data, including 18 features of houses and their sales price. What type of machine learning algorithm would you use?

   1. Classifier
   2. Regression
   3. Decision trees
   4. Reinforcement learning

2. You have been asked to build a machine learning model that will predict if a news article is a story about technology or another topic. Which of the following would you use?

   1. Logistic regression
   2. K-means clustering
   3. Simple linear regression
   4. Multiple linear regression

3. A startup is collecting IoT data from sensors placed on manufacturing equipment. The sensors send data every five seconds. The data includes a machine identifier, a timestamp, and several numeric values. The startup is developing a model to identify unusual readings. What type of unsupervised learning technique would they use?

   1. Clustering
   2. K-means
   3. Anomaly detection
   4. Reinforcement learning

4. You want to study deep learning and decide to start with the basics. You build a binary classifier using an artificial neuron. What algorithm would you use to train it?

1. Perceptron
2. SVM
3. Decision tree
4. Linear regression

5. A group of machine learning engineers has been assigned the task of building a machine learning model to predict the price of gold on the open market. Many features could be used, and the engineers believe that the optimal model will be complex. They want to understand the minimum predictive value of a model that they can build from the data that they have. What would they build?

1. Multiclass classifier
2. K clusters
3. Baseline model
4. Binary classifier

6. You are preparing a dataset to build a classifier. The data includes several continuous values, each in the range 0.00 to 100.00. You'd like to have a discrete feature derive each continuous value. What type of feature engineering would you use?

1. Bucketing
2. Dimension reduction
3. Principal component analysis
4. Gradient descent

7. You have been tasked with developing a classification model. You have reviewed the data that you will use for training and testing and realize that there are a number of outliers that you think might lead to overfitting. What technique would you use to reduce the impact of those outliers on the model?

1. Gradient descent
2. Large number of epochs
3. L2 regularization
4. Backpropagation

8. You have built a deep learning neural network that has 8 layers, and each layer has 100 fully connected nodes. The model fits the training data quite well with an F1 score of 98 out of 100. The model performs poorly when the test data is used, resulting in an F1 score of 62 out of 100. What technique would you use to try to improve performance of this model?

1. User more epochs
2. Dropout
3. Add more layers
4. ReLU

9. Your team is building a classifier to identify counterfeit products on an e-commerce site. Most of the products on the site are legitimate, and only about 3 percent of the products are counterfeit. You are concerned that, as is, the dataset will lead to a model that always predicts that products are legitimate. Which of the following techniques could you use to prevent this?

1. Undersampling
2. Dropout
3. L1 regularization
4. AUC

10. You are reviewing a dataset and find that the data is relatively high quality. There are no missing values and only a few outliers. You build a model based on the dataset that has high accuracy, precision, and recall when applied to the test data. When you use the model in production, however, it renders poor results. What might have caused this condition?

1. Applying L1 regularization
2. Applying dropout
3. Reporting bias
4. Automation bias

Support / Sign Out

PREV
Chapter 10 Choosing Training and Serving Infrastructure

NEXT
Chapter 12 Leveraging Prebuilt Models as a Service