aws

Developer Guide

Amazon Machine Learning

Version Latest

Amazon Machine Learning: Developer Guide

Table of Contents

We are no longer updating the Amazon Machine Learning service or accepting new users for it. This documentation is available for existing users, but we are no longer updating it. For more information, see What is Amazon Machine Learning.

What is Amazon Machine Learning?

We are no longer updating the Amazon Machine Learning (Amazon ML) service or accepting new users for it. This documentation is available for existing users, but we are no longer updating it.

AWS now provides a robust, cloud-based service — Amazon SageMaker AI — so that developers of all skill levels can use machine learning technology. SageMaker AI is a fully managed machine learning service that helps you create powerful machine learning models. With SageMaker AI, data scientists and developers can build and train machine learning models, and then directly deploy them into a production-ready hosted environment.

For more information, see the SageMaker AI documentation.

Topics

Amazon Machine Learning Key Concepts

This section summarizes the following key concepts and describes in greater detail how they are used within Amazon ML:

- Datasources contain metadata associated with data inputs to Amazon ML
- ML Models generate predictions using the patterns extracted from the input data
- Evaluations measure the quality of ML models
- Batch Predictions *asynchronously* generate predictions for multiple input data observations
- Real-time Predictions *synchronously* generate predictions for individual data observations

Datasources

A datasource is an object that contains metadata about your input data. Amazon ML reads your input data, computes descriptive statistics on its attributes, and stores the statistics—along with a schema and other information—as part of the datasource object. Next, Amazon ML uses the datasource to train and evaluate an ML model and generate batch predictions.

> **Important**
>
> A datasource does not store a copy of your input data. Instead, it stores a reference to the Amazon S3 location where your input data resides. If you move or change the Amazon S3 file, Amazon ML cannot access or use it to create a ML model, generate evaluations, or generate predictions.

The following table defines terms that are related to datasources.

Term	Definition
Attribute	A unique, named property within an observation. In tabular-formatted data such as spreadsheets or comma-separated values (CSV) files, the column headings represent the attributes, and the rows contain values for each attribute. Synonyms: variable, variable name, field, column
Datasource Name	(Optional) Allows you to define a human-readable name for a datasourc e. These names enable you to find and manage your datasources in the Amazon ML console.
Input Data	Collective name for all the observations that are referred to by a datasource.
Location	Location of input data. Currently, Amazon ML can use data that is stored within Amazon S3 buckets, Amazon Redshift databases, or MySQL databases in Amazon Relational Database Service (RDS).
Observation	A single input data unit. For example, if you are creating an ML model to detect fraudulent transactions, your input data will consist of many observations, each representing an individual transaction. Synonyms: record, example, instance, row
Row ID	(Optional) A flag that, if specified, identifies an attribute in the input data to be included in the prediction output. This attribute makes it easier to associate which prediction corresponds with which observation.

Term	Definition
	Synonyms: row identifier
Schema	The information needed to interpret the input data, including attribute names and their assigned data types, and names of special attributes.
Statistics	Summary statistics for each attribute in the input data. These statistics serve two purposes: The Amazon ML console displays them in graphs to help you understand your data at-a-glance and identify irregularities or errors. Amazon ML uses them during the training process to improve the quality of the resulting ML model.
Status	Indicates the current state of the datasource, such as **In Progress**, **Completed**, or **Failed**.
Target Attribute	In the context of training an ML model, the target attribute identifies the name of the attribute in the input data that contains the "correct" answers. Amazon ML uses this to discover patterns in the input data and generate an ML model. In the context of evaluating and generating predictions, the target attribute is the attribute whose value will be predicted by a trained ML model. Synonyms: target

ML Models

An ML model is a mathematical model that generates predictions by finding patterns in your data. Amazon ML supports three types of ML models: binary classification, multiclass classification and regression.

The following table defines terms that are related to ML models.

Term	Definition
Regression	The goal of training a regression ML model is to predict a numeric value.

Term	Definition
Multiclass	The goal of training a multiclass ML model is to predict values that belong to a limited, pre-defined set of permissible values.
Binary	The goal of training a binary ML model is to predict values that can only have one of two states, such as true or false.
Model Size	ML models capture and store patterns. The more patterns a ML model stores, the bigger it will be. ML model size is described in Mbytes.
Number of Passes	When you train an ML model, you use data from a datasource. It is sometimes beneficial to use each data record in the learning process more than once. The number of times that you let Amazon ML use the same data records is called the number of passes.
Regularization	Regularization is a machine learning technique that you can use to obtain higher-quality models. Amazon ML offers a default setting that works well for most cases.

Evaluations

An evaluation measures the quality of your ML model and determines if it is performing well.

The following table defines terms that are related to evaluations.

Term	Definition
Model Insights	Amazon ML provides you with a metric and a number of insights that you can use to evaluate the predictive performance of your model.
AUC	Area Under the ROC Curve (AUC) measures the ability of a binary ML model to predict a higher score for positive examples as compared to negative examples.
Macro-averaged F1-score	The macro-averaged F1-score is used to evaluate the predictive performance of multiclass ML models.

Term	Definition
RMSE	The Root Mean Square Error (RMSE) is a metric used to evaluate the predictive performance of regression ML models.
Cut-off	ML models work by generating numeric prediction scores. By applying a cut-off value, the system converts these scores into 0 and 1 labels.
Accuracy	Accuracy measures the percentage of correct predictions.
Precision	Precision shows the percentage of actual positive instances (as opposed to false positives) among those instances that have been retrieved (those predicted to be positive). In other words, how many selected items are positive?
Recall	Recall shows the percentage of actual positives among the total number of relevant instances (actual positives). In other words, how many positive items are selected?

Batch Predictions

Batch predictions are for a set of observations that can run all at once. This is ideal for predictive analyses that do not have a real-time requirement.

The following table defines terms that are related to batch predictions.

Term	Definition
Output Location	The results of a batch prediction are stored in an S3 bucket output location.
Manifest File	This file relates each input data file with its associated batch prediction results. It is stored in the S3 bucket output location.

Real-time Predictions

Real-time predictions are for applications with a low latency requirement, such as interactive web, mobile, or desktop applications. Any ML model can be queried for predictions by using the low latency real-time prediction API.

The following table defines terms that are related to real-time predictions.

Term	Definition
Real-time Prediction API	The Real-time Prediction API accepts a single input observation in the request payload and returns the prediction in the response.
Real-time Prediction Endpoint	To use an ML model with the real-time prediction API, you need to create a real-time prediction endpoint. Once created, the endpoint contains the URL that you can use to request real-time predictions.

Accessing Amazon Machine Learning

You can access Amazon ML by using any of the following:

Amazon ML console

You can access the Amazon ML console by signing into the AWS Management Console, and opening the Amazon ML console at https://console.aws.amazon.com/machinelearning/.

AWS CLI

For information about how to install and configure the AWS CLI, see Getting Set Up with the AWS Command Line Interface in the AWS Command Line Interface User Guide.

Amazon ML API

For more information about the Amazon ML API, see Amazon ML API Reference.

AWS SDKs

For more information about the AWS SDKs, see Tools for Amazon Web Services.

Regions and Endpoints

Amazon Machine Learning (Amazon ML) supports real-time prediction endpoints in the following two regions:

Region name	Region	Endpoint	Protocol
US East (N. Virginia)	us-east-1	machinelearning.us-east-1.amazonaws.com	HTTPS
Europe (Ireland)	eu-west-1	machinelearning.eu-west-1.amazonaws.com	HTTPS

You can host data sets, train and evaluate models, and trigger predictions in any region.

We recommend that you keep all of your resources in the same region. If your input data is in a different region than your Amazon ML resources, you accrue cross regional data transfer fees. You can call a real-time prediction endpoint from any region, but calling an endpoint from a region that does not have the endpoint that you're calling can impact real-time prediction latencies.

Pricing for Amazon ML

With AWS services, you pay only for what you use. There are no minimum fees and no upfront commitments.

Amazon Machine Learning (Amazon ML) charges an hourly rate for the compute time used to compute data statistics and train and evaluate models, and then you pay for the number of predictions generated for your application. For real-time predictions, you also pay an hourly reserved capacity charge based on the size of your model.

Amazon ML estimates the costs for predictions only in the Amazon ML console.

For more information about Amazon ML pricing, see *Amazon Machine Learning Pricing*.

Topics

Estimating Batch Prediction Cost

When you request batch predictions from an Amazon ML model using the Create Batch Prediction wizard, Amazon ML estimates the cost of these predictions. The method to compute the estimate varies based on the type of data that is available.

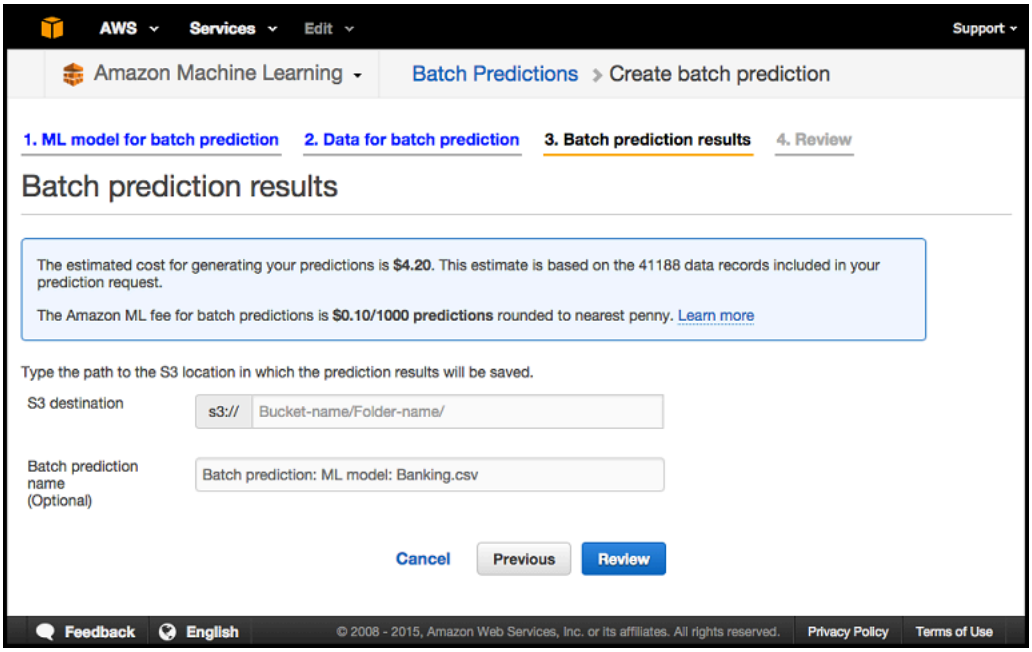
Estimating Batch Prediction Cost When Data Statistics Are Available

The most accurate cost estimate is obtained when Amazon ML has already computed summary statistics on the datasource used to request predictions. These statistics are always computed for datasources that have been created using the Amazon ML console. API users must set the `ComputeStatistics` flag to `True` when creating datasources programmatically using the CreateDataSourceFromS3, CreateDataSourceFromRedshift, or the CreateDataSourceFromRDS APIs. The datasource must be in the `READY` state for the statistics to be available.

One of the statistics that Amazon ML computes is the number of data records. When the number of data records is available, the Amazon ML Create Batch Prediction wizard estimates the number of predictions by multiplying the number of data records by the fee for batch predictions.

Your actual cost may vary from this estimate for the following reasons:

- Some of the data records might fail processing. You are not billed for predictions from failed data records.
- The estimate doesn't take into account pre-existing credits or other adjustments that are applied by AWS.



Estimating Batch Prediction Cost When Only Data Size Is Available

When you request a batch prediction and the data statistics for the request datasource are not available, Amazon ML estimates the cost based on the following:

- The total data size that is computed and persisted during datasource validation
- The average data record size, which Amazon ML estimates by reading and parsing the first 100 MB of your data file

To estimate the cost of your batch prediction, Amazon ML divides the total data size by the average data record size. This method of cost prediction is less precise than the method used when the number of data records is available because the first records of your data file might not accurately represent the average record size.

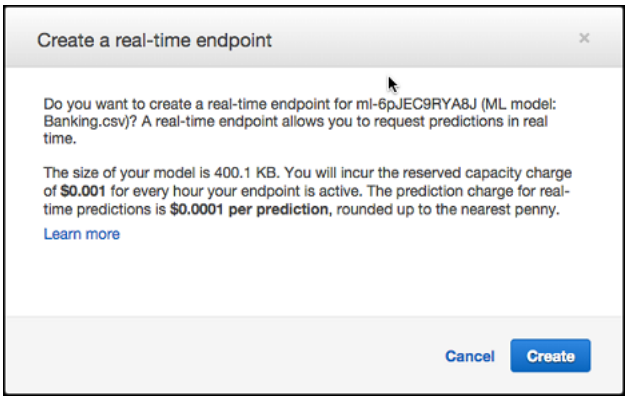
Estimating Batch Prediction Cost When Neither Data Statistics nor Data Size Are Available

When neither data statistics nor the data size are available, Amazon ML cannot estimate the cost of your batch predictions. This is commonly the case when the data source you are using to request batch predictions has not yet been validated by Amazon ML. This can happen when you have created a datasource that is based on an Amazon Redshift (Amazon Redshift) or Amazon Relational Database Service (Amazon RDS) query, and the data transfer has not yet completed, or when datasource creation is queued up behind other operations in your account. In this case, the

Amazon ML console informs you about the fees for batch prediction. You can choose to proceed with the batch prediction request without an estimate, or to cancel the wizard and return after the datasource used for predictions is in the INPROGRESS or READY state.

Estimating Real-Time Prediction Cost

When you create a real-time prediction endpoint using the Amazon ML console, you will be shown the estimated reserve capacity charge, which is an ongoing charge for reserving the endpoint for prediction processing. This charge varies based on the size of the model, as explained on the service pricing page. You will also be informed about the standard Amazon ML real-time prediction charge.



Machine Learning Concepts

Machine learning (ML) can help you use historical data to make better business decisions. ML algorithms discover patterns in data, and construct mathematical models using these discoveries. Then you can use the models to make predictions on future data. For example, one possible application of a machine learning model would be to predict how likely a customer is to purchase a particular product based on their past behavior.

Topics

Solving Business Problems with Amazon Machine Learning

You can use Amazon Machine Learning to apply machine learning to problems for which you have existing examples of actual answers. For example, if you want to use Amazon Machine Learning to predict if an email is spam, you will need to collect email examples that are correctly labeled as spam or not spam. You then can use machine learning to generalize from these email examples to predict how likely new email is spam or not. This approach of learning from data that has been labeled with the actual answer is known as supervised machine learning.

You can use supervised ML approaches for these specific machine learning tasks: binary classification (predicting one of two possible outcomes), multiclass classification (predicting one of more than two outcomes) and regression (predicting a numeric value).

Examples of binary classification problems:

- Will the customer buy this product or not buy this product?
- Is this email spam or not spam?
- Is this product a book or a farm animal?
- Is this review written by a customer or a robot?

Examples of multiclass classification problems:

- Is this product a book, movie, or clothing?
- Is this movie a romantic comedy, documentary, or thriller?
- Which category of products is most interesting to this customer?

Examples of regression classification problems:

- What will the temperature be in Seattle tomorrow?
- For this product, how many units will sell?
- How many days before this customer stops using the application?
- What price will this house sell for?

When to Use Machine Learning

It is important to remember that ML is not a solution for every type of problem. There are certain cases where robust solutions can be developed without using ML techniques. For example, you don't need ML if you can determine a target value by using simple rules, computations, or predetermined steps that can be programmed without needing any data-driven learning.

Use machine learning for the following situations:

- *You cannot code the rules:* Many human tasks (such as recognizing whether an email is spam or not spam) cannot be adequately solved using a simple (deterministic), rule-based solution. A large number of factors could influence the answer. When rules depend on too many factors and many of these rules overlap or need to be tuned very finely, it soon becomes difficult for a human to accurately code the rules. You can use ML to effectively solve this problem.
- *You cannot scale:* You might be able to manually recognize a few hundred emails and decide whether they are spam or not. However, this task becomes tedious for millions of emails. ML solutions are effective at handling large-scale problems.

Building a Machine Learning Application

Building ML applications is an iterative process that involves a sequence of steps. To build an ML application, follow these general steps:

1. Frame the core ML problem(s) in terms of what is observed and what answer you want the model to predict.

2. Collect, clean, and prepare data to make it suitable for consumption by ML model training algorithms. Visualize and analyze the data to run sanity checks to validate the quality of the data and to understand the data.
3. Often, the raw data (input variables) and answer (target) are not represented in a way that can be used to train a highly predictive model. Therefore, you typically should attempt to construct more predictive input representations or features from the raw variables.
4. Feed the resulting features to the learning algorithm to build models and evaluate the quality of the models on data that was held out from model building.
5. Use the model to generate predictions of the target answer for new data instances.

Formulating the Problem

The first step in machine learning is to decide what you want to predict, which is known as the label or target answer. Imagine a scenario in which you want to manufacture products, but your decision to manufacture each product depends on its number of potential sales. In this scenario, you want to predict how many times each product will be purchased (predict number of sales). There are multiple ways to define this problem by using machine learning. Choosing how to define the problem depends on your use case or business need.

Do you want to predict the number of purchases your customers will make for each product (in which case the target is numeric and you're solving a regression problem)? Or do you want to predict which products will get more than 10 purchases (in which case the target is binary and you're solving a binary classification problem)?

It is important to avoid over-complicating the problem and to frame the simplest solution that meets your needs. However, it is also important to avoid losing information, especially information in the historical answers. Here, converting an actual past sales number into a binary variable "over 10" versus "fewer" would lose valuable information. Investing time in deciding which target makes most sense for you to predict will save you from building models that don't answer your question.

Collecting Labeled Data

ML problems start with data—preferably, lots of data (examples or observations) for which you already know the target answer. Data for which you already know the target answer is called *labeled data*. In supervised ML, the algorithm teaches itself to learn from the labeled examples that we provide.

Each example/observation in your data must contain two elements:

- The target – The answer that you want to predict. You provide data that is labeled with the target (correct answer) to the ML algorithm to learn from. Then, you will use the trained ML model to predict this answer on data for which you do not know the target answer.
- Variables/features – These are attributes of the example that can be used to identify patterns to predict the target answer.

For example, for the email classification problem, the target is a label that indicates whether an email is spam or not spam. Examples of variables are the sender of the email, the text in the body of the email, the text in the subject line, the time the email was sent, and the existence of previous correspondence between the sender and receiver.

Often, data is not readily available in a labeled form. Collecting and preparing the variables and the target are often the most important steps in solving an ML problem. The example data should be representative of the data that you will have when you are using the model to make a prediction. For example, if you want to predict whether an email is spam or not, you must collect both positive (spam emails) and negative (non-spam emails) for the machine learning algorithm to be able to find patterns that will distinguish between the two types of email.

Once you have the labelled data, you might need to convert it to a format that is acceptable to your algorithm or software. For example, to use Amazon ML you need to convert the data to comma-separated (CSV) format with each example making up one row of the CSV file, each column containing one input variable, and one column containing the target answer.

Analyzing Your Data

Before feeding your labeled data to an ML algorithm, it is a good practice to inspect your data to identify issues and gain insights about the data you are using. The predictive power of your model will only be as good as the data you feed it.

When analyzing your data, you should keep the following considerations in mind:

- Variable and target data summaries – It is useful to understand the values that your variables take and which values are dominant in your data. You could run these summaries by a subject matter expert for the problem that you want to solve. Ask yourself or the subject matter expert: Does the data match your expectations? Does it look like you have a data collection problem? Is one class in your target more frequent than the other classes? Are there more missing values or invalid data than you expect?

- Variable-target correlations – Knowing the correlation between each variable and the target class is helpful because a high correlation implies that there is a relationship between the variable and the target class. In general, you want to include variables with high correlation because they are the ones with higher predictive power (signal), and leave out variables with low correlation because they are likely irrelevant.

In Amazon ML, you can analyze your data by creating a data source and by reviewing the resulting data report.

Feature Processing

After getting to know your data through data summaries and visualizations, you might want to transform your variables further to make them more meaningful. This is known as *feature processing*. For example, say you have a variable that captures the date and time at which an event occurred. This date and time will never occur again and hence won't be useful to predict your target. However, if this variable is transformed into features that represent the hour of the day, the day of the week, and the month, these variables could be useful to learn if the event tends to happen at a particular hour, weekday, or month. Such feature processing to form more generalizable data points to learn from can provide significant improvements to the predictive models.

Other examples of common feature processing:

- Replacing missing or invalid data with more meaningful values (e.g., if you know that a missing value for a product type variable actually means it is a book, you can then replace all missing values in the product type with the value for book). A common strategy used to impute missing values is to replace missing values with the mean or median value. It is important to understand your data before choosing a strategy for replacing missing values.
- Forming Cartesian products of one variable with another. For example, if you have two variables, such as population density (urban, suburban, rural) and state (Washington, Oregon, California), there might be useful information in the features formed by a Cartesian product of these two variables resulting in features (urban_Washington, suburban_Washington, rural_Washington, urban_Oregon, suburban_Oregon, rural_Oregon, urban_California, suburban_California, rural_California).
- Non-linear transformations such as binning numeric variables to categories. In many cases, the relationship between a numeric feature and the target is not linear (the feature value does not increase or decrease monotonically with the target). In such cases, it might be useful to bin the

numeric feature into categorical features representing different ranges of the numeric feature. Each categorical feature (bin) can then be modeled as having its own linear relationship with the target. For example, say you know that the continuous numeric feature age is not linearly correlated with the likelihood to purchase a book. You can bin age into categorical features that might be able to capture the relationship with the target more accurately. The optimum number of bins for a numeric variable is dependent on characteristics of the variable and its relationship to the target, and this is best determined through experimentation. Amazon ML suggests the optimal bin number for a numeric feature based on data statistics in the suggested recipe. See the Developer Guide for details about the *suggested recipe*.

- Domain-specific features (e.g., you have length, breadth, and height as separate variables; you can create a new volume feature to be a product of these three variables).
- Variable-specific features. Some variable types such as text features, features that capture the structure of a web page, or the structure of a sentence have generic ways of processing that help extract structure and context. For example, forming *n-grams* from text "the fox jumped over the fence" can be represented with *unigrams*: the, fox, jumped, over, fence or *bigrams*: the fox, fox jumped, jumped over, over the, the fence.

Including more relevant features helps to improve prediction power. Clearly, it is not always possible to know the features with "signal" or predictive influence in advance. So it is good to include all features that can potentially be related to the target label and let the model training algorithm pick the features with the strongest correlations. In Amazon ML, feature processing can be specified in the recipe when creating a model. See the Developer Guide for a list of available feature processors.

Splitting the Data into Training and Evaluation Data

The fundamental goal of ML is to *generalize* beyond the data instances used to train models. We want to evaluate the model to estimate the quality of its pattern generalization for data the model has not been trained on. However, because future instances have unknown target values and we cannot check the accuracy of our predictions for future instances now, we need to use some of the data that we already know the answer for as a proxy for future data. Evaluating the model with the same data that was used for training is not useful, because it rewards models that can "remember" the training data, as opposed to generalizing from it.

A common strategy is to take all available labeled data, and split it into training and evaluation subsets, usually with a ratio of 70-80 percent for training and 20-30 percent for evaluation. The ML system uses the training data to train models to see patterns, and uses the evaluation

data to evaluate the predictive quality of the trained model. The ML system evaluates predictive performance by comparing predictions on the evaluation data set with true values (known as ground truth) using a variety of metrics. Usually, you use the “best” model on the evaluation subset to make predictions on future instances for which you do not know the target answer.

Amazon ML splits data sent for training a model through the Amazon ML console into 70 percent for training and 30 percent for evaluation. By default, Amazon ML uses the first 70 percent of the input data in the order it appears in the source data for the training datasource and the remaining 30 percent of the data for the evaluation datasource. Amazon ML also allows you to select a random 70 percent of the source data for training instead of using the first 70 percent, and using the complement of this random subset for evaluation. You can use Amazon ML APIs to specify custom split ratios and to provide training and evaluation data that was split outside of Amazon ML. Amazon ML also provides strategies for splitting your data. For more information on splitting strategies, see Splitting Your Data.

Training the Model

You are now ready to provide the ML algorithm (that is, the *learning algorithm*) with the training data. The algorithm will learn from the training data patterns that map the variables to the target, and it will output a model that captures these relationships. The ML model can then be used to get predictions on new data for which you do not know the target answer.

Linear Models

There are a large number of ML models available. Amazon ML learns one type of ML model: linear models. The term linear model implies that the model is specified as a linear combination of features. Based on training data, the learning process computes one weight for each feature to form a model that can predict or estimate the target value. For example, if your target is the amount of insurance a customer will purchase and your variables are age and income, a simple linear model would be the following:

```
Estimated target = 0.2 + 5·age + 0.0003·income
```

Learning Algorithm

The learning algorithm’s task is to learn the weights for the model. The weights describe the likelihood that the patterns that the model is learning reflect actual relationships in the data. A learning algorithm consists of a loss function and an optimization technique. The loss is the penalty that is incurred when the estimate of the target provided by the ML model does not

equal the target exactly. A loss function quantifies this penalty as a single value. An optimization technique seeks to minimize the loss. In Amazon Machine Learning, we use three loss functions, one for each of the three types of prediction problems. The optimization technique used in Amazon ML is online Stochastic Gradient Descent (SGD). SGD makes sequential passes over the training data, and during each pass, updates feature weights one example at a time with the aim of approaching the optimal weights that minimize the loss.

Amazon ML uses the following learning algorithms:

- For binary classification, Amazon ML uses logistic regression (logistic loss function + SGD).
- For multiclass classification, Amazon ML uses multinomial logistic regression (multinomial logistic loss + SGD).
- For regression, Amazon ML uses linear regression (squared loss function + SGD).

Training Parameters

The Amazon ML learning algorithm accepts parameters, called hyperparameters or training parameters, that allow you to control the quality of the resulting model. Depending on the hyperparameter, Amazon ML auto-selects settings or provides static defaults for the hyperparameters. Although default hyperparameter settings generally produce useful models, you might be able to improve the predictive performance of your models by changing hyperparameter values. The following sections describe common hyperparameters associated with learning algorithms for linear models, such as those created by Amazon ML.

Learning Rate

The learning rate is a constant value used in the Stochastic Gradient Descent (SGD) algorithm. Learning rate affects the speed at which the algorithm reaches (converges to) the optimal weights. The SGD algorithm makes updates to the weights of the linear model for every data example it sees. The size of these updates is controlled by the learning rate. Too large a learning rate might prevent the weights from approaching the optimal solution. Too small a value results in the algorithm requiring many passes to approach the optimal weights.

In Amazon ML, the learning rate is auto-selected based on your data.

Model Size

If you have many input features, the number of possible patterns in the data can result in a large model. Large models have practical implications, such as requiring more RAM to hold the model

while training and when generating predictions. In Amazon ML, you can reduce the model size by using L1 regularization or by specifically restricting the model size by specifying the maximum size. Note that if you reduce the model size too much, you could reduce your model's predictive power.

For information about the default model size, see Training Parameters: Types and Default Values. For more information about regularization, see Regularization.

Number of Passes

The SGD algorithm makes sequential passes over the training data. The `Number of passes` parameter controls the number of passes that the algorithm makes over the training data. More passes result in a model that fits the data better (if the learning rate is not too large), but the benefit diminishes with an increasing number of passes. For smaller data sets, you can significantly increase the number of passes, which allows the learning algorithm to effectively fit the data more closely. For extremely large datasets, a single pass might suffice.

For information about the default number of passes, see Training Parameters: Types and Default Values.

Data Shuffling

In Amazon ML, you must shuffle your data because the SGD algorithm is influenced by the order of the rows in the training data. Shuffling your training data results in better ML models because it helps the SGD algorithm avoid solutions that are optimal for the first type of data it sees, but not for the full range of data. Shuffling mixes up the order of your data so that the SGD algorithm doesn't encounter one type of data for too many observations in succession. If it sees only one type of data for many successive weight updates, the algorithm might not be able to correct the model weights for a new data type because the update might be too large. Additionally, when the data isn't presented randomly, it's difficult for the algorithm to find the optimal solution for all of the data types quickly; in some cases, the algorithm might never find the optimal solution. Shuffling the training data helps the algorithm to converge on the optimal solution sooner.

For example, say you want to train an ML model to predict a product type, and your training data includes movie, toy, and video game product types. If you sort the data by the product type column before uploading the data to Amazon S3, then the algorithm sees the data alphabetically by product type. The algorithm sees all of your data for movies first, and your ML model begins to learn patterns for movies. Then, when your model encounters data on toys, every update that the algorithm makes would fit the model to the toy product type, even if those updates degrade the patterns that fit movies. This sudden switch from movie to toy type can produce a model that doesn't learn how to predict product types accurately.

For information about the default shuffling type, see Training Parameters: Types and Default Values.

Regularization

Regularization helps prevent linear models from overfitting training data examples (that is, memorizing patterns instead of generalizing them) by penalizing extreme weight values. L1 regularization has the effect of reducing the number of features used in the model by pushing to zero the weights of features that would otherwise have small weights. As a result, L1 regularization results in sparse models and reduces the amount of noise in the model. L2 regularization results in smaller overall weight values, and stabilizes the weights when there is high correlation between the input features. You control the amount of L1 or L2 regularization applied by using the `Regularization type` and `Regularization amount` parameters. An extremely large regularization value could result in all features having zero weights, preventing a model from learning patterns.

For information about the default regularization values, see Training Parameters: Types and Default Values.

Evaluating Model Accuracy

The goal of the ML model is to learn patterns that generalize well for unseen data instead of just memorizing the data that it was shown during training. Once you have a model, it is important to check if your model is performing well on unseen examples that you have not used for training the model. To do this, you use the model to predict the answer on the evaluation dataset (held out data) and then compare the predicted target to the actual answer (ground truth).

A number of metrics are used in ML to measure the predictive accuracy of a model. The choice of accuracy metric depends on the ML task. It is important to review these metrics to decide if your model is performing well.

Binary Classification

The actual output of many binary classification algorithms is a prediction score. The score indicates the system's certainty that the given observation belongs to the positive class. To make the decision about whether the observation should be classified as positive or negative, as a consumer of this score, you will interpret the score by picking a classification threshold (cut-off) and compare the score against it. Any observations with scores higher than the threshold are then predicted as the positive class and scores lower than the threshold are predicted as the negative class.

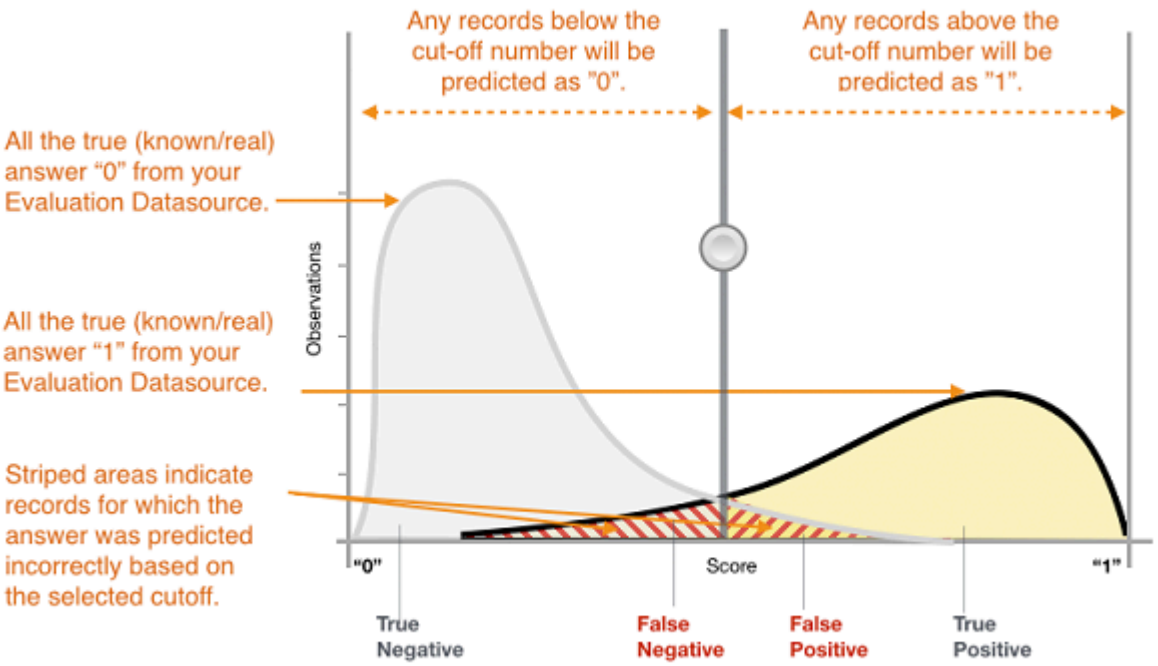


Figure 1: Score Distribution for a Binary Classification Model

The predictions now fall into four groups based on the actual known answer and the predicted answer: correct positive predictions (true positives), correct negative predictions (true negatives), incorrect positive predictions (false positives) and incorrect negative predictions (false negatives).

Binary classification accuracy metrics quantify the two types of correct predictions and two types of errors. Typical metrics are accuracy (ACC), precision, recall, false positive rate, F1-measure. Each metric measures a different aspect of the predictive model. Accuracy (ACC) measures the fraction of correct predictions. Precision measures the fraction of actual positives among those examples that are predicted as positive. Recall measures how many actual positives were predicted as positive. F1-measure is the harmonic mean of precision and recall.

AUC is a different type of metric. It measures the ability of the model to predict a higher score for positive examples as compared to negative examples. Since AUC is independent of the selected threshold, you can get a sense of the prediction performance of your model from the AUC metric without picking a threshold.

Depending on your business problem, you might be more interested in a model that performs well for a specific subset of these metrics. For example, two business applications might have very different requirements for their ML models:

- One application might need to be extremely sure about the positive predictions actually being positive (high precision) and be able to afford to misclassify some positive examples as negative (moderate recall).
- Another application might need to correctly predict as many positive examples as possible (high recall) and will accept some negative examples being misclassified as positive (moderate precision).

In Amazon ML, observations get a predicted score in the range [0,1]. The score threshold to make the decision of classifying examples as 0 or 1 is set by default to be 0.5. Amazon ML allows you to review the implications of choosing different score thresholds and allows you to pick an appropriate threshold that matches your business need.

Multiclass Classification

Unlike the process for binary classification problems, you do not need to choose a score threshold to make predictions. The predicted answer is the class (i.e., label) with the highest predicted score. In some cases, you might want to use the predicted answer only if it is predicted with a high score. In this case, you might choose a threshold on the predicted scores based on which you will accept the predicted answer or not.

Typical metrics used in multiclass are the same as the metrics used in the binary classification case. The metric is calculated for each class by treating it as a binary classification problem after grouping all the other classes as belonging to the second class. Then the binary metric is averaged over all the classes to get either a macro average (treat each class equally) or weighted average (weighted by class frequency) metric. In Amazon ML, the macro average F1-measure is used to evaluate the predictive success of a multiclass classifier.

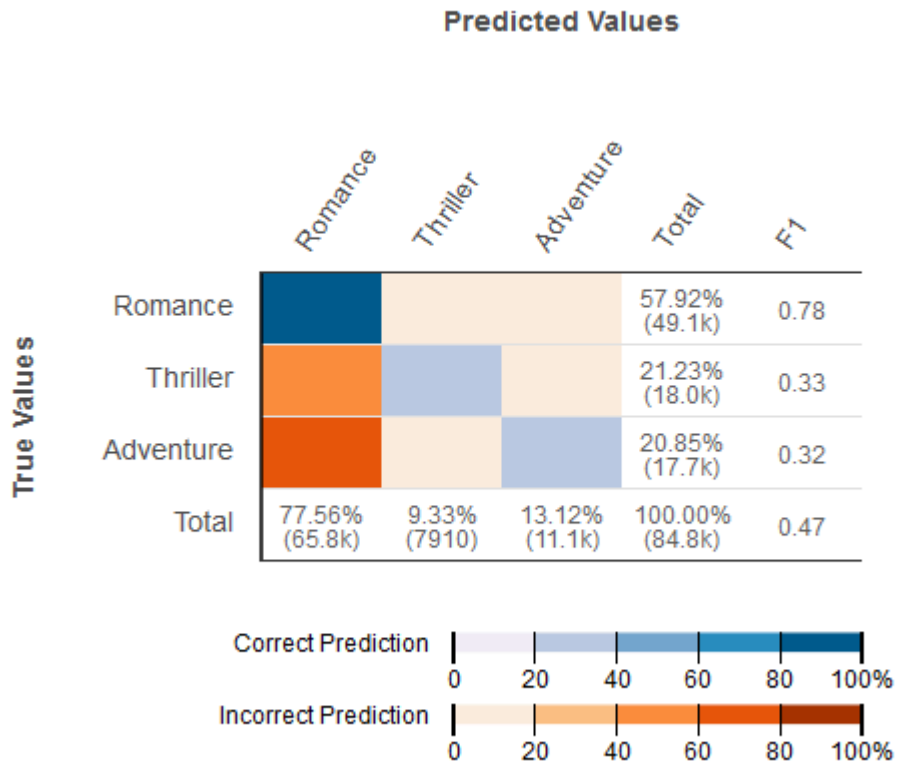


Figure 2: Confusion Matrix for a multiclass classification model

It is useful to review the *confusion matrix* for multiclass problems. The confusion matrix is a table that shows each class in the evaluation data and the number or percentage of correct predictions and incorrect predictions.

Regression

For regression tasks, the typical accuracy metrics are root mean square error (RMSE) and mean absolute percentage error (MAPE). These metrics measure the distance between the predicted numeric target and the actual numeric answer (ground truth). In Amazon ML, the RMSE metric is used to evaluate the predictive accuracy of a regression model.

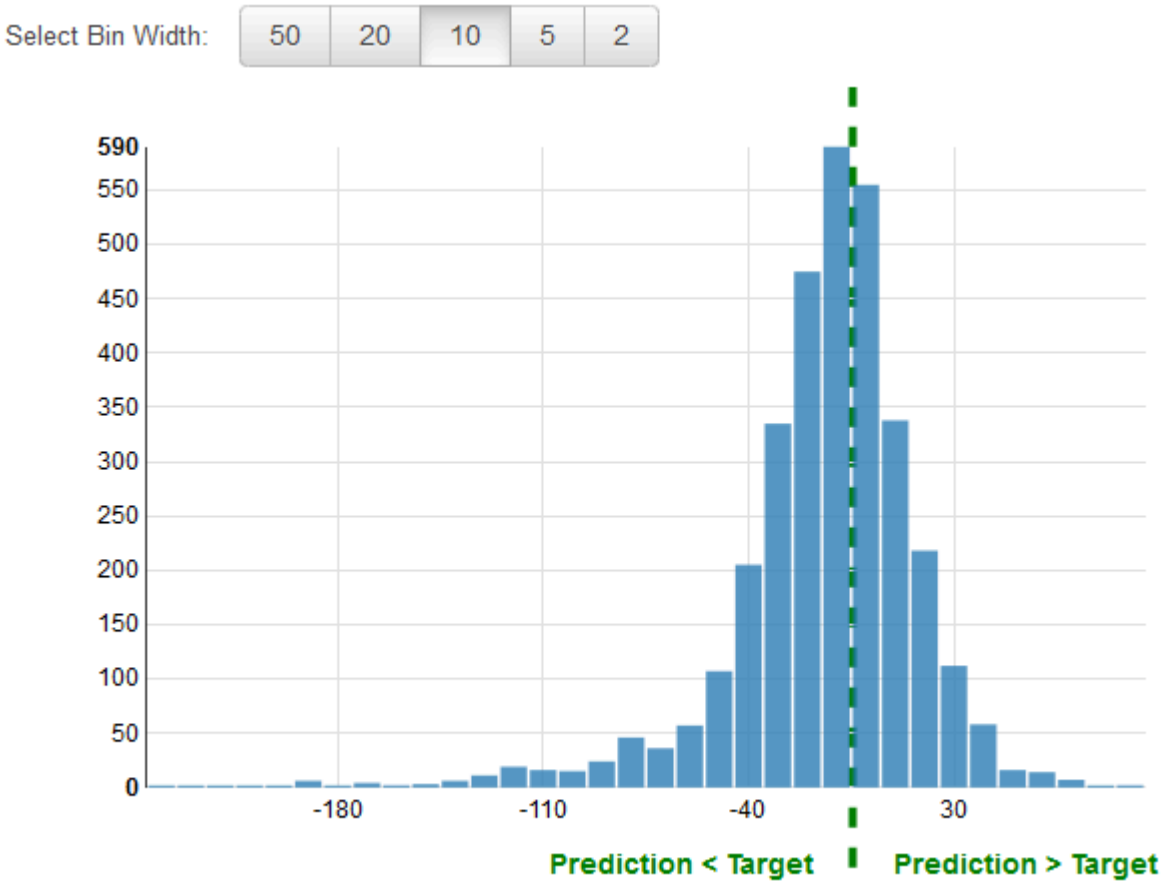


Figure 3: Distribution of residuals for a Regression model

It is common practice to review the *residuals* for regression problems. A residual for an observation in the evaluation data is the difference between the true target and the predicted target. Residuals represent the portion of the target that the model is unable to predict. A positive residual indicates that the model is underestimating the target (the actual target is larger than the predicted target). A negative residual indicates an overestimation (the actual target is smaller than the predicted target). The histogram of the residuals on the evaluation data when distributed in a bell shape and centered at zero indicates that the model makes mistakes in a random manner and does not systematically over or under predict any particular range of target values. If the residuals do not form a zero-centered bell shape, there is some structure in the model's prediction error. Adding more variables to the model might help the model capture the pattern that is not captured by the current model.

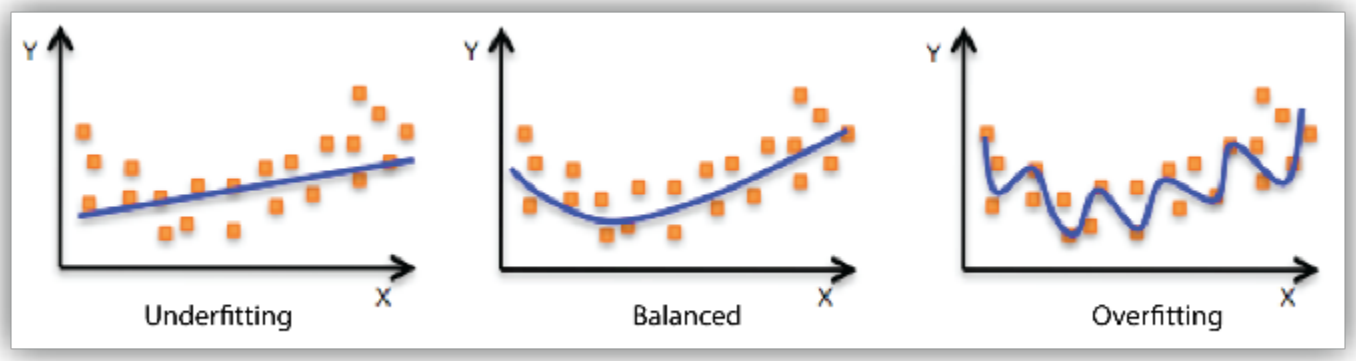
Improving Model Accuracy

Obtaining a ML model that matches your needs usually involves iterating through this ML process and trying out a few variations. You might not get a very predictive model in the first iteration, or you might want to improve your model to get even better predictions. To improve performance, you could iterate through these steps:

1. Collect data: Increase the number of training examples
2. Feature processing: Add more variables and better feature processing
3. Model parameter tuning: Consider alternate values for the training parameters used by your learning algorithm

Model Fit: Underfitting vs. Overfitting

Understanding model fit is important for understanding the root cause for poor model accuracy. This understanding will guide you to take corrective steps. We can determine whether a predictive model is underfitting or overfitting the training data by looking at the prediction error on the training data and the evaluation data.



Your model is *underfitting* the training data when the model performs poorly on the training data. This is because the model is unable to capture the relationship between the input examples (often called X) and the target values (often called Y). Your model is *overfitting* your training data when you see that the model performs well on the training data but does not perform well on the evaluation data. This is because the model is memorizing the data it has seen and is unable to generalize to unseen examples.

Poor performance on the training data could be because the model is too simple (the input features are not expressive enough) to describe the target well. Performance can be improved by increasing model flexibility. To increase model flexibility, try the following:

- Add new domain-specific features and more feature Cartesian products, and change the types of feature processing used (e.g., increasing n-grams size)
- Decrease the amount of regularization used

If your model is overfitting the training data, it makes sense to take actions that reduce model flexibility. To reduce model flexibility, try the following:

- Feature selection: consider using fewer feature combinations, decrease n-grams size, and decrease the number of numeric attribute bins.
- Increase the amount of regularization used.

Accuracy on training and test data could be poor because the learning algorithm did not have enough data to learn from. You could improve performance by doing the following:

- Increase the amount of training data examples.
- Increase the number of passes on the existing training data.

Using the Model to Make Predictions

Now that you have an ML model that performs well, you will use it to make predictions. In Amazon Machine Learning, there are two ways to use a model to make predictions:

Batch Predictions

Batch prediction is useful when you want to generate predictions for a set of observations all at once, and then take action on a certain percentage or number of the observations. Typically, you do not have a low latency requirement for such an application. For example, when you want to decide which customers to target as part of an advertisement campaign for a product, you will get prediction scores for all customers, sort your model's predictions to identify which customers are most likely to purchase, and then target maybe the top 5% customers that are most likely to purchase.

Online Predictions

Online prediction scenarios are for cases when you want to generate predictions on a one-by-one basis for each example independent of the other examples, in a low-latency environment. For example, you could use predictions to make immediate decisions about whether a particular transaction is likely to be a fraudulent transaction.

Retraining Models on New Data

For a model to predict accurately, the data that it is making predictions on must have a similar distribution as the data on which the model was trained. Because data distributions can be expected to drift over time, deploying a model is not a one-time exercise but rather a continuous process. It is a good practice to continuously monitor the incoming data and retrain your model on newer data if you find that the data distribution has deviated significantly from the original training data distribution. If monitoring data to detect a change in the data distribution has a high overhead, then a simpler strategy is to train the model periodically, for example, daily, weekly, or monthly. In order to retrain models in Amazon ML, you would need to create a new model based on your new training data.

The Amazon Machine Learning Process

The following table describes how to use the Amazon ML console to perform the ML process outlined in this document.

ML Process	Amazon ML Task
Analyze your data	To analyze your data in Amazon ML, create a datasource and review the data insights page.
Split data into training and evaluation datasources	Amazon ML can split the datasource to use 70% of the data for model training and 30% for evaluating your model's predictive performance. When you use the Create ML Model wizard with the default settings, Amazon ML splits the data for you. If you use the Create ML Model wizard with the custom settings, and choose to evaluate the ML model, you will see an option for allowing

ML Process	Amazon ML Task
	Amazon ML to split the data for you and run an evaluation on 30% of the data.
Shuffle your training data	When you use the Create ML Model wizard with the default settings, Amazon ML shuffles your data for you. You can also shuffle your data before importing it into Amazon ML.
Process features	The process of putting together training data in an optimal format for learning and generalization is known as feature transformation. When you use the Create ML Model wizard with default settings, Amazon ML suggests feature processing settings for your data. To specify feature processing settings, use the Create ML Model wizard's **Custom** option and provide a feature processing recipe.
Train the model	When you use the Create ML Model wizard to create a model in Amazon ML, Amazon ML trains your model.
Select model parameters	In Amazon ML, you can tune four parameters that affect your model's predictive performance: model size, number of passes, type of shuffling , and regularization. You can set these parameters when you use the Create ML Model wizard to create an ML model and choose the **Custom** option.
Evaluate the model performance	Use the Create Evaluation wizard to assess your model's predictive performance.
Feature selection	The Amazon ML learning algorithm can drop features that don't contribute much to the learning process. To indicate that you want to drop those features, choose the `L1 regularization` parameter when you create the ML model.
Set a score threshold for prediction accuracy	Review the model's predictive performance in the evaluation report at different score thresholds, and then set the score threshold based on your business application. The score threshold determines how the model defines a prediction match. Adjust the number to control false positives and false negatives.

ML Process	Amazon ML Task
Use the model	Use your model to get predictions for a batch of observations by using the Create Batch Prediction wizard. Or, get predictions for individual observations on demand by enabling the ML model to process real-time predictions using the `Predict` API.

Setting Up Amazon Machine Learning

You need an AWS account before you can use Amazon Machine Learning for the first time. If you don't have an account, see Sign Up for AWS.

Sign Up for AWS

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all services in AWS, including Amazon ML. You are charged only for the services that you use. If you have an AWS account already, skip this step. If you don't have an AWS account, use the following procedure to create one.

To sign up for an AWS account

1. Go to http://aws.amazon.com and choose **Sign Up**.
2. Follow the on-screen instructions.

Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

Tutorial: Using Amazon ML to Predict Responses to a Marketing Offer

With Amazon Machine Learning (Amazon ML), you can build and train predictive models and host your applications in a scalable cloud solution. In this tutorial, we show you how to use the Amazon ML console to create a datasource, build a machine learning (ML) model, and use the model to generate predictions that you can use in your applications.

Our sample exercise shows how to identify potential customers for a targeted marketing campaign, but you can apply the same principles to create and use a variety of ML models. To complete the sample exercise, you will use publicly available banking and marketing datasets from the University of California at Irvine (UCI) Machine Learning Repository. These datasets contain general information about customers, and information about how they responded to previous marketing contacts. You will use this data to identify which customers are most likely to subscribe to your new product, a bank term deposit, also known as a certificate of deposit (CD).

> **Warning**
>
> This tutorial is not included in the AWS free tier. For more information about Amazon ML pricing, see *Amazon Machine Learning Pricing*.

Prerequisite

To perform the tutorial, you need to have an AWS account. If you don't have an AWS account, see Setting Up Amazon Machine Learning.

Steps

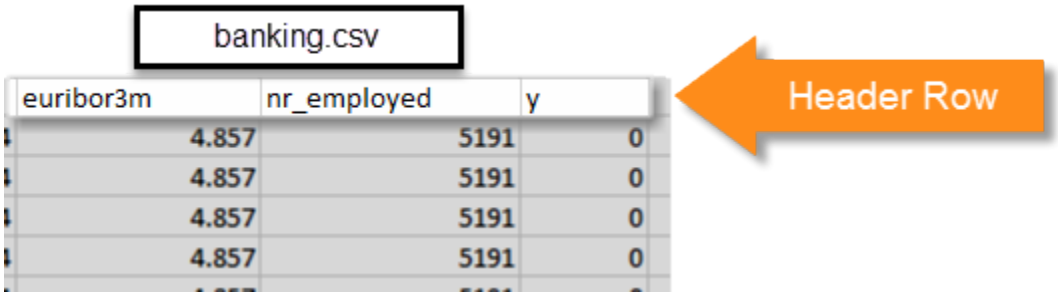
Step 1: Prepare Your Data

In machine learning, you typically obtain the data and ensure that it is well formatted before starting the training process. For the purposes of this tutorial, we obtained a sample dataset from the UCI Machine Learning Repository, formatted it to conform to Amazon ML guidelines, and made it available for you to download. Download the dataset from our Amazon Simple Storage Service (Amazon S3) storage location and upload it to your own S3 bucket by following the procedures in this topic.

For Amazon ML formatting requirements, see Understanding the Data Format for Amazon ML.

To download the datasets

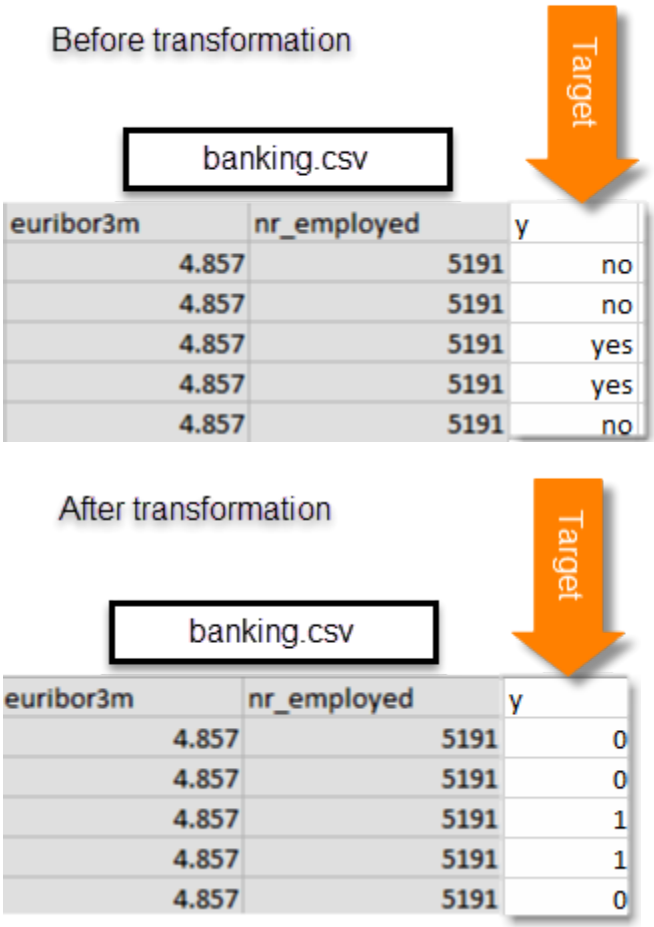
1. Download the file that contains the historical data for customers who have purchased products similar to your bank term deposit by clicking banking.zip. Unzip the folder and save the banking.csv file to your computer.
2. Download the file that you will use to predict whether potential customers will respond to your offer by clicking banking-batch.zip. Unzip the folder and save the banking-batch.csv file to your computer.
3. Open `banking.csv`. You will see rows and columns of data. The *header row* contains the attribute names for each column. An *attribute* is a unique, named property that describes a particular characteristic of each customer; for example, nr_employed indicates the customer's employment status. Each row represents the collection of observations about a single customer.



You want your ML model to answer the question "Will this customer subscribe to my new product?". In the `banking.csv` dataset, the answer to this question is attribute **y**, which contains the values 1 (for yes) or 0 (for no). The attribute that you want Amazon ML to learn how to predict is known as the *target attribute*.

> **Note**
>
> Attribute **y** is a binary attribute. It can contain only one of two values, in this case 0 or 1. In the original UCI dataset, the **y** attribute is either Yes or No. We have edited the original dataset for you. All values of attribute **y** that mean yes are now 1, and all values that mean no are now 0. If you use your own data, you can use other values for a binary attribute. For more information about valid values, see Using the AttributeType Field.

The following examples show the data before and after we changed the values in attribute **y** to the binary attributes 0 and 1.



The `banking-batch.csv` file doesn't contain the **y** attribute. After you have created an ML model, you will use the model to predict **y** for each record in that file.

Next, upload the `banking.csv` and `banking-batch.csv` files to Amazon S3.

To upload the files to an Amazon S3 location

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **All Buckets** list, create a bucket or choose the location where you want to upload the files.
3. In the navigation bar, choose **Upload**.
4. Choose **Add Files**.
5. In the dialog box, navigate to your desktop, choose `banking.csv` and `banking-batch.csv`, and then choose **Open**.

Now you are ready to create your training datasource.

Step 2: Create a Training Datasource

After you upload the `banking.csv` dataset to your Amazon Simple Storage Service (Amazon S3) location, you use it to create a training datasource. A datasource is an Amazon Machine Learning (Amazon ML) object that contains the location of your input data and important metadata about your input data. Amazon ML uses the datasource for operations like ML model training and evaluation.

To create a datasource, provide the following:

- The Amazon S3 location of your data and permission to access the data
- The schema, which includes the names of the attributes in the data and the type of each attribute (Numeric, Text, Categorical, or Binary)
- The name of the attribute that contains the answer that you want Amazon ML to learn to predict, the target attribute

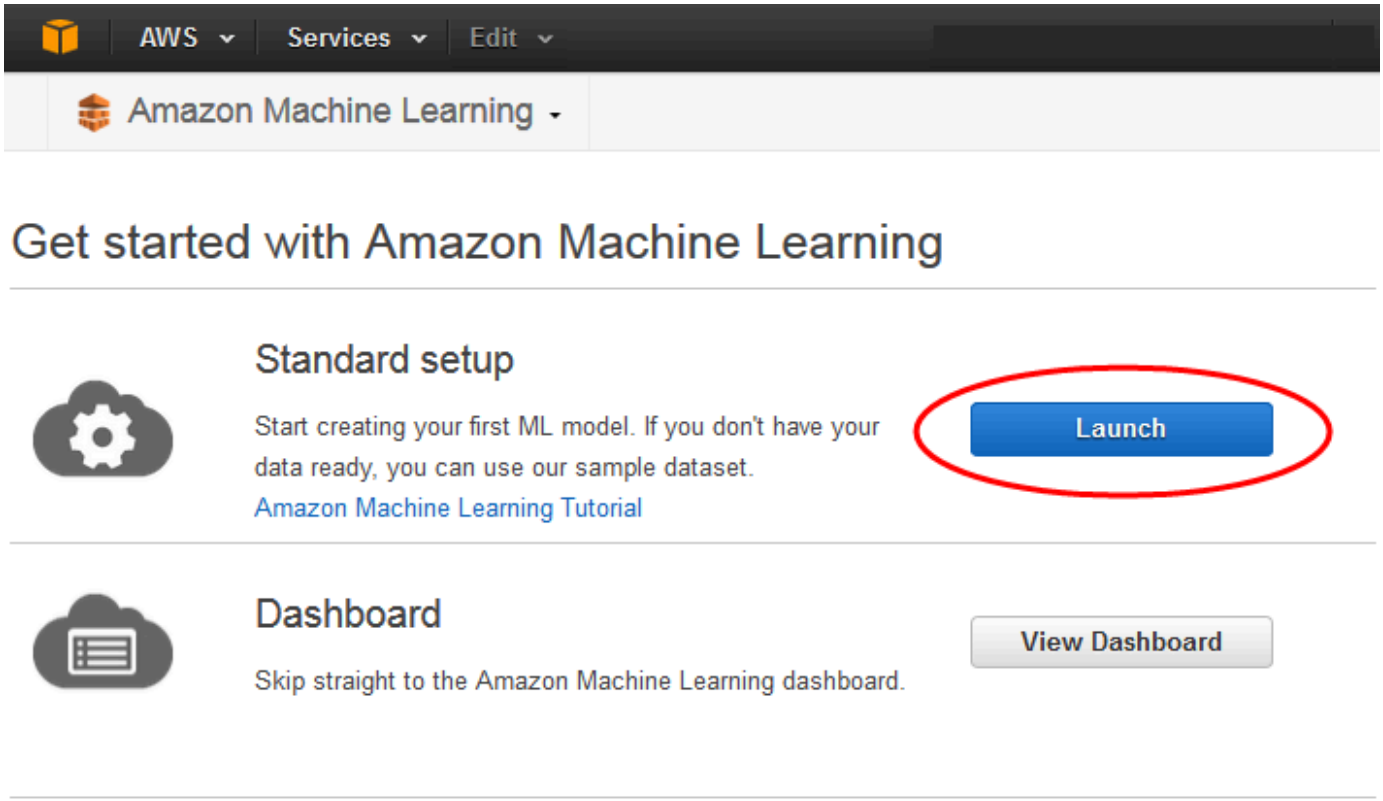
> **Note**
>
> The datasource doesn't actually store your data, it only references it. Avoid moving or changing the files stored in Amazon S3. If you do move or change them, Amazon ML can't access them to create an ML model, generate evaluations, or generate predictions.

To create the training datasource

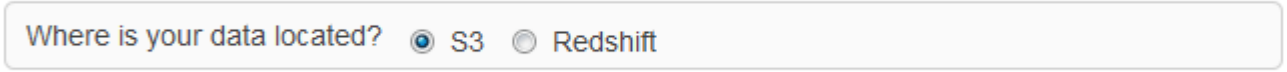
1. Open the Amazon Machine Learning console at https://console.aws.amazon.com/machinelearning/.
2. Choose **Get started**.

> **Note**
>
> This tutorial assumes that this is your first time using Amazon ML. If you have used Amazon ML before, you can use the **Create new...** drop down list on the Amazon ML dashboard to create a new datasource.

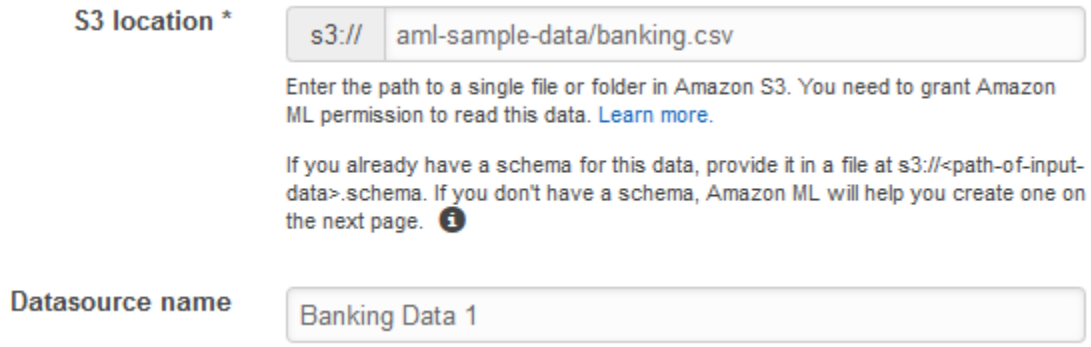
3. On the **Get started with Amazon Machine Learning** page, choose **Launch**.



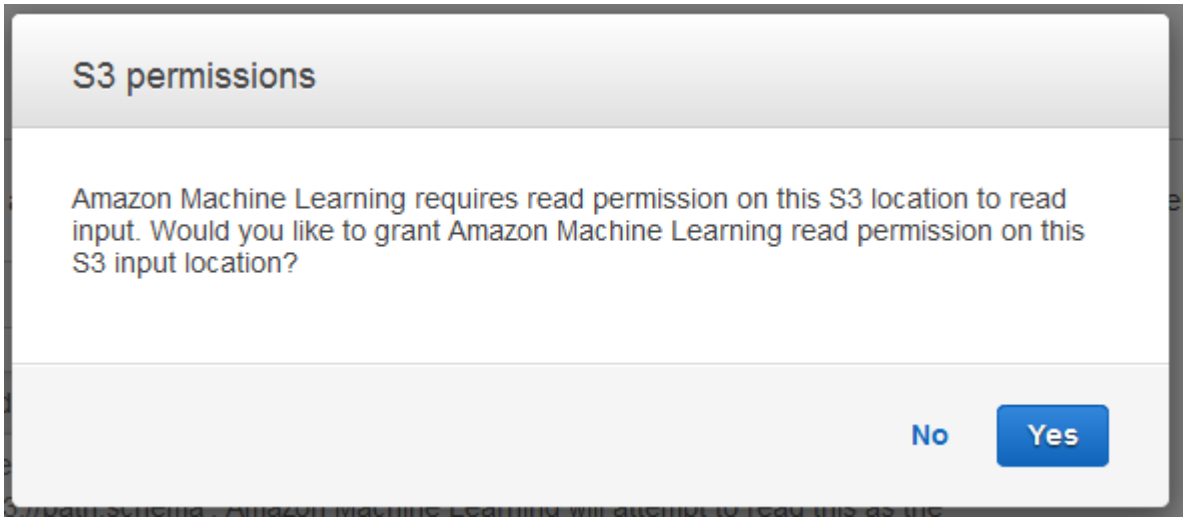
4. On the **Input Data** page, for **Where is your data located**?, make sure that **S3** is selected.



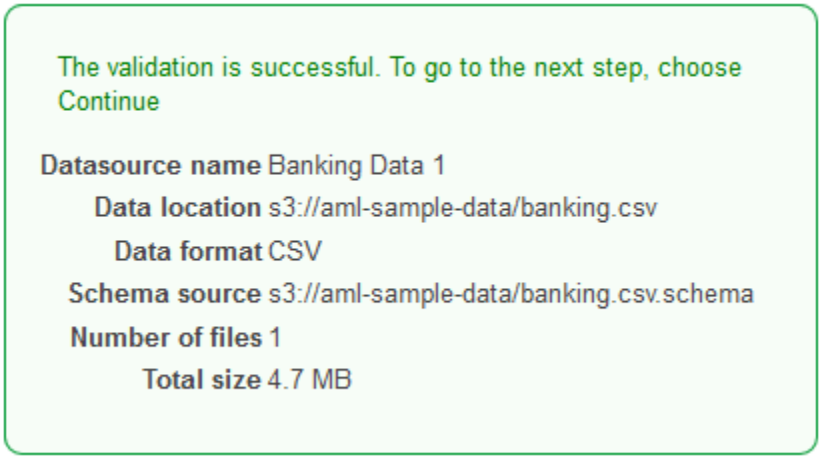
5. For **S3 Location**, type the full location of the `banking.csv` file from Step 1: Prepare Your Data. For example: *`your-bucket`***`/banking.csv`**. Amazon ML prepends s3:// to your bucket name for you.
6. For **Datasource name**, type **`Banking Data 1`**.



7. Choose **Verify**.
8. In the **S3 permissions** dialog box, choose **Yes**.



9. If Amazon ML can access and read the data file at the S3 location, you will see a page similar to the following. Review the properties, and then choose **Continue**.



Next, you establish a schema. A *schema* is the information Amazon ML needs to interpret the input data for an ML model, including attribute names and their assigned data types, and the names of special attributes. There are two ways to provide Amazon ML with a schema:

- Provide a separate schema file when you upload your Amazon S3 data.
- Allow Amazon ML to infer the attribute types and create a schema for you.

In this tutorial, we'll ask Amazon ML to infer the schema.

For information about creating a separate schema file, see Creating a Data Schema for Amazon ML.

To allow Amazon ML to infer the schema

1. On the **Schema** page, Amazon ML shows you the schema that it inferred. Review the data types that Amazon ML inferred for the attributes. It is important that attributes are assigned the correct data type to help Amazon ML ingest the data correctly and to enable the correct feature processing on the attributes.

   - Attributes that have only two possible states, such as yes or no, should be marked as **Binary**.
   - Attributes that are numbers or strings that are used to denote a category should be marked as **Categorical**.
   - Attributes that are numeric quantities for which the order is meaningful should be marked as **Numeric**.
   - Attributes that are strings that you would like to treat as words delimited by spaces should be marked as **Text**.

Name	Data Type	Sample Field Value 1
age	Numeric	56
campaign	Numeric	1
cons_conf_idx	Numeric	-36.4
cons_price_idx	Numeric	93.994
contact	Categorical	telephone
day_of_week	Categorical	mon
default	Categorical	no
duration	Numeric	261
education	Categorical	basic.4y
emp_var_rate	Numeric	1.1

2. In this tutorial, Amazon ML has correctly identified the data types for all of the attributes, so choose **Continue**.

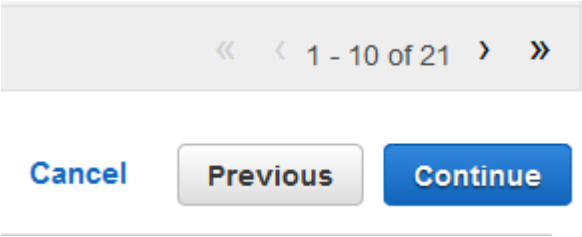
Next, select a target attribute.

Remember that the target is the attribute that the ML model must learn to predict. Attribute **y** indicates whether an individual has subscribed to a campaign in the past: 1 (yes) or 0 (no).

> **Note**
>
> Choose a target attribute only if you will use the datasource for training and evaluating ML models.

To select y as the target attribute

1. In the lower right of the table, choose the single arrow to advance to the last page of the table, where the attribute named y appears.



2. In the **Target** column, select y.

   Amazon ML confirms that **y** is selected as your target.
3. Choose **Continue**.
4. On the **Row ID** page, for **Does your data contain an identifier?** , make sure that **No**, the default, is selected.
5. Choose **Review**, and then choose **Continue**.

Now that you have a training datasource, you're ready to create your model.

Step 3: Create an ML Model

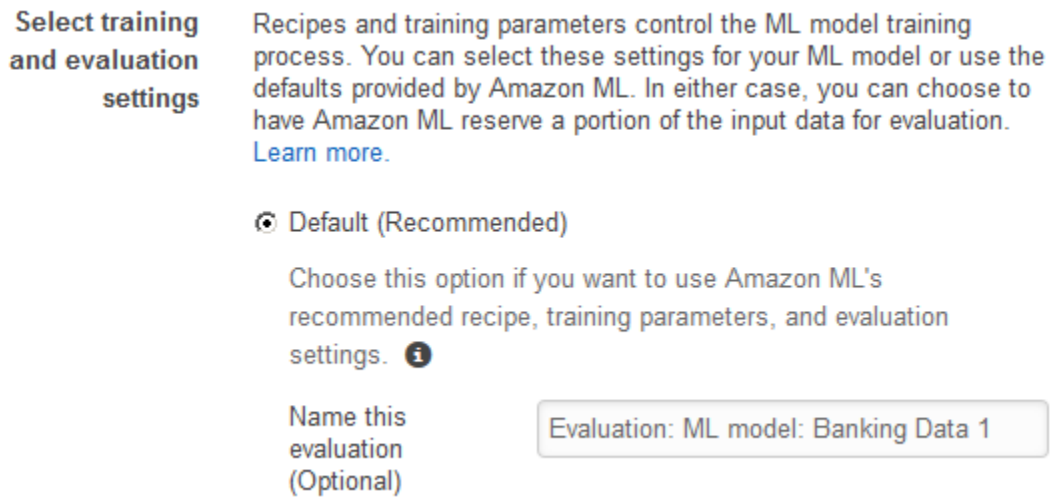
After you've created the training datasource, you use it to create an ML model, train the model, and then evaluate the results. The ML model is a collection of patterns that Amazon ML finds in your data during training. You use the model to create predictions.

To create an ML model

1. Because the Get started wizard creates both a training datasource and a model, Amazon Machine Learning (Amazon ML) automatically uses the training datasource that you just created, and takes you directly to the **ML model settings** page. On the **ML model settings** page, for **ML model name**, make sure that the default, **`ML model: Banking Data 1`**, is displayed.

Using a friendly name, such as the default, helps you easily identify and manage the ML model.

2. For **Training and evaluation settings**, ensure that **Default** is selected.



3. For **Name this evaluation**, accept the default, **`Evaluation: ML model: Banking Data 1`**.
4. Choose **Review**, review your settings, and then choose **Finish**.

   After you choose **Finish**, Amazon ML adds your model to the processing queue. When Amazon ML creates your model, it applies the defaults and performs the following actions:

   - Splits the training datasource into two sections, one containing 70% of the data and one containing the remaining 30%
   - Trains the ML model on the section that contains 70% of the input data
   - Evaluates the model using the remaining 30% of the input data

   While your model is in the queue, Amazon ML reports the status as **Pending**. While Amazon ML creates your model, it reports the status as **In Progress**. When it has completed all actions, it reports the status as **Completed**. Wait for the evaluation to complete before proceeding.

Now you are ready to review your model's performance and set a cut-off score.

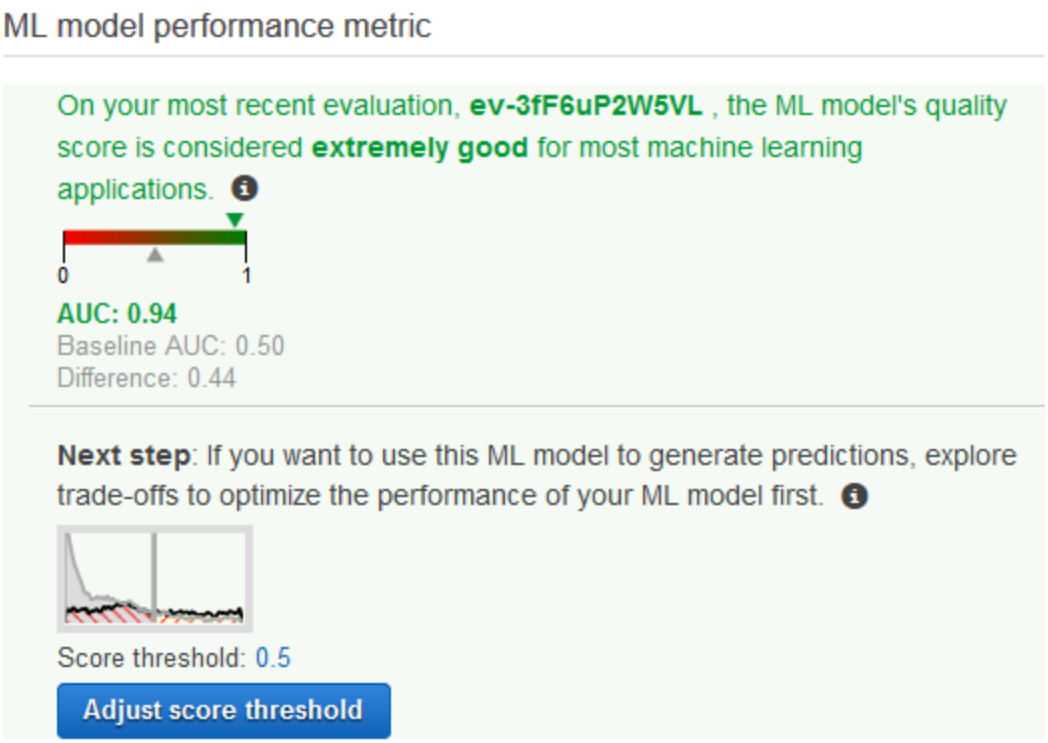
For more information about training and evaluating models, see Training ML Models and evaluate an ML model.

Step 4: Review the ML Model's Predictive Performance and Set a Score Threshold

Now that you've created your ML model and Amazon Machine Learning (Amazon ML) has evaluated it, let's see if it is good enough to put to use. During evaluation, Amazon ML computed an industry-standard quality metric, called the Area Under a Curve (AUC) metric, that expresses the performance quality of your ML model. Amazon ML also interprets the AUC metric to tell you if the quality of the ML model is adequate for most machine learning applications. (Learn more about AUC in Measuring ML Model Accuracy.) Let's review the AUC metric, and then adjust the score threshold or cut-off to optimize your model's predictive performance.

To review the AUC metric for your ML model

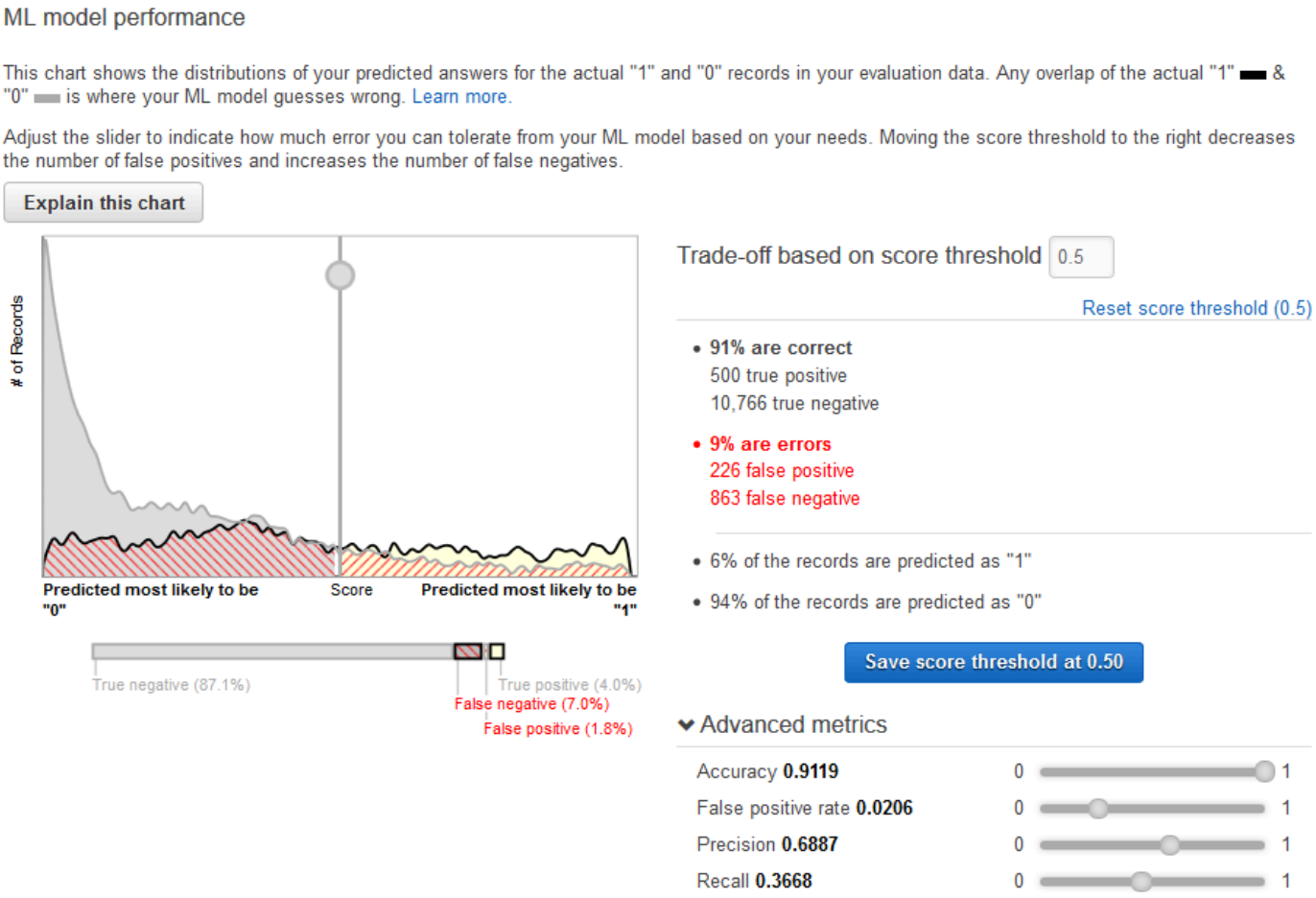
1. On the **ML model summary** page, in the **ML model report** navigation pane, choose **Evaluations**, choose **Evaluation: ML model: Banking model 1**, and then choose **Summary**.
2. On the **Evaluation summary** page, review the evaluation summary, including the model's AUC performance metric.



The ML model generates numeric prediction scores for each record in a prediction datasource, and then applies a threshold to convert these scores into binary labels of 0 (for no) or 1 (for yes). By changing the *score threshold*, you can adjust how the ML model assigns these labels. Now, set the score threshold.

To set a score threshold for your ML model

1. On the **Evaluation Summary** page, choose **Adjust Score Threshold.**

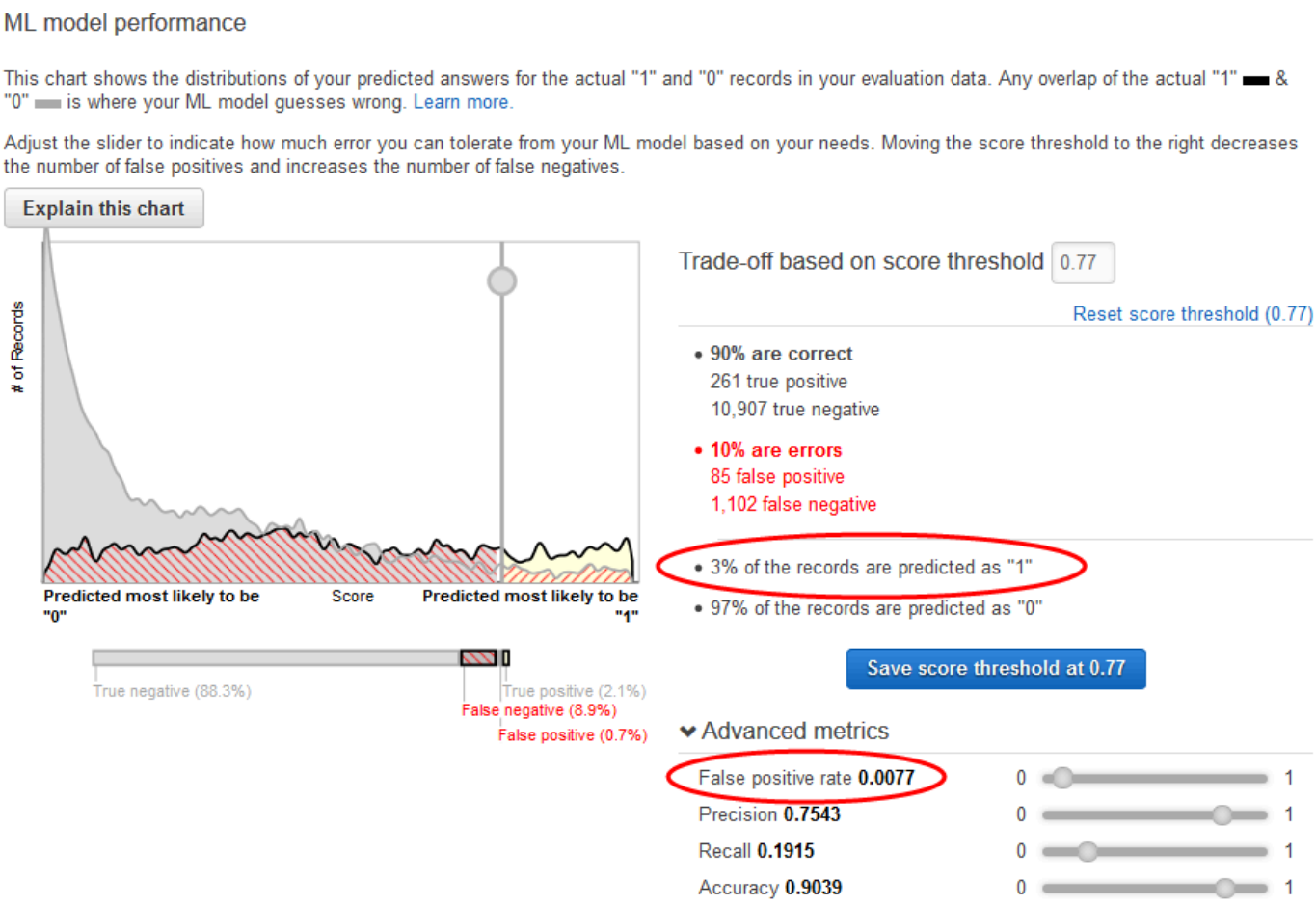


You can fine-tune your ML model performance metrics by adjusting the score threshold. Adjusting this value changes the level of confidence that the model must have in a prediction before it considers the prediction to be positive. It also changes how many false negatives and false positives you are willing to tolerate in your predictions.

You can control the cutoff for what the model considers a positive prediction by increasing the score threshold until it considers only the predictions with the highest likelihood of being true positives as positive. You can also reduce the score threshold until you no longer have any

false negatives. Choose your cutoff to reflect your business needs. For this tutorial, each false positive costs campaign money, so we want a high ratio of true positives to false positives.

2. Let's say you want to target the top 3% of the customers that will subscribe to the product. Slide the vertical selector to set the score threshold to a value that corresponds to **3% of the records are predicted as "1"**.



Note the impact of this score threshold on the ML model's performance: the false positive rate is 0.007. Let's assume that that false positive rate is acceptable.

3. Choose **Save score threshold at 0.77**.

Every time you use this ML model to make predictions, it will predict records with scores over 0.77 as "1", and the rest of the records as "0".

To learn more about the score threshold, see Binary Classification.

Now you are ready to create predictions using your model.

Step 5: Use the ML Model to Generate Predictions

Amazon Machine Learning (Amazon ML) can generate two types of predictions—batch and real-time.

A *real-time prediction* is a prediction for a single observation that Amazon ML generates on demand. Real-time predictions are ideal for mobile apps, websites, and other applications that need to use results interactively.

A *batch prediction* is a set of predictions for a group of observations. Amazon ML processes the records in a batch prediction together, so processing can take some time. Use batch predictions for applications that require predictions for set of observations or predictions that don't use results interactively.

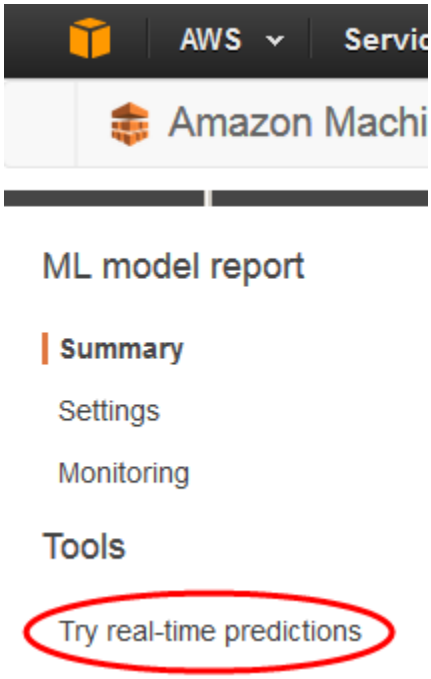
For this tutorial, you will generate a real-time prediction that predicts whether one potential customer will subscribe to the new product. You will also generate predictions for a large batch of potential customers. For the batch prediction, you will use the `banking-batch.csv` file that you uploaded in Step 1: Prepare Your Data.

Let's start with a real-time prediction.

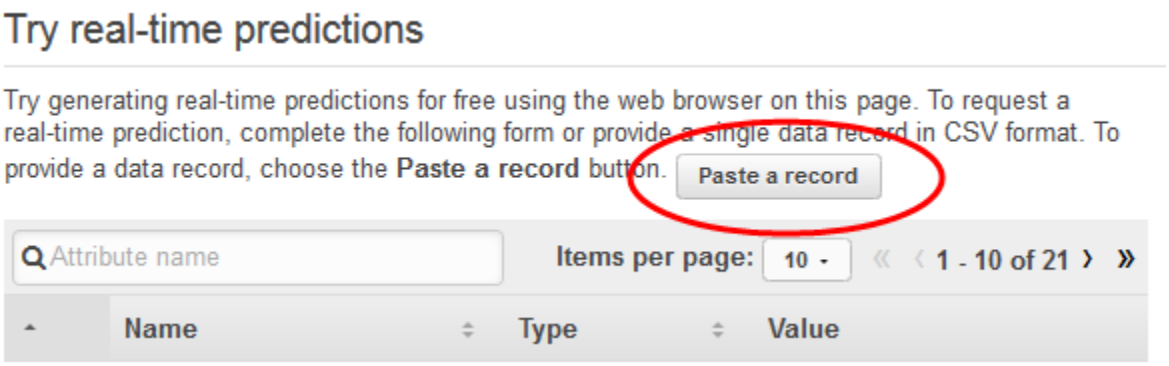
> **Note**
>
> For applications that require real-time predictions, you must create a real-time endpoint for the ML model. You accrue charges while a real-time endpoint is available. Before you commit to using real-time predictions and begin incurring the cost associated with them, you can try using the real-time prediction feature in your web browser, without creating a real-time endpoint. That's what we'll do for this tutorial.

To try a real-time prediction

1. In the **ML model report** navigation pane, choose **Try real-time predictions**.



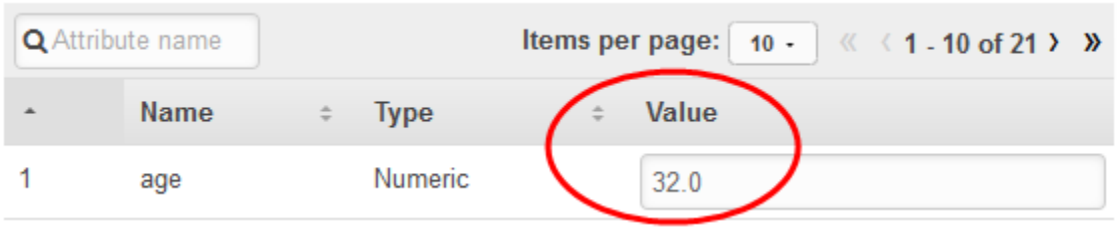
2. Choose **Paste a record**.



3. In the **Paste a record** dialog box, paste the following observation:

```
32,services,divorced,basic.9y,no,unknown,yes,cellular,dec,mon,110,1,11,0,nonexistent,-1.8,9
```

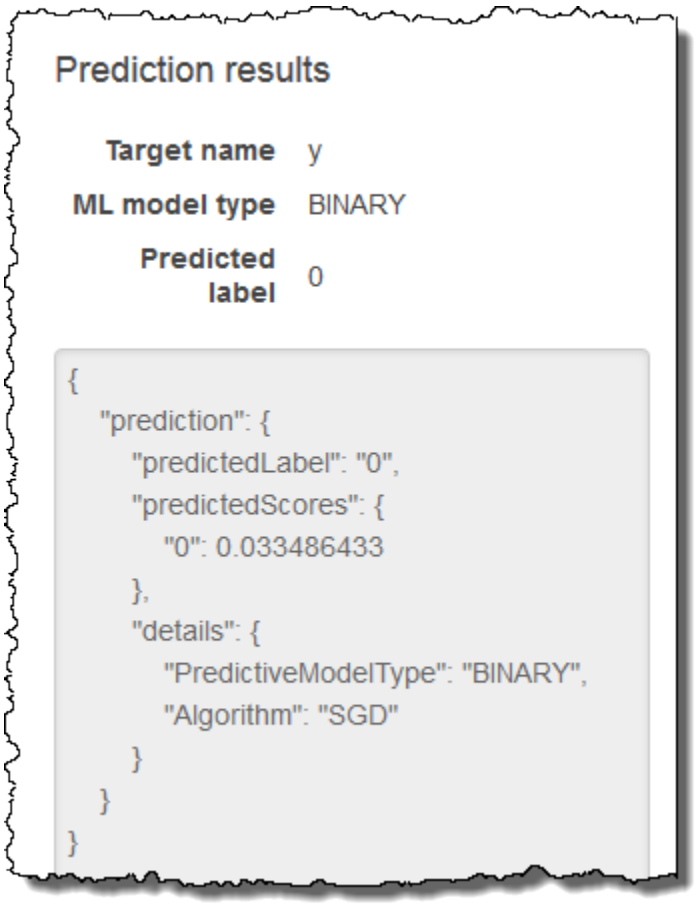
4. In the **Paste a record** dialog box, choose **Submit** to confirm that you want to generate a prediction for this observation. Amazon ML populates the values in the real-time prediction form.



> **Note**
>
> You can also populate the **Value** fields by typing in individual values. Regardless of the method you choose, you should provide an observation that wasn't used to train the model.

5. At the bottom of the page, choose **Create prediction**.

   The prediction appears in the **Prediction results** pane on the right. This prediction has a **Predicted label** of `0`, which means that this potential customer is unlikely to respond to the campaign. A **Predicted label** of 1 would mean that the customer is likely to respond to the campaign.

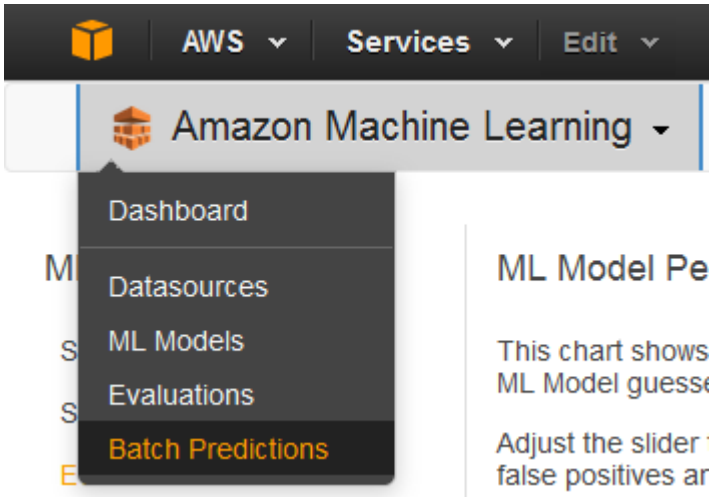


Now, create a batch prediction. You will provide Amazon ML with the name of the ML model you are using; the Amazon Simple Storage Service (Amazon S3) location of the input data for which

you want to generate predictions (Amazon ML will create a batch prediction datasource from this data); and the Amazon S3 location for storing the results.

To create a batch prediction

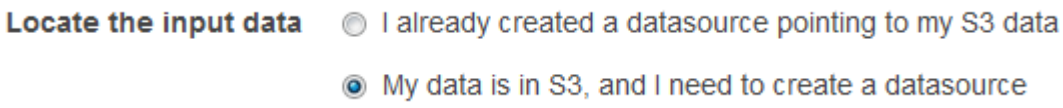
1. Choose **Amazon Machine Learning**, and then choose **Batch Predictions**.



2. Choose **Create new batch prediction**.
3. On the **ML model for batch predictions** page, choose **ML model: Banking Data 1**.

   Amazon ML displays the ML model name, ID, creation time, and the associated datasource ID.
4. Choose **Continue**.
5. To generate predictions, you need to provide Amazon ML the data that you need predictions for. This is called the *input data*. First, put the input data into a datasource so that Amazon ML can access it.

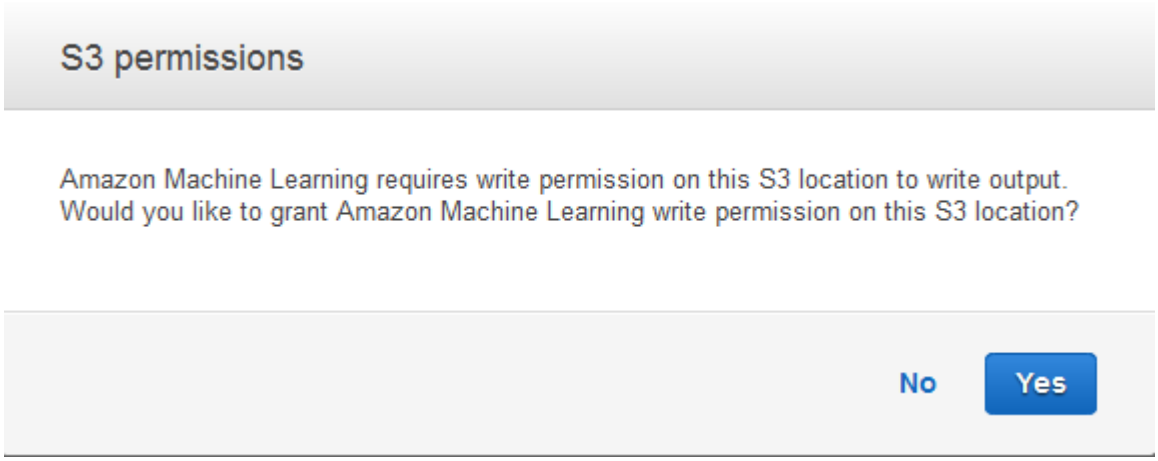
   For **Locate the input data**, choose **My data is in S3, and I need to create a datasource**.



6. For **Datasource name**, type **`Banking Data 2`**.
7. For **S3 Location**, type the full location of the `banking-batch.csv` file: ***`your-bucket`*`/`****`banking-batch.csv`***.
8. For **Does the first line in your CSV contain the column names?**, choose **Yes**.
9. Choose **Verify**.

Amazon ML validates the location of your data.

10. Choose **Continue**.
11. For **S3 destination**, type the name of the Amazon S3 location where you uploaded the files in Step 1: Prepare Your Data. Amazon ML uploads the prediction results there.
12. For **Batch prediction name**, accept the default, **`Batch prediction: ML model: Banking Data 1`**. Amazon ML chooses the default name based on the model it will use to create predictions. In this tutorial, the model and the predictions are named after the training datasource, `Banking Data 1`.
13. Choose **Review**.
14. In the **S3 permissions** dialog box, choose **Yes**.

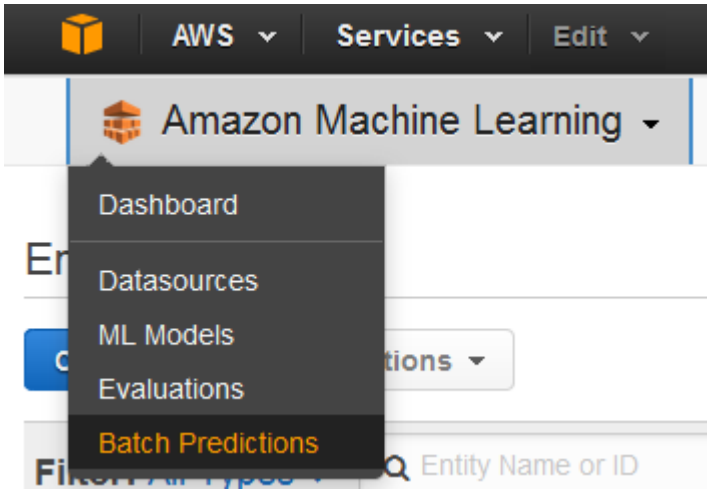


15. On the **Review** page, choose **Finish**.

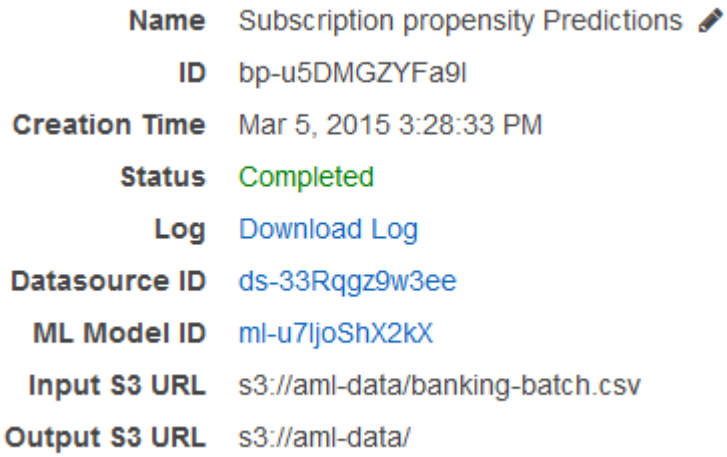
    The batch prediction request is sent to Amazon ML and entered into a queue. The time it takes Amazon ML to process a batch prediction depends on the size of your datasource and the complexity of your ML model. While Amazon ML processes the request, it reports a status of **In Progress**. After the batch prediction has completed, the request's status changes to **Completed**. Now, you can view the results.

To view the predictions

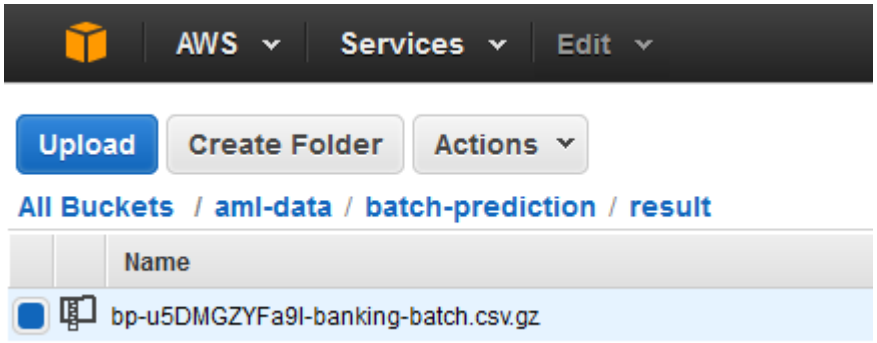
1. Choose **Amazon Machine Learning**, and then choose **Batch Predictions**.



2. In the list of predictions, choose **Batch prediction: ML model: Banking Data 1**. The **Batch prediction info** page appears.



3. To view the results of the batch prediction, go to the Amazon S3 console at https://console.aws.amazon.com/s3/ and navigate to the Amazon S3 location referenced in the **Output S3 URL** field. From there, navigate to the results folder, which will have a name similar to `s3://aml-data/batch-prediction/result`.



The prediction is stored in a compressed .gzip file with the .gz extension.

4. Download the prediction file to your desktop, uncompress it, and open it.

bestAnswer	score
0	0.06046
0	0.00507
0	0.01410
0	0.00170
0	0.00184
0	0.07133
0	0.30811

The file has two columns, **bestAnswer** and **score**, and a row for each observation in your datasource. The results in the **bestAnswer** column are based on the score threshold of 0.77 that you set in Step 4: Review the ML Model's Predictive Performance and Set a Score Threshold. A **score** greater than 0.77 results in a **bestAnswer** of 1, which is a positive response or prediction, and a **score** less than 0.77 results in a **bestAnswer** of 0, which is a negative response or prediction.

The following examples show positive and negative predictions based on the score threshold of 0.77.

Positive prediction:

bestAnswer	score
1	0.8228876

In this example, the value for **bestAnswer** is 1, and the value of **score** is 0.8228876. The value for **bestAnswer** is 1 because the **score** is greater than the score threshold of 0.77. A **bestAnswer** of 1 indicates that the customer is likely to purchase your product, and is, therefore, considered a positive prediction.

Negative prediction:

bestAnswer	score
0	0.7695356

In this example, the value of **bestAnswer** is 0 because the **score** value is 0.7695356, which is less than the score threshold of 0.77. The **bestAnswer** of 0 indicates that the customer is unlikely to purchase your product, and is, therefore, considered a negative prediction.

Each row of the batch result corresponds to a row in your batch input (an observation in your datasource).

After analyzing the predictions, you can execute your targeted marketing campaign; for example, by sending fliers to everyone with a predicted score of 1.

Now that you have created, reviewed, and used your model, clean up the data and AWS resources you created to avoid incurring unnecessary charges and to keep your workspace uncluttered.

Step 6: Clean Up

To avoid accruing additional Amazon Simple Storage Service (Amazon S3) charges, delete the data stored in Amazon S3. You are not charged for other unused Amazon ML resources, but we recommend that you delete them to keep your workspace clean.

To delete the input data stored in Amazon S3

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Navigate to the Amazon S3 location where you stored the `banking.csv` and `banking-batch.csv` files.
3. Select the `banking.csv`, `banking-batch.csv`, and `.writePermissionCheck.tmp` files.
4. Choose **Actions**, and then choose **Delete**.
5. When prompted for confirmation, choose **OK**.

Although you aren't charged for keeping the record of the batch prediction that Amazon ML ran or the datasources, model, and evaluation that you created during the tutorial, we recommend that you delete them to prevent cluttering your workspace.

To delete the batch predictions

1. Navigate to the Amazon S3 location where you stored the output of the batch prediction.
2. Choose the `batch-prediction` folder.
3. Choose **Actions**, and then choose **Delete**.
4. When prompted for confirmation, choose **OK**.

To delete the Amazon ML resources

1. On the Amazon ML dashboard, select the following resources.
   - The `Banking Data 1` datasource
   - The `Banking Data 1_[percentBegin=0, percentEnd=70, strategy=sequential]` datasource
   - The `Banking Data 1_[percentBegin=70, percentEnd=100, strategy=sequential]` datasource
   - The `Banking Data 2` datasource
   - The `ML model: Banking Data 1` ML model
   - The `Evaluation: ML model: Banking Data 1` evaluation
2. Choose **Actions**, and then choose **Delete**.
3. In the dialog box, choose **Delete** to delete all selected resources.

You have now successfully completed the tutorial. To continue using the console to create datasources, models, and predictions see the Amazon Machine Learning Developer Guide. To learn how to use the API, see the Amazon Machine Learning API Reference.

Creating and Using Datasources

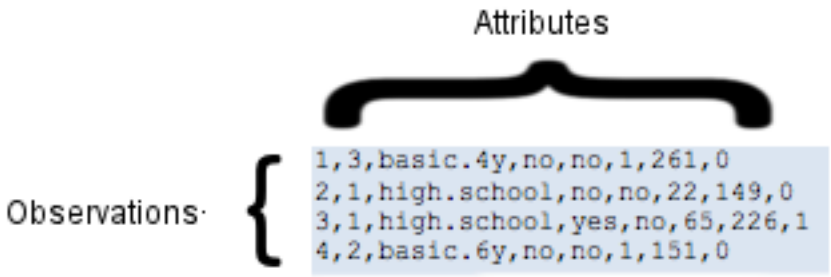
You can use Amazon ML datasources to train an ML model, evaluate an ML model, and generate batch predictions using an ML model. Datasource objects contain metadata about your input data. When you create a datasource, Amazon ML reads your input data, computes descriptive statistics on its attributes, and stores the statistics, a schema, and other information as part of the datasource object. After you create a datasource, you can use the Amazon ML data insights to explore statistical properties of your input data, and you can use the datasource to train an ML model.

> **Note**
>
> This section assumes that you are familiar with Amazon Machine Learning concepts.

Topics

Understanding the Data Format for Amazon ML

Input data is the data that you use to create a datasource. You must save your input data in the comma-separated values (.csv) format. Each row in the .csv file is a single data record or observation. Each column in the .csv file contains an attribute of the observation. For example, the following figure shows the contents of a .csv file that has four observations, each in its own row. Each observation contains eight attributes, separated by a comma. The attributes represent the following information about each individual represented by an observation: customerId,jobId,education,housing,loan,campaign,duration,willRespondToCampaign.



Attributes

Amazon ML requires names for each attribute. You can specify attribute names by:

- Including the attribute names in the first line (also known as a header line) of the .csv file that you use as your input data
- Including the attribute names in a separate schema file that is located in the same S3 bucket as your input data

For more information about using schema files, see Creating a Data Schema.

The following example of a .csv file includes the names of the attributes in the header line.

```
customerId,jobId,education,housing,loan,campaign,duration,willRespondToCampaign

1,3,basic.4y,no,no,1,261,0

2,1,high.school,no,no,22,149,0

3,1,high.school,yes,no,65,226,1

4,2,basic.6y,no,no,1,151,0
```

Input File Format Requirements

The .csv file that contains your input data must meet the following requirements:

- Must be in plain text using a character set such as ASCII, Unicode, or EBCDIC.
- Consist of observations, one observation per line.

- For each observation, the attribute values must be separated by commas.
- If an attribute value contains a comma (the delimiter), the entire attribute value must be enclosed in double quotes.
- Each observation must be terminated with an end-of-line character, which is a special character or sequence of characters indicating the end of a line.
- Attribute values cannot include end-of-line characters, even if the attribute value is enclosed in double quotes.
- Every observation must have the same number of attributes and sequence of attributes.
- Each observation must be no larger than 100 KB. Amazon ML rejects any observation larger than 100 KB during processing. If Amazon ML rejects more than 10,000 observations, it rejects the entire .csv file.

Using Multiple Files As Data Input to Amazon ML

You can provide your input to Amazon ML as a single file, or as a collection of files. Collections must satisfy these conditions:

- All files must have the same data schema.
- All files must reside in the same Amazon Simple Storage Service (Amazon S3) prefix, and the path that you provide for the collection must end with a forward slash ('/') character.

For example, if your data files are named input1.csv, input2.csv, and input3.csv, and your S3 bucket name is s3://examplebucket, your file paths might look like this:

s3://examplebucket/path/to/data/input1.csv

s3://examplebucket/path/to/data/input2.csv

s3://examplebucket/path/to/data/input3.csv

You would provide the following S3 location as input to Amazon ML:

's3://examplebucket/path/to/data/'

End-of-Line Characters in CSV Format

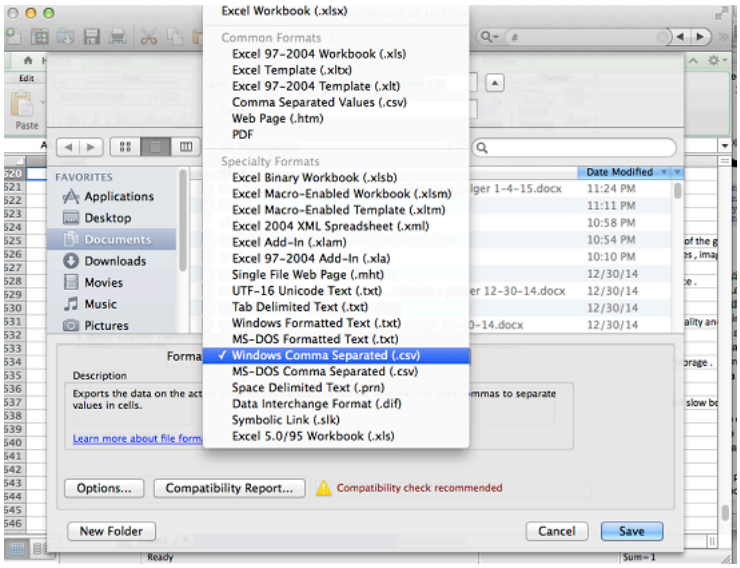
When you create your .csv file, each observation will be terminated by a special end-of-line character. This character is not visible, but is automatically included at the end of each observation

when you press your **Enter** or **Return** key. The special character that represents the end-of-line varies depending on your operating system. Unix systems, such as Linux or OS X, use a *line feed* character that is indicated by "\n" (ASCII code 10 in decimal or 0x0a in hexadecimal). Microsoft Windows uses two characters called *carriage return* and *line feed* that are indicated by "\r\n" (ASCII codes 13 and 10 in decimal or 0x0d and 0x0a in hexadecimal).

If you want to use OS X and Microsoft Excel to create your .csv file, perform the following procedure. Be sure to choose the correct format.

To save a .csv file if you use OS X and Excel

1. When saving the .csv file, choose **Format,** and then choose **Windows Comma Separated (.csv)**.
2. Choose **Save**.



> **Important**
>
> Do not save the .csv file by using the **Comma Separated Values (.csv)** or **MS-DOS Comma Separated (.csv)** formats because Amazon ML is unable to read them.

Creating a Data Schema for Amazon ML

A *schema* is composed of all attributes in the input data and their corresponding data types. It allows Amazon ML to understand the data in the datasource. Amazon ML uses the information in the schema to read and interpret the input data, compute statistics, apply the correct attribute

transformations, and fine-tune its learning algorithms. If you don't provide a schema, Amazon ML infers one from the data.

Example Schema

For Amazon ML to read the input data correctly and produce accurate predictions, each attribute must be assigned the correct data type. Let's walk through an example to see how data types are assigned to attributes, and how the attributes and data types are included in a schema. We'll call our example "Customer Campaign" because we want to predict which customers will respond to our email campaign. Our input file is a .csv file with nine columns:

```
1,3,web developer,basic.4y,no,no,1,261,0

2,1,car repair,high.school,no,no,22,149,0

3,1,car mechanic,high.school,yes,no,65,226,1

4,2,software developer,basic.6y,no,no,1,151,0
```

This the schema for this data:

```
{
    "version": "1.0",
    "rowId": "customerId",
    "targetAttributeName": "willRespondToCampaign",
    "dataFormat": "CSV",
    "dataFileContainsHeader": false,
    "attributes": [
        {
            "attributeName": "customerId",
            "attributeType": "CATEGORICAL"
        },
        {
            "attributeName": "jobId",
            "attributeType": "CATEGORICAL"
        },
        {
            "attributeName": "jobDescription",
            "attributeType": "TEXT"
        },
        {
            "attributeName": "education",
```

```
            "attributeType": "CATEGORICAL"
        },
        {
            "attributeName": "housing",
            "attributeType": "CATEGORICAL"

        },
        {
            "attributeName": "loan",
            "attributeType": "CATEGORICAL"

        },
        {
            "attributeName": "campaign",
            "attributeType": "NUMERIC"
        },
        {
            "attributeName": "duration",
            "attributeType": "NUMERIC"
        },

        {
            "attributeName": "willRespondToCampaign",
            "attributeType": "BINARY"
        }

    ]
}
```

In the schema file for this example, the value for the `rowId` is `customerId`:

```
"rowId": "customerId",
```

The attribute `willRespondToCampaign` is defined as the target attribute:

```
"targetAttributeName": "willRespondToCampaign ",
```

The `customerId` attribute and the `CATEGORICAL` data type are associated with the first column, the `jobId` attribute and the `CATEGORICAL` data type are associated with the second column, the `jobDescription` attribute and the `TEXT` data type are associated with the third column, the `education` attribute and the `CATEGORICAL` data type are associated with the fourth column, and

so on. The ninth column is associated with the `willRespondToCampaign` attribute with a `BINARY` data type, and this attribute also is defined as the target attribute.

Using the targetAttributeName Field

The `targetAttributeName` value is the name of the attribute that you want to predict. You must assign a `targetAttributeName` when creating or evaluating a model.

When you are training or evaluating an ML model, the `targetAttributeName` identifies the name of the attribute in the input data that contains the "correct" answers for the target attribute. Amazon ML uses the target, which includes the correct answers, to discover patterns and generate a ML model.

When you are evaluating your model, Amazon ML uses the target to check the accuracy of your predictions. After you have created and evaluated the ML model, you can use data with an unassigned `targetAttributeName` to generate predictions with your ML model.

You define the target attribute in the Amazon ML console when you create a datasource, or in a schema file. If you create your own schema file, use the following syntax to define the target attribute:

```
"targetAttributeName": "exampleAttributeTarget",
```

In this example, `exampleAttributeTarget` is the name of the attribute in your input file that is the target attribute.

Using the rowID Field

The `row ID` is an optional flag associated with an attribute in the input data. If specified, the attribute marked as the `row ID` is included in the prediction output. This attribute makes it easier to associate which prediction corresponds with which observation. An example of a good `row ID` is a customer ID or a similar unique attribute.

> **Note**
>
> The row ID is for your reference only. Amazon ML doesn't use it when training an ML model. Selecting an attribute as a row ID excludes it from being used for training an ML model.

You define the `row ID` in the Amazon ML console when you create a datasource, or in a schema file. If you are creating your own schema file, use the following syntax to define the `row ID`:

```
"rowId": "exampleRow",
```

In the preceding example, `exampleRow` is the name of the attribute in your input file that is defined as the row ID.

When generating batch predictions, you might get the following output:

```
tag,bestAnswer,score
55,0,0.46317
102,1,0.89625
```

In this example, `RowID` represents the attribute `customerId`. For example, customerId `55` is predicted to respond to our email campaign with low confidence (0.46317) , while `customerId` `102` is predicted to respond to our email campaign with high confidence (0.89625).

Using the AttributeType Field

In Amazon ML, there are four data types for attributes:

Binary

Choose `BINARY` for an attribute that has only two possible states, such as `yes` or `no`.

For example, the attribute `isNew`, for tracking whether a person is a new customer, would have a `true` value to indicate that the individual is a new customer, and a `false` value to indicate that he or she is not a new customer.

Valid negative values are `0`, `n`, `no`, `f`, and `false`.

Valid positive values are `1`, `y`, `yes`, `t`, and `true`.

Amazon ML ignores the case of binary inputs and strips the surrounding white space. For example, `" FaLSe "` is a valid binary value. You can mix the binary values that you use in the same datasource, such as using `true`, `no`, and `1`. Amazon ML outputs only `0` and `1` for binary attributes.

Categorical

Choose `CATEGORICAL` for an attribute that takes on a limited number of unique string values. For example, a user ID, the month, and a zip code are categorical values. Categorical attributes are treated as a single string, and are not tokenized further.

Numeric

Choose `NUMERIC` for an attribute that takes a quantity as a value.

For example, temperature, weight, and click rate are numeric values.

Not all attributes that hold numbers are numeric. Categorical attributes, such as days of the month and IDs, are often represented as numbers. To be considered numeric, a number must be comparable to another number. For example, the customer ID `664727` tells you nothing about the customer ID `124552`, but a weight of `10` tells you that that attribute is heavier than an attribute with a weight of `5`. Days of the month are not numeric, because the first of one month could occur before or after the second of another month.

> **Note**
>
> When you use Amazon ML to create your schema, it assigns the `Numeric` data type to all attributes that use numbers. If Amazon ML creates your schema, check for incorrect assignments and set those attributes to `CATEGORICAL`.

Text

Choose `TEXT` for an attribute that is a string of words. When reading in text attributes, Amazon ML converts them into tokens, delimited by white spaces.

For example, `email subject` becomes `email` and `subject`, and `email-subject here` becomes `email-subject` and `here`.

If the data type for a variable in the training schema does not match the data type for that variable in the evaluation schema, Amazon ML changes the evaluation data type to match the training data type. For example, if the training data schema assigns a data type of `TEXT` to the variable `age`, but the evaluation schema assigns a data type of `NUMERIC` to `age`, then Amazon ML treats the ages in the evaluation data as `TEXT` variables instead of `NUMERIC`.

For information about statistics associated with each data type, see Descriptive Statistics.

Providing a Schema to Amazon ML

Every datasource needs a schema. You can choose from two ways to provide Amazon ML with a schema:

- Allow Amazon ML to infer the data types of each attribute in the input data file and automatically create a schema for you.
- Provide a schema file when you upload your Amazon Simple Storage Service (Amazon S3) data.

Allowing Amazon ML to Create Your Schema

When you use the Amazon ML console to create a datasource, Amazon ML uses simple rules, based on the values of your variables, to create your schema. We strongly recommend that you review the Amazon ML-created schema, and correct the data types if they aren't accurate.

Providing a Schema

After you create your schema file, you need to make it available to Amazon ML. You have two options:

1. Provide the schema by using the Amazon ML console.

   Use the console to create your datasource, and include the schema file by appending the .schema extension to the file name of your input data file. For example, if the Amazon Simple Storage Service (Amazon S3) URI to your input data is s3://my-bucket-name/data/input.csv, the URI to your schema will be s3://my-bucket-name/data/input.csv.schema. Amazon ML automatically locates the schema file that you provide instead of attempting to infer the schema from your data.

   To use a directory of files as your data input to Amazon ML, append the .schema extension to your directory path. For example, if your data files reside in the location s3://examplebucket/path/to/data/, the URI to your schema will be s3://examplebucket/path/to/data/.schema.
2. Provide the schema by using the Amazon ML API.

   If you plan to call the Amazon ML API to create your datasource, you can upload the schema file into Amazon S3, and then provide the URI to that file in the `DataSchemaLocationS3` attribute of the `CreateDataSourceFromS3` API. For more information, see CreateDataSourceFromS3.

You can provide the schema directly in the payload of `CreateDataSource*` APIs instead of first saving it to Amazon S3. You do this by placing the full schema string in the `DataSchema` attribute of `CreateDataSourceFromS3`, `CreateDataSourceFromRDS`, or `CreateDataSourceFromRedshift` APIs. For more information, see the Amazon Machine Learning API Reference.

Splitting Your Data

The fundamental goal of an ML model is to make accurate predictions on future data instances beyond those used to train models. Before using an ML model to make predictions, we need to evaluate the predictive performance of the model. To estimate the quality of an ML models predictions with data it has not seen, we can reserve, or split, a portion of the data for which we already know the answer as a proxy for future data and evaluate how well the ML model predicts the correct answers for that data. You split the datasource into a portion for the training datasource and a portion for the evaluation datasource.

Amazon ML provides three options for splitting your data:

- **Pre-split the data** - You can split the data into two data input locations, before uploading them to Amazon Simple Storage Service (Amazon S3) and creating two separate datasources with them.
- **Amazon ML sequential split** - You can tell Amazon ML to split your data sequentially when creating the training and evaluation datasources.
- **Amazon ML random split** - You can tell Amazon ML to split your data using a seeded random method when creating the training and evaluation datasources.

Pre-splitting Your Data

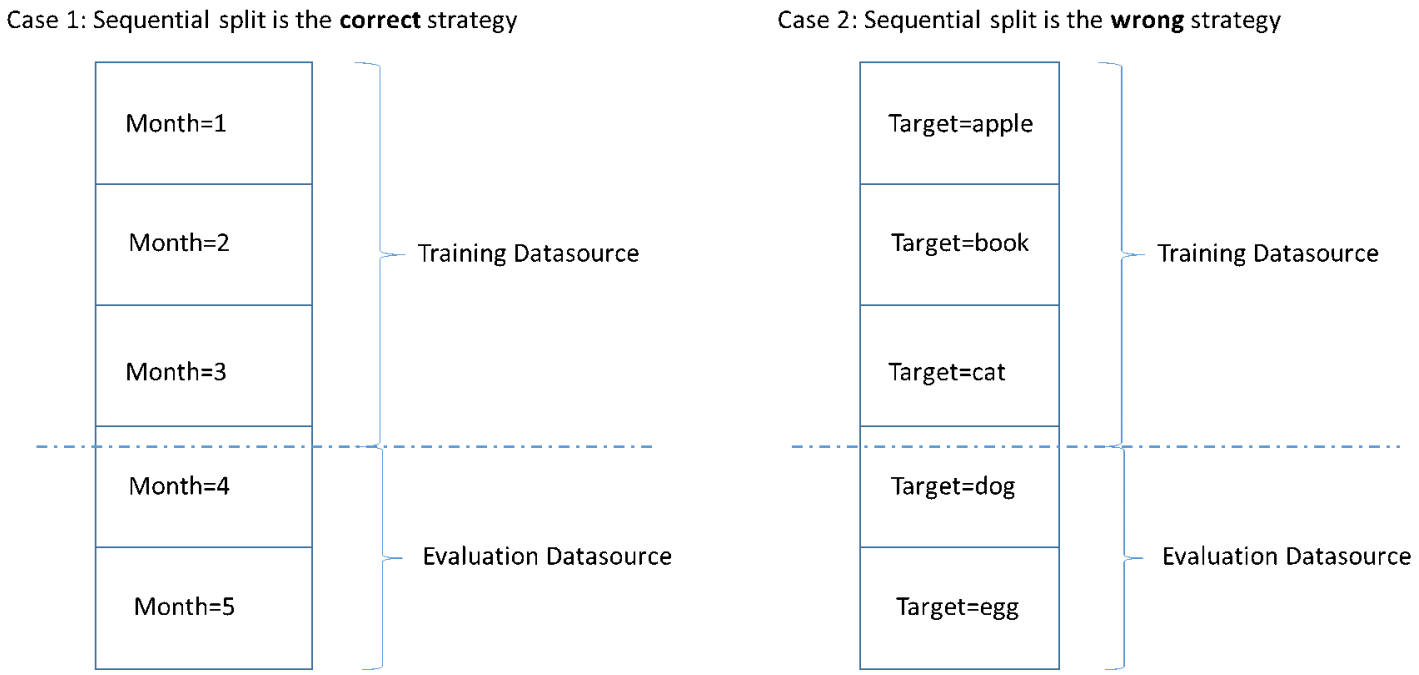
If you want explicit control over the data in your training and evaluation datasources, split your data into separate data locations, and create a separate datasources for the input and evaluation locations.

Sequentially Splitting Your Data

A simple way to split your input data for training and evaluation is to select non-overlapping subsets of your data while preserving the order of the data records. This approach is useful if you

want to evaluate your ML models on data for a certain date or within a certain time range. For example, say that you have customer engagement data for the past five months, and you want to use this historical data to predict customer engagement in the next month. Using the beginning of the range for training, and the data from the end of the range for evaluation might produce a more accurate estimate of the model's quality than using records data drawn from the entire data range.

The following figure shows examples of when you should use a sequential splitting strategy versus when you should use a random strategy.



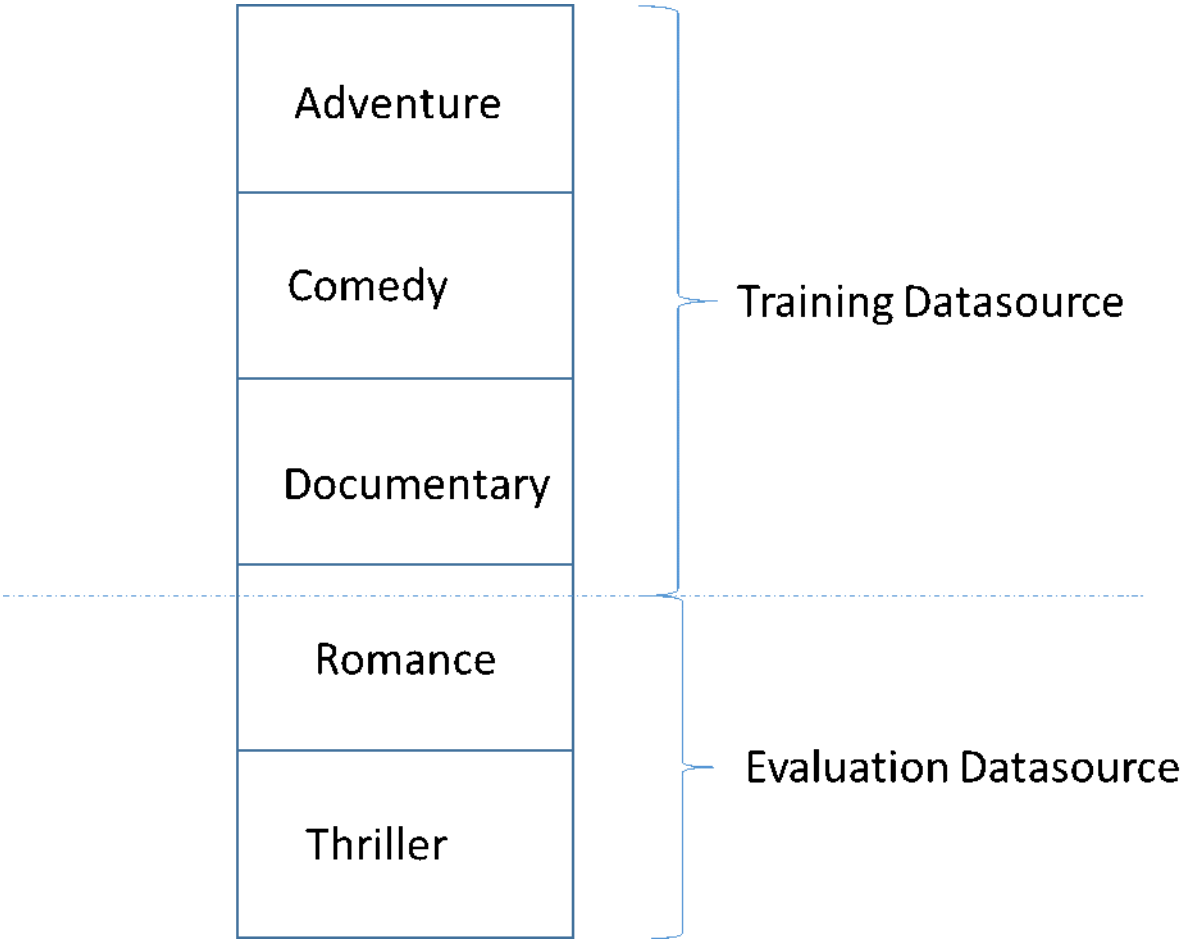
When you create a datasource, you can choose to split your datasource sequentially, and Amazon ML uses the first 70 percent of your data for training and the remaining 30 percent of the data for evaluation. This is the default approach when you use the Amazon ML console to split your data.

Randomly Splitting Your Data

Randomly splitting the input data into training and evaluation datasources ensures that the distribution of the data is similar in the training and evaluation datasources. Choose this option when you don't need to preserve the order of your input data.

Amazon ML uses a seeded pseudo-random number generation method to split your data. The seed is based partly on an input string value and partially on the content of the data itself. By default, the Amazon ML console uses the S3 location of the input data as the string. API users can provide a custom string. This means that given the same S3 bucket and data, Amazon ML splits the data the same way every time. To change how Amazon ML splits the

data, you can use the `CreateDatasourceFromS3`, `CreateDatasourceFromRedshift`, or `CreateDatasourceFromRDS` API and provide a value for the seed string. When using these APIs to create separate datasources for training and evaluation, it is important to use the same seed string value for both datasources and the complement flag for one datasource, to ensure that there is no overlap between the training and evaluation data.



A common pitfall in developing a high-quality ML model is evaluating the ML model on data that is not similar to the data used for training. For example, say you are using ML to predict the genre of movies, and your training data contains movies from the Adventure, Comedy, and Documentary genres. However, your evaluation data contains only data from the Romance and Thriller genres. In this case, the ML model did not learn any information about the Romance and Thriller genres, and the evaluation did not evaluate how well the model learned patterns for the Adventure, Comedy, and Documentary genres. As a result, the genre information is useless, and the quality of the ML model predictions for all of the genres is compromised. The model and evaluation are too dissimilar (have extremely different descriptive statistics) to be useful. This can happen when input data is sorted by one of the columns in the dataset, and then split sequentially.

If your training and evaluation datasources have different data distributions, you will see an evaluation alert in your model evaluation. For more information about evaluation alerts, see Evaluation Alerts.

You do not need to use random splitting in Amazon ML if you have already randomized your input data, for example, by randomly shuffling your input data in Amazon S3, or by using a Amazon Redshift SQL query's `random()` function or a MySQL SQL query's `rand()` function when creating the datasources. In these cases, you can rely on the sequential split option to create training and evaluation datasources with similar distributions.

Data Insights

Amazon ML computes descriptive statistics on your input data that you can use to understand your data.

Descriptive Statistics

Amazon ML computes the following descriptive statistics for different attribute types:

Numeric:

- Distribution histograms
- Number of invalid values
- Minimum, median, mean, and maximum values

Binary and categorical:

- Count (of distinct values per category)
- Value distribution histogram
- Most frequent values
- Unique values counts
- Percentage of true value (binary only)
- Most prominent words
- Most frequent words

Text:

- Name of the attribute

- Correlation to target (if a target is set)
- Total words
- Unique words
- Range of the number of words in a row
- Range of word lengths
- Most prominent words

Accessing Data Insights on the Amazon ML console

On the Amazon ML console, you can choose the name or ID of any datasource to view its **Data Insights** page. This page provides metrics and visualizations that enable you to learn about the input data associated with the datasource, including the following information:

- Data summary
- Target distributions
- Missing values
- Invalid values
- Summary statistics of variables by data type
- Distributions of variables by data type

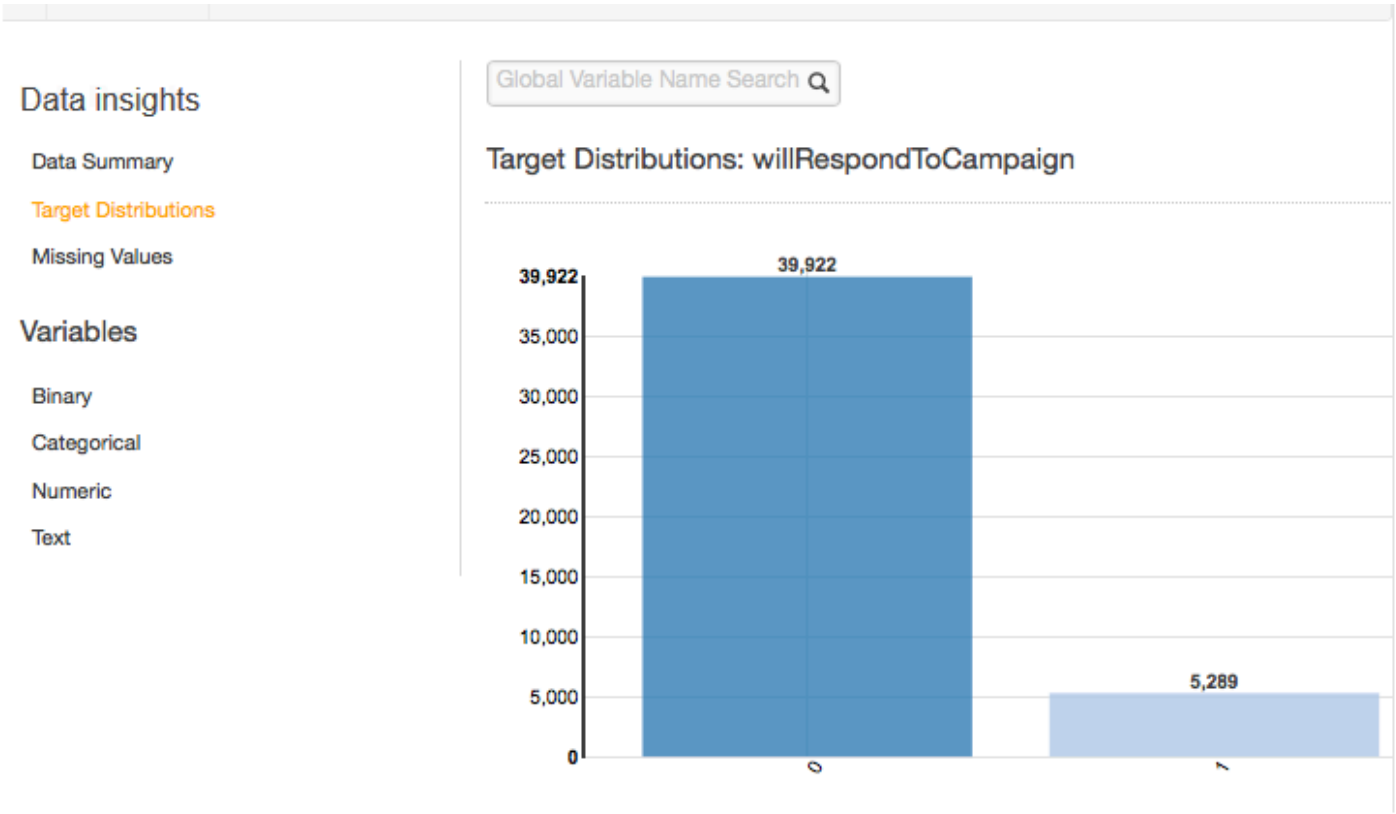
The following sections describe the metrics and visualizations in greater detail.

Data Summary

The data summary report of a datasource displays summary information, including the datasource ID, name, where it was completed, current status, target attribute, input data information (S3 bucket location, data format, number of records processed and number of bad records encountered during processing) as well as the number of variables by data type.

Target Distributions

The target distributions report shows the distribution of the target attribute of the datasource. In the following example, there are 39,922 observations where the willRespondToCampaign target attribute equals 0. This is the number of customers who did not respond to the email campaign. There are 5,289 observations where willRespondToCampaign equals 1. This is the number of customers who responded to the email campaign.



Missing Values

The missing values report lists the attributes in the input data for which values are missing. Only attributes with numeric data types can have missing values. Because missing values can affect the quality of training an ML model, we recommend that missing values be provided, if possible.

During ML model training, if the target attribute is missing, Amazon ML rejects the corresponding record. If the target attribute is present in the record, but a value for another numeric attribute is missing, then Amazon ML overlooks the missing value. In this case, Amazon ML creates a substitute attribute and sets it to 1 to indicate that this attribute is missing. This allows Amazon ML to learn patterns from the occurrence of missing values.

Invalid Values

Invalid values can occur only with Numeric and Binary data types. You can find invalid values by viewing the summary statistics of variables in the data type reports. In the following examples, there is one invalid value in the duration Numeric attribute and two invalid values in the Binary data type (one in the housing attribute and one in the loan attribute).

Numeric Variables

Variables	Correlations to Target	Missing Values	Invalid Values	Range	Mean	Median	Preview
duration	0.05165	2 (0%)	1 (0%)	0 - 4918	258.1618	180	

Binary Variables

Variables	Correlations to Target	Percent True	Invalid Values	Preview
campaign	NA	100%	27667 (61%)	
housing	0.01842	56%	1 (0%)	
loan	0.00656	16%	1 (0%)	
willRespondToCampaign	NA	12%	0 (0%)	

Variable-Target Correlation

After you create a datasource, Amazon ML can evaluate the datasource and identify the correlation, or impact, between variables and the target. For example, the price of a product might have a significant impact on whether or not it is a best seller, while the dimensions of the product might have little predictive power.

It is generally a best practice to include as many variables in your training data as possible. However, the noise introduced by including many variables with little predictive power might negatively affect the quality and accuracy of your ML model.

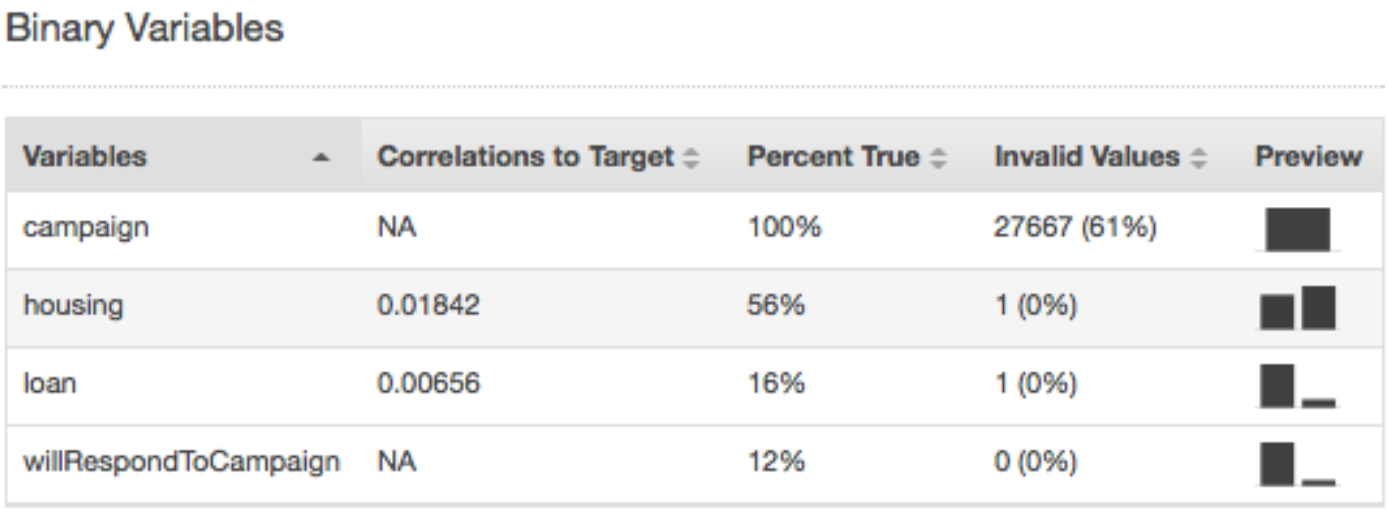
You might be able to improve the predictive performance of your model by removing variables that have little impact when you train your model. You can define which variables are made available to the machine learning process in a *recipe*, which is a transformation mechanism of Amazon ML. To learn more about recipes, see Data Transformation for Machine Learning.

Summary Statistics of Attributes by Data Type

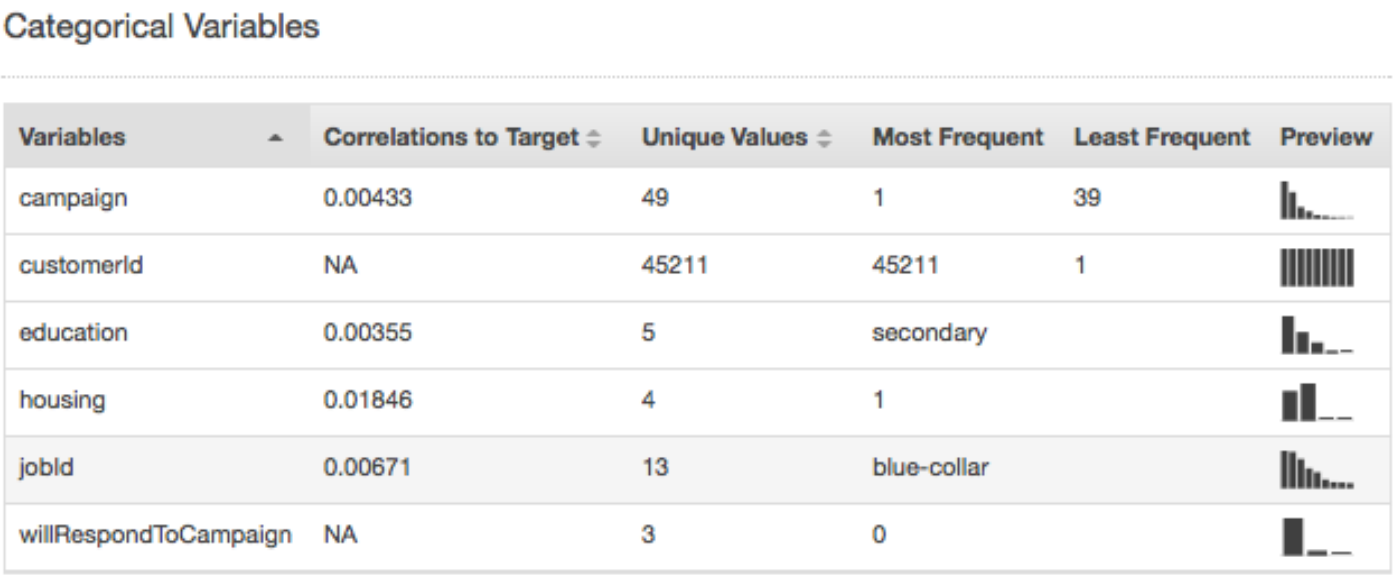
In the data insights report, you can view attribute summary statistics by the following data types:

- Binary
- Categorical
- Numeric
- Text

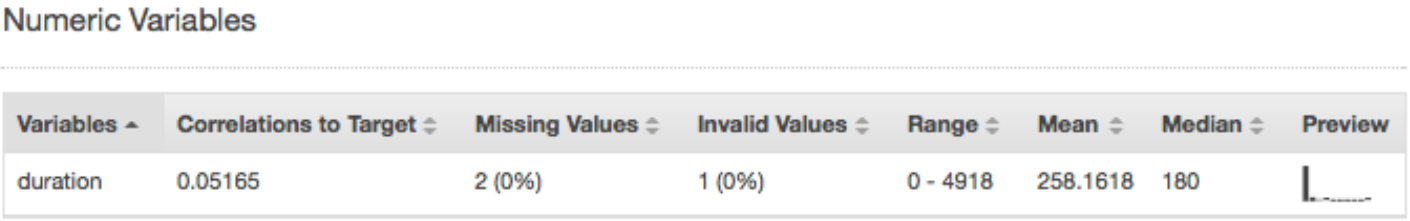
Summary statistics for the Binary data type show all binary attributes. The **Correlations to target** column shows the information shared between the target column and the attribute column. The **Percent true** column shows the percentage of observations that have value 1. The **Invalid values** column shows the number of invalid values as well as the percentage of invalid values for each attribute. The **Preview** column provides a link to a graphical distribution for each attribute.



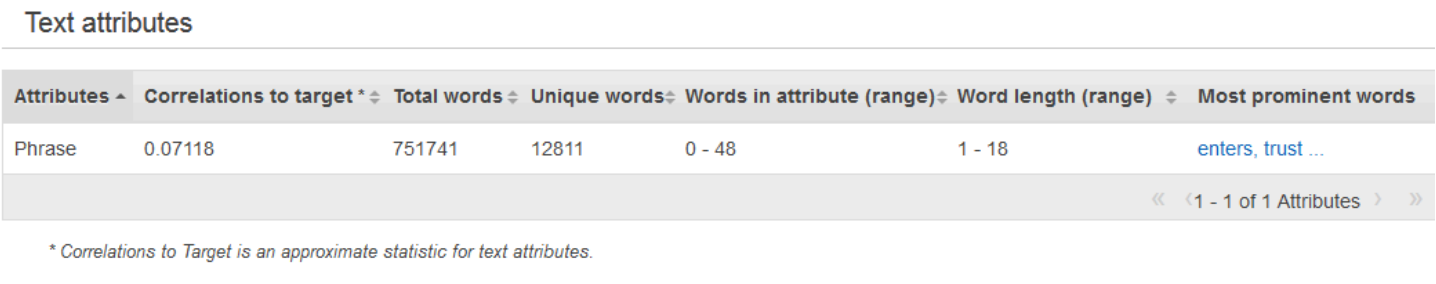
Summary statistics for the Categorical data type show all Categorical attributes with the number of unique values, most frequent value, and least frequent value. The **Preview** column provides a link to a graphical distribution for each attribute.



Summary statistics for the Numeric data type show all Numeric attributes with the number of missing values, invalid values, range of values, mean, and median. The **Preview** column provides a link to a graphical distribution for each attribute.



Summary statistics for the Text data type show all of the Text attributes, the total number of words in that attribute, the number of unique words in that attribute, the range of words in an attribute, the range of word lengths, and the most prominent words. The **Preview** column provides a link to a graphical distribution for each attribute.



The next example shows the Text data type statistics for a text variable called review, with four records.

```
1. The fox jumped over the fence.
2. This movie is intriguing.
3.
4. Fascinating movie.
```

The columns for this example would show the following information.

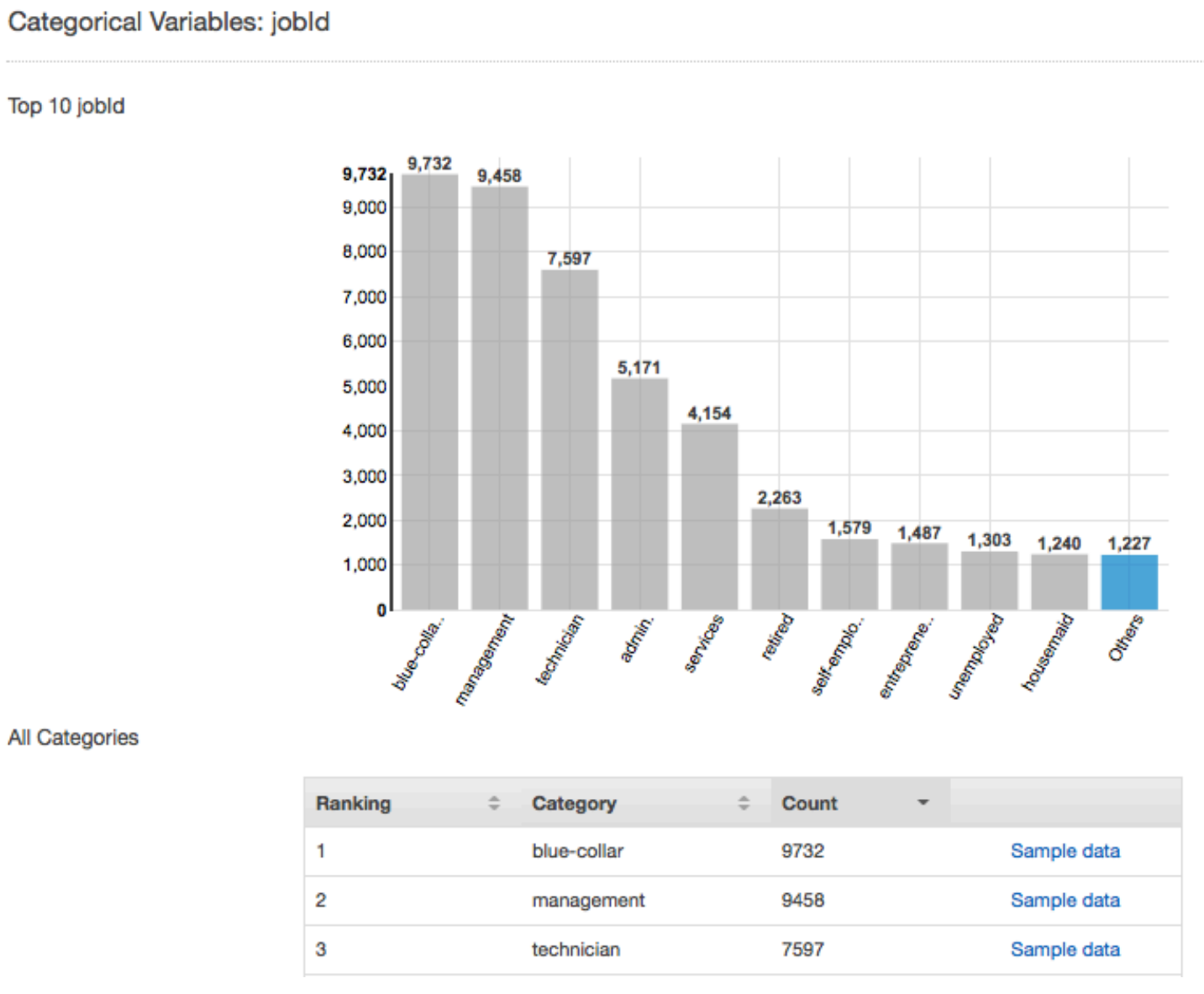
- The **Attributes** column shows the name of the variable. In this example, this column would say "review."
- The **Correlations to target** column exists only if a target is specified. Correlation measures the amount of information this attribute provides about the target. The higher the correlation, the more this attribute tells you about the target. Correlation is measured in terms of mutual information between a simplified representation of the text attribute and the target.
- The **Total words** column shows the number of words generated from tokenizing each record, delimiting the words by white space. In this example, this column would say "12".
- The **Unique words** column shows the number of unique words for an attribute. In this example, this column would say "10."
- The **Words in attribute (range)** column shows the number of words in a single row in the attribute. In this example, this column would say "0-6."
- The **Word length (range)** column shows the range of how many characters are in the words. In this example, this column would say "2-11."
- The **Most prominent words** column shows a ranked list of words that appear in the attribute. If there is a target attribute, words are ranked by their correlation to the target, meaning that the words that have the highest correlation are listed first. If no target is present in the data, then the words are ranked by their entropy.

Understanding the Distribution of Categorical and Binary Attributes

By clicking the **Preview** link associated with a categorical or binary attribute, you can view that attribute's distribution as well as the sample data from the input file for each categorical value of the attribute.

For example, the following screenshot shows the distribution for the categorical attribute jobId. The distribution displays the top 10 categorical values, with all other values grouped as "other". It

ranks each of the top 10 categorical values with the number of observations in the input file that contain that value, as well as a link to view sample observations from the input data file.

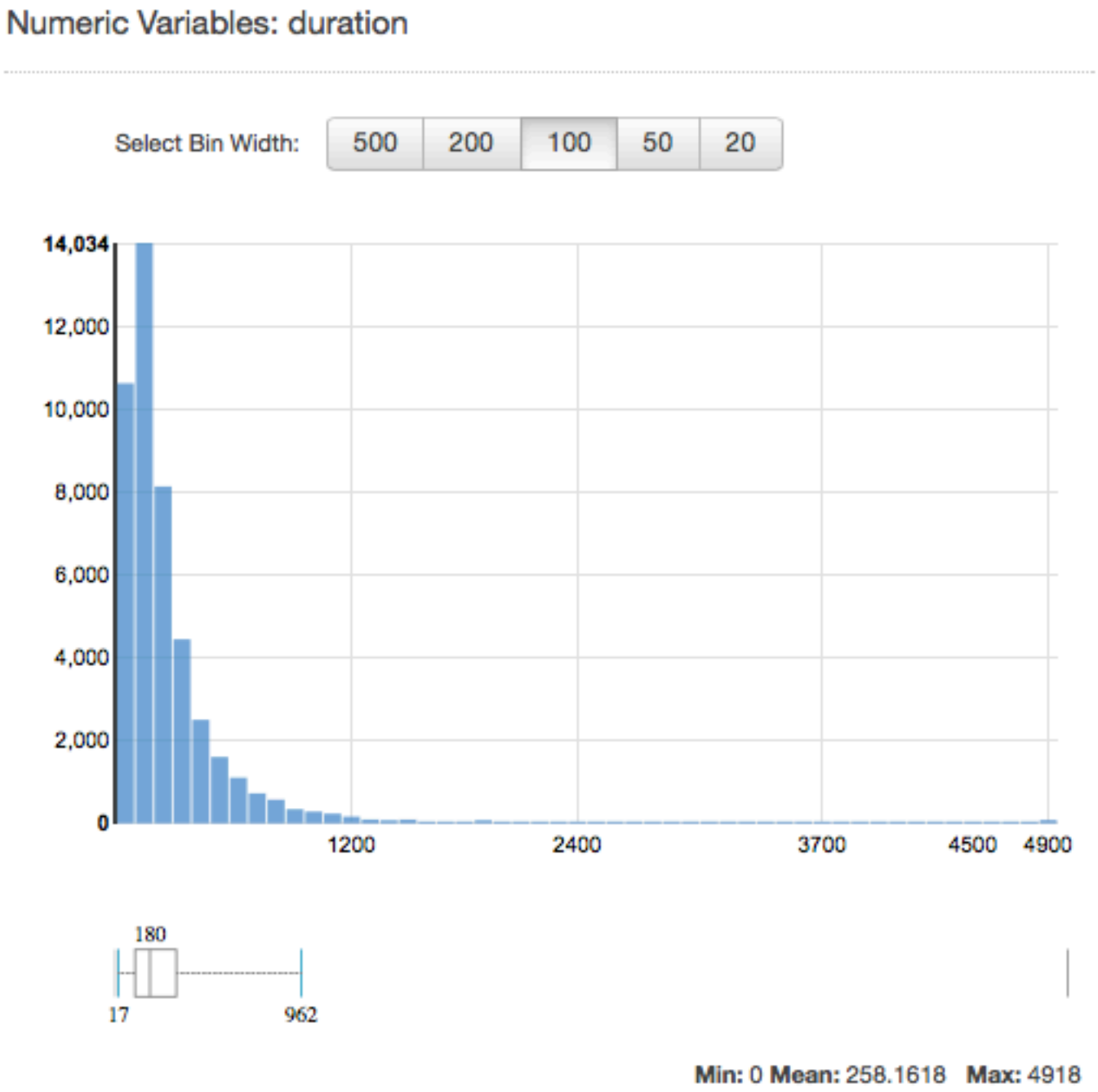


Ranking	Category	Count	
1	blue-collar	9732	Sample data
2	management	9458	Sample data
3	technician	7597	Sample data

Understanding the Distribution of Numeric Attributes

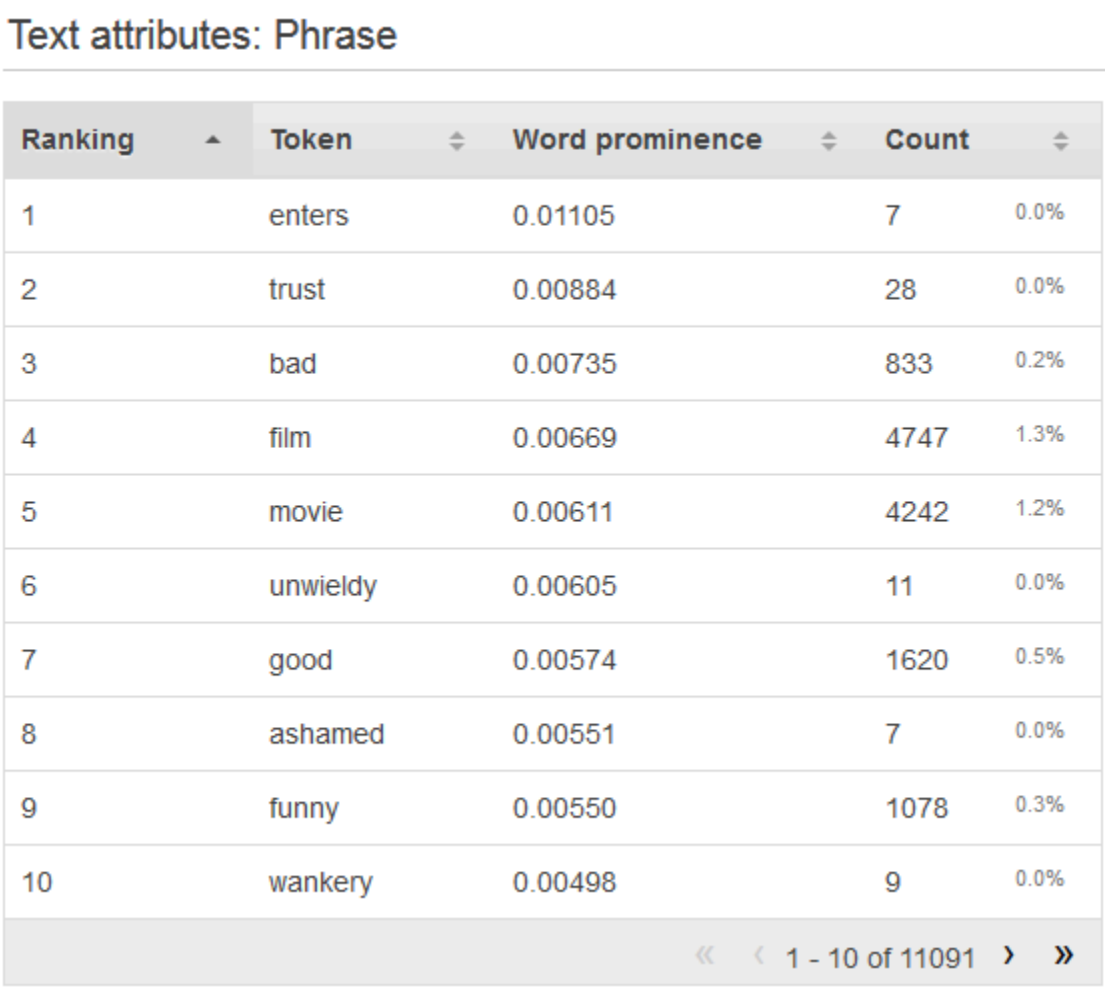
To view the distribution of a numeric attribute, click the **Preview** link of the attribute. When viewing the distribution of a numeric attribute, you can choose bin sizes of 500, 200, 100, 50, or 20. The larger the bin size, the smaller number of bar graphs that will be displayed. In addition, the resolution of the distribution will be coarse for large bin sizes. In contrast, setting the bucket size to 20 increases the resolution of the displayed distribution.

The minimum, mean, and maximum values are also displayed, as shown in the following screenshot.



Understanding the Distribution of Text Attributes

To view the distribution of a text attribute, click the **Preview** link of the attribute. When viewing the distribution of a text attribute, you will see the following information.



Ranking

Text tokens are ranked by the amount of information they convey, most informative to least informative.

Token

Token shows the word from the input text that the row of statistics is about.

Word prominence

If there is a target attribute, words are ranked by their correlation to the target, so that words that have the highest correlation are listed first. If no target is present in the data, then the words are ranked by their entropy, ie, the amount of information they can communicate.

Count number

Count number shows the number of input records that the token appeared in.

Count percentage

Count percentage shows the percentage of input data rows the token appeared in.

Using Amazon S3 with Amazon ML

Amazon Simple Storage Service (Amazon S3) is storage for the Internet. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web. Amazon ML uses Amazon S3 as a primary data repository for the following tasks:

- To access your input files to create datasource objects for training and evaluating your ML models.
- To access your input files to generate batch predictions.
- When you generate batch predictions by using your ML models, to output the prediction file to an S3 bucket that you specify.
- To copy data that you've stored in Amazon Redshift or Amazon Relational Database Service (Amazon RDS) into a .csv file and upload it to Amazon S3.

To enable Amazon ML to perform these tasks, you must grant permissions to Amazon ML to access your Amazon S3 data.

> **Note**
>
> You cannot output batch prediction files to an S3 bucket that accepts only server-side encrypted files. Make sure that your bucket policy allows uploading unencrypted files by confirming that the policy does not include a `Deny` effect for the `s3:PutObject` action when there is no `s3:x-amz-server-side-encryption` header in the request. For more information about S3 server-side encryption bucket policies, see Protecting Data Using Server-Side Encryption in the *Amazon Simple Storage Service User Guide*.

Uploading Your Data to Amazon S3

You must upload your input data to Amazon Simple Storage Service (Amazon S3) because Amazon ML reads data from Amazon S3 locations. You can upload your data directly to Amazon S3 (for example, from your computer), or Amazon ML can copy data that you've stored in Amazon Redshift or Amazon Relational Database Service (RDS) into a .csv file and upload it to Amazon S3.

For more information about copying your data from Amazon Redshift or Amazon RDS, see Using Amazon Redshift with Amazon ML or Using Amazon RDS with Amazon ML, respectively.

The remainder of this section describes how to upload your input data directly from your computer to Amazon S3. Before you begin the procedures in this section, you need to have your data in a .csv file. For information about how to correctly format your .csv file so that Amazon ML can use it, see Understanding the Data Format for Amazon ML.

To upload your data from your computer to Amazon S3

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3.
2. Create a bucket or choose an existing bucket.
   a. To create a bucket, choose **Create Bucket**. Name your bucket, choose a region (you can choose any available region), and then choose **Create**. For more information, see Create a Bucket in the *Amazon Simple Storage Getting Started Guide*.
   b. To use an existing bucket, search for the bucket by choosing the bucket in the **All Buckets** list. When the bucket name appears, select it, and then choose **Upload**.
3. In the **Upload** dialog box, choose **Add Files**.
4. Navigate to the folder that contains your input data .csv file, and then choose **Open**.

Permissions

To grant permissions for Amazon ML to access one of your S3 buckets, you must edit the bucket policy.

For information about granting Amazon ML permission to read data from your bucket in Amazon S3, see Granting Amazon ML Permissions to Read Your Data from Amazon S3.

For information about granting Amazon ML permission to output the batch prediction results to your bucket in Amazon S3, see Granting Amazon ML Permissions to Output Predictions to Amazon S3 .

For information about managing access permissions to Amazon S3 resources, see the Amazon S3 Developer Guide.

Creating an Amazon ML Datasource from Data in Amazon Redshift

If you have data stored in Amazon Redshift, you can use the **Create Datasource** wizard in the Amazon Machine Learning (Amazon ML) console to create a datasource object. When you create a datasource from Amazon Redshift data, you specify the cluster that contains your data and the SQL query to retrieve your data. Amazon ML executes the query by invoking the Amazon Redshift `Unload` command on the cluster. Amazon ML stores the results in the Amazon Simple Storage Service (Amazon S3) location of your choice, and then uses the data stored in Amazon S3 to create the datasource. The datasource, Amazon Redshift cluster, and S3 bucket must all be in the same region.

> **Note**
>
> Amazon ML doesn't support creating datasources from Amazon Redshift clusters in private VPCs. The cluster must have a public IP address.

Topics

Required Parameters for the Create Datasource Wizard

To allow Amazon ML to connect to your Amazon Redshift database and read data on your behalf, you must provide the following:

- The Amazon Redshift `ClusterIdentifier`
- The Amazon Redshift database name
- The Amazon Redshift database credentials (user name and password)
- The Amazon ML Amazon Redshift AWS Identity and Access Management (IAM) role
- The Amazon Redshift SQL query
- (Optional) The location of the Amazon ML schema

- The Amazon S3 staging location (where Amazon ML puts the data before it creates the datasource)

Additionally, you need to ensure that the IAM users or roles who create Amazon Redshift datasources (whether through the console or by using the `CreateDatasourceFromRedshift` action) have the `iam:PassRole` permission.

Amazon Redshift `ClusterIdentifier`

Use this case-sensitive parameter to enable Amazon ML to find and connect to your cluster. You can obtain the cluster identifier (name) from the Amazon Redshift console. For more information about clusters, see Amazon Redshift Clusters.

Amazon Redshift Database Name

Use this parameter to tell Amazon ML which database in the Amazon Redshift cluster contains the data that you want to use as your datasource.

Amazon Redshift Database Credentials

Use these parameters to specify the username and password of the Amazon Redshift database user in whose context the security query will be executed.

> **Note**
>
> Amazon ML requires an Amazon Redshift username and password to connect to your Amazon Redshift database. After unloading the data to Amazon S3, Amazon ML never reuses your password, nor does it store it.

Amazon ML Amazon Redshift Role

Use this parameter to specify the name of the IAM role that Amazon ML should use to configure the security groups for the Amazon Redshift cluster and the bucket policy for the Amazon S3 staging location.

If you don't have an IAM role that can access Amazon Redshift, Amazon ML can create a role for you. When Amazon ML creates a role, it creates and attaches a customer managed policy to an IAM role. The policy that Amazon ML creates grants Amazon ML permission to access only the cluster that you specify.

If you already have an IAM role to access Amazon Redshift, you can type the ARN of the role, or choose the role from the drop down list. IAM roles with Amazon Redshift access are listed at the top of the drop down.

The IAM role must have the following contents:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "Service": "machinelearning.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": { "aws:SourceAccount": "123456789012" },
           "ArnLike": { "aws:SourceArn": "arn:aws:machinelearning:us-
east-1:123456789012:datasource/*" }
        }
    }]
}
```

For more information about Customer Managed Policies, see Customer Managed Policies in the *IAM User Guide*.

Amazon Redshift SQL Query

Use this parameter to specify the SQL SELECT query that Amazon ML executes on your Amazon Redshift database to select your data. Amazon ML uses the Amazon Redshift UNLOAD action to securely copy the results of your query to an Amazon S3 location.

> **Note**
>
> Amazon ML works best when input records are in a random order (shuffled). You can easily shuffle the results of your Amazon Redshift SQL query by using the Amazon Redshift **random()** function. For example, let's say that this is the original query:
>
> ```
> "SELECT col1, col2, … FROM training_table"
> ```

You can embed random shuffling by updating the query like this:

```
"SELECT col1, col2, … FROM training_table ORDER BY random()"
```

Schema Location (Optional)

Use this parameter to specify the Amazon S3 path to your schema for the Amazon Redshift data that Amazon ML will export.

If you don't provide a schema for your datasource, the Amazon ML console automatically creates an Amazon ML schema based on the data schema of the Amazon Redshift SQL query. Amazon ML schemas have fewer data types than Amazon Redshift schemas, so it is not a one-to-one conversion. The Amazon ML console converts Amazon Redshift data types to Amazon ML data types using the following conversion scheme.

Amazon Redshift Data Types	Amazon Redshift Aliases	Amazon ML Data Type
SMALLINT	INT2	NUMERIC
INTEGER	INT, INT4	NUMERIC
BIGINT	INT8	NUMERIC
DECIMAL	NUMERIC	NUMERIC
REAL	FLOAT4	NUMERIC
DOUBLE PRECISION	FLOAT8, FLOAT	NUMERIC
BOOLEAN	BOOL	BINARY
CHAR	CHARACTER, NCHAR, BPCHAR	CATEGORICAL
VARCHAR	CHARACTER VARYING, NVARCHAR, TEXT	TEXT

Amazon Redshift Data Types	Amazon Redshift Aliases	Amazon ML Data Type
DATE		TEXT
TIMESTAMP	TIMESTAMP WITHOUT TIME ZONE	TEXT

To be converted to Amazon ML `Binary` data types, the values of the Amazon Redshift Booleans in your data must be supported Amazon ML Binary values. If your Boolean data type has unsupported values, Amazon ML converts them to the most specific data type it can. For example, if an Amazon Redshift Boolean has the values `0`, `1`, and `2`, Amazon ML converts the Boolean to a `Numeric` data type. For more information about supported binary values, see Using the AttributeType Field.

If Amazon ML can't figure out a data type, it defaults to `Text`.

After Amazon ML converts the schema, you can review and correct the assigned Amazon ML data types in the Create Datasource wizard, and revise the schema before Amazon ML creates the datasource.

Amazon S3 Staging Location

Use this parameter to specify the name of the Amazon S3 staging location where Amazon ML stores the results of the Amazon Redshift SQL query. After creating the datasource, Amazon ML uses the data in the staging location instead of returning to Amazon Redshift.

> **Note**
>
> Because Amazon ML assumes the IAM role defined by the Amazon ML Amazon Redshift role, Amazon ML has permissions to access any objects in the specified Amazon S3 staging location. Because of this, we recommend that you store only files that don't contain sensitive information in the Amazon S3 staging location. For example, if your root bucket is `s3://mybucket/`, we suggest that you create a location to store only the files that you want Amazon ML to access, such as `s3://mybucket/AmazonMLInput/`.

Creating a Datasource with Amazon Redshift Data (Console)

The Amazon ML console provides two ways to create a datasource using Amazon Redshift data. You can create a datasource by completing the Create Datasource wizard, or, if you already have a datasource created from Amazon Redshift data, you can copy the original datasource and modify its settings. Copying a datasource allows you to easily create multiple similar datasources.

For information about creating a datasource using the API, see CreateDataSourceFromRedshift.

For more information about the parameters in the following procedures, see Required Parameters for the Create Datasource Wizard.

Topics

Creating a Datasource (Console)

To unload data from Amazon Redshift into an Amazon ML datasource, use the Create Datasource wizard.

To create a datasource from data in Amazon Redshift

1. Open the Amazon Machine Learning console at https://console.aws.amazon.com/machinelearning/.
2. On the Amazon ML dashboard, under **Entities**, choose **Create new...**, and then choose **Datasource**.
3. On the **Input data** page, choose **Amazon Redshift**.
4. In the Create Datasource wizard, for **Cluster identifier**, type the name of your cluster.
5. For **Database name**, type the name of the Amazon Redshift database.
6. For **Database user name**, type your database username.
7. For **Database password**, type your database password.
8. For **IAM role**, choose your IAM role. If you don't already have one, choose **Create a new role**. Amazon ML creates an IAM Amazon Redshift role for you.
9. To test your Amazon Redshift settings, choose **Test Access** (next to **IAM role**). If Amazon ML can't connect to Amazon Redshift with the provided settings, you can't continue creating a datasource. For troubleshooting help, see Troubleshooting Errors.

10. For **SQL query**, type your SQL query.
11. For **Schema location**, choose whether you want Amazon ML to create a schema for you. If you have created a schema yourself, type the Amazon S3 path to your schema file.
12. For **Amazon S3 staging location**, type the Amazon S3 path to the bucket where you want Amazon ML to put the data it unloads from Amazon Redshift.
13. (Optional) For **Datasource name**, type a name for your datasource.
14. Choose **Verify**. Amazon ML verifies that it can connect to your Amazon Redshift database.
15. On the **Schema** page, review the data types for all attributes and correct them, as necessary.
16. Choose **Continue**.
17. If you want to use this datasource to create or evaluate an ML model, for **Do you plan to use this dataset to create or evaluate an ML model?**, choose **Yes**. If you choose **Yes**, choose your target row. For information about targets, see Using the targetAttributeName Field.

    If you want to use this datasource along with a model that you have already created to create predictions, choose **No**.
18. Choose **Continue**.
19. For **Does your data contain an identifier?**, if your data doesn't contain a row identifier, choose **No**.

    If your data does contain a row identifier, choose **Yes**. For information about row identifiers, see Using the rowID Field.
20. Choose **Review**.
21. On the **Review** page, review your settings, and then choose **Finish**.

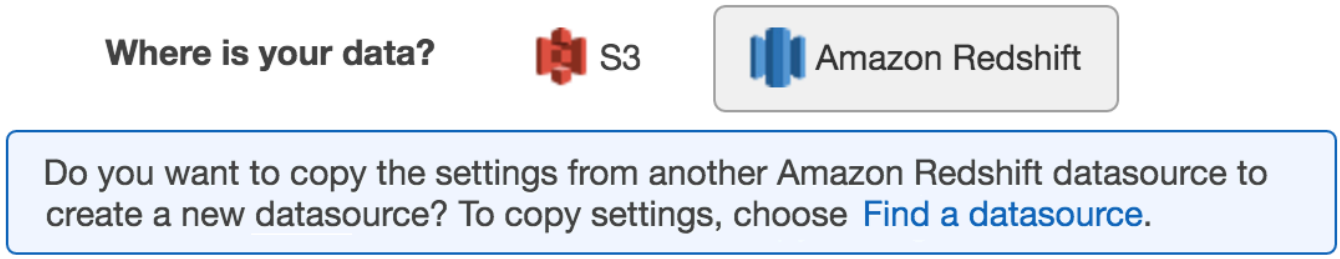
After you have created a datasource, you can use it to create an ML model. If you have already created a model, you can use the datasource to evaluate an ML model or generate predictions.

Copying a Datasource (Console)

When you want to create a datasource that is similar to an existing datasource, you can use the Amazon ML console to copy the original datasource and modify its settings. For example, you might choose to start with an existing datasource, and then modify the data schema to match your data more closely; change the SQL query used to unload data from Amazon Redshift; or specify a different AWS Identity and Access Management (IAM) user to access the Amazon Redshift cluster.

To copy and modify an Amazon Redshift datasource

1. Open the Amazon Machine Learning console at https://console.aws.amazon.com/machinelearning/.
2. On the Amazon ML dashboard, under **Entities**, choose **Create new...**, and then choose **Datasource**.
3. On the **Input data** page, for **Where is your data?**, choose **Amazon Redshift**. If you already have a datasource created from Amazon Redshift data, you have the option of copying settings from another datasource.



   If you don't already have a datasource created from Amazon Redshift data, this option doesn't appear.
4. Choose **Find a datasource**.
5. Select the datasource that you want to copy, and choose **Copy settings**. Amazon ML auto-populates most of the datasource settings with settings from the original datasource. It doesn't copy the database password, schema location, or datasource name from the original datasource.
6. Modify any of the auto-populated settings that you want to change. For example, if you want to change the data that Amazon ML unloads from Amazon Redshift, change the SQL query.
7. For **Database password**, type your database password. Amazon ML doesn't store or reuse your password, so you must always provide it.
8. (Optional) For **Schema location**, Amazon ML pre-selects **I want Amazon ML to generate a recommended schema** for you. If you have already created a schema, choose **I want to use the schema that I created and stored in Amazon S3** and type the path to your schema file in Amazon S3.
9. (Optional) For **Datasource name**, type a name for your datasource. Otherwise, Amazon ML generates a new datasource name for you.
10. Choose **Verify**. Amazon ML verifies that it can connect to your Amazon Redshift database.

11. (Optional) If Amazon ML inferred the schema for you, on the **Schema** page, review the data types for all attributes and correct them, as necessary.
12. Choose **Continue**.
13. If you want to use this datasource to create or evaluate an ML model, for **Do you plan to use this dataset to create or evaluate an ML model?**, choose **Yes**. If you choose **Yes**, choose your target row. For information about targets, see Using the targetAttributeName Field.

    If you want to use this datasource along with a model that you have already created to create predictions, choose **No**.
14. Choose **Continue**.
15. For **Does your data contain an identifier?**, if your data doesn't contain a row identifier, choose **No**.

    If your data contains a row identifier, choose **Yes**, and select the row that you want to use as an identifier. For information about row identifiers, see Using the rowID Field.
16. Choose **Review**.
17. Review your settings, and then choose **Finish**.

After you have created a datasource, you can use it to create an ML model. If you have already created a model, you can use the datasource to evaluate an ML model or generate predictions.

Troubleshooting Amazon Redshift Issues

As you create your Amazon Redshift datasource, ML models, and evaluation, Amazon Machine Learning (Amazon ML) reports the status of your Amazon ML objects in the Amazon ML console. If Amazon ML returns error messages, use the following information and resources to troubleshoot the issues.

For answers to general questions about Amazon ML, see the Amazon Machine Learning FAQs. You can also search for answers and post questions in the Amazon Machine Learning forum.

Topics

- Troubleshooting Errors
- Contacting AWS Support

Troubleshooting Errors

The format of the role is invalid. Provide a valid IAM role. For example, arn:aws:iam::YourAccountID:role/YourRedshiftRole.

Cause

The format of the Amazon Resource Name (ARN) of your IAM role is incorrect.

Solution

In the Create Datasource wizard, correct the ARN for your role. For information about formatting role ARNs, see IAM ARNs in the *IAM User Guide*. The region is optional for IAM role ARNs.

The role is invalid. Amazon ML can't assume the <role ARN> IAM role. Provide a valid IAM role and make it accessible to Amazon ML.

Cause

Your role isn't set up to allow Amazon ML to assume it.

Solution

In the IAM console, edit your role so that it has a trust policy that allows Amazon ML to assume the role attached to it.

This <user ARN> user is not authorized to pass the <role ARN> IAM role.

Cause

Your IAM user doesn't have a permissions policy that allows it to pass a role to Amazon ML.

Solution

Attach a permissions policy to your IAM user that allows you to pass roles to Amazon ML. You can attach a permissions policy to your IAM user in the IAM console.

Passing an IAM role across accounts isn't allowed. The IAM role must belong to this account.

Cause

You can't pass a role that belongs to another IAM account.

Solution

Sign in to the AWS account that you used to create the role. You can see your IAM roles in your IAM console.

The specified role doesn't have permissions to perform the operation. Provide a role that has a policy that provides Amazon ML the required permissions.

Cause

Your IAM role doesn't have the permissions to perform the requested operation.

Solution

Edit the permission policy attached to your role in the IAM console to provide the required permissions.

Amazon ML can't configure a security group on that Amazon Redshift cluster with the specified IAM role.

Cause

Your IAM role doesn't have the permissions required to configure an Amazon Redshift security cluster.

Solution

Edit the permission policy attached to your role in the IAM console to provide the required permissions.

An error occurred when Amazon ML attempted to configure a security group on your cluster. Try again later.

Cause

When Amazon ML tried to connect to your Amazon Redshift cluster, it encountered an issue.

Solution

Verify that the IAM role that you provided in the Create Datasource wizard has all of the required permissions.

The format of the cluster ID is invalid. Cluster IDs must begin with a letter and must contain only alphanumeric characters and hyphens. They can't contain two consecutive hyphens or end with a hyphen.

Cause

Your Amazon Redshift cluster ID format is incorrect.

Solution

In the Create Datasource wizard, correct your cluster ID so that it contains only alphanumeric characters and hyphens and doesn't contain two consecutive hyphens or end with a hyphen.

There is no <Amazon Redshift cluster name> cluster, or the cluster is not in the same region as your Amazon ML service. Specify a cluster in the same region as this Amazon ML.

Cause

Amazon ML can't find your Amazon Redshift cluster because it's not located in the region where you are creating an Amazon ML datasource.

Solution

Verify that your cluster exists on the Amazon Redshift console Clusters page, that you are creating a datasource in the same region where your Amazon Redshift cluster is located, and that the cluster ID specified in the Create Datasource wizard is correct.

Amazon ML can't read the data in your Amazon Redshift cluster. Provide the correct Amazon Redshift cluster ID.

Cause

Amazon ML can't read the data in the Amazon Redshift cluster that you specified.

Solution

In the Create Datasource wizard, specify the correct Amazon Redshift cluster ID, verify that you are creating a datasource in the same region that has your Amazon Redshift cluster, and that your cluster is listed on the Amazon Redshift Clusters page.

The <Amazon Redshift cluster name> cluster isn't publicly accessible.

Cause

Amazon ML can't access your cluster because the cluster is not publicly accessible and does not have a public IP address.

Solution

Make the cluster publicly accessible and give it a public IP address. For information about making clusters publicly accessible, see Modifying a Cluster in the *Amazon Redshift Management Guide*.

The <Redshift> cluster status isn't available to Amazon ML. Use the Amazon Redshift console to view and resolve this cluster status issue. The cluster status must be "available."

Cause

Amazon ML can't see the cluster status.

Solution

Make sure that your cluster is available. For information on checking the status of your cluster, see Getting an Overview of Cluster Status in the *Amazon Redshift Management Guide*. For information on rebooting the cluster so that it is available, see Rebooting a Cluster in the *Amazon Redshift Management Guide*.

There is no <database name> database in this cluster. Verify that the database name is correct or specify another cluster and database.

Cause

Amazon ML can't find the specified database in the specified cluster.

Solution

Verify that the database name entered in the Create Datasource wizard is correct, or specify the correct cluster and database names.

Amazon ML couldn't access your database. Provide a valid password for database user <user name>.

Cause

The password you provided in the Create Datasource wizard to allow Amazon ML to access your Amazon Redshift database is incorrect.

Solution

Provide the correct password for your Amazon Redshift database user.

An error occurred when Amazon ML attempted to validate the query.

Cause

There's an issue with your SQL query.

Solution

Verify that your query is valid SQL.

An error occurred when executing your SQL query. Verify the database name and the provided query. Root cause: {serverMessage}.

Cause

Amazon Redshift was unable to run your query.

Solution

Verify that you specified the correct database name in the Create Datasource wizard, and that your query is valid SQL.

An error occurred when executing your SQL query. Root cause: {serverMessage}.

Cause

Amazon Redshift was unable to find the specified table.

Solution

Verify that the table you specified in the Create Datasource wizard is present in your Amazon Redshift cluster database, and that you entered the correct cluster ID, database name, and SQL query.

Contacting AWS Support

If you have AWS Premium Support, you can create a technical support case at the AWS Support Center.

Using Data from an Amazon RDS Database to Create an Amazon ML Datasource

Amazon ML allows you to create a datasource object from data stored in a MySQL database in Amazon Relational Database Service (Amazon RDS). When you perform this action, Amazon ML creates an AWS Data Pipeline object that executes the SQL query that you specify, and places the output into an S3 bucket of your choice. Amazon ML uses that data to create the datasource.

> **Note**
>
> Amazon ML supports only MySQL databases in VPCs.

Before Amazon ML can read your input data, you must export that data to Amazon Simple Storage Service (Amazon S3). You can set up Amazon ML to perform the export for you by using the API. (RDS is limited to the API, and is not available from the console.)

In order for Amazon ML to connect to your MySQL database in Amazon RDS and read data on your behalf, you need to provide the following:

- The RDS DB instance identifier
- The MySQL database name
- The AWS Identity and Access Management (IAM) role that is used to create, activate, and execute the data pipeline
- The database user credentials:
  - User name
  - Password
- The AWS Data Pipeline security information:
  - The IAM resource role
  - The IAM service role
- The Amazon RDS security information:
  - The subnet ID
  - The security group IDs
- The SQL query that specifies the data that you want to use to create the datasource
- The S3 output location (bucket) used to store the results of the query
- (Optional) The location of the data schema file

Additionally, you need to ensure that the IAM users or roles that create Amazon RDS datasources by using the CreateDataSourceFromRDS operation have the `iam:PassRole` permission. For more information, see Controlling Access to Amazon ML Resources -with IAM.

Topics

RDS Database Instance Identifier

The RDS DB instance identifier is a unique name that you supply that identifies the database instance that Amazon ML should use when interacting with Amazon RDS. You can find the RDS DB instance identifier in the Amazon RDS console.

MySQL Database Name

MySQL Database Name specifies the name of the MySQL database in the RDS DB instance.

Database User Credentials

To connect to the RDS DB instance, you must supply the user name and password of the database user who has sufficient permissions to execute the SQL query that you provide.

AWS Data Pipeline Security Information

To enable secure AWS Data Pipeline access, you must provide the names of the IAM resource role and the IAM service role.

An EC2 instance assumes the resource role to copy data from Amazon RDS to Amazon S3. The easiest way to create this resource role is by using the `DataPipelineDefaultResourceRole` template, and listing **`machinelearning.aws.com`** as a trusted service. For more information about the template, see Setting Up IAM roles in the *AWS Data Pipeline Developer Guide*.

If you create your own role, it must have the following contents:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "Service": "machinelearning.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": { "aws:SourceAccount": "123456789012" },
            "ArnLike": { "aws:SourceArn": "arn:aws:machinelearning:us-
east-1:123456789012:datasource/*" }
        }
    }]
}
```

AWS Data Pipeline assumes the service role to monitor the progress of copying data from Amazon RDS to Amazon S3. The easiest way to create this resource role is by using the `DataPipelineDefaultRole` template, and listing `machinelearning.aws.com` as a trusted service. For more information about the template, see Setting Up IAM roles in the *AWS Data Pipeline Developer Guide*.

Amazon RDS Security Information

To enable secure Amazon RDS access, you need to provide the `VPC Subnet ID` and `RDS Security Group IDs`. You also need to set up appropriate ingress rules for the VPC subnet that is pointed at by the `Subnet ID` parameter, and provide the ID of the security group that has this permission.

MySQL SQL Query

The `MySQL SQL Query` parameter specifies the SQL SELECT query that you want to execute on your MySQL database. The results of the query is copied to the S3 output location (bucket) that you specify.

> **Note**
>
> Machine learning technology works best when input records are presented in random order (shuffled). You can easily shuffle the results of your MySQL SQL query by using the `rand()` function. For example, let's say that this is the original query:
> "SELECT col1, col2, ... FROM training_table"
> You can add random shuffling by updating the query like this:
> "SELECT col1, col2, ... FROM training_table ORDER BY rand()"

S3 Output Location

The `S3 Output Location` parameter specifies the name of the "staging" Amazon S3 location where the results of the MySQL SQL query is output.

> **Note**
>
> You need to ensure that Amazon ML has permissions to read data from this location once the data is exported from Amazon RDS. For information about setting these permissions, see Granting Amazon ML Permissions to Read Your Data from Amazon S3 .

Training ML Models

The process of training an ML model involves providing an ML algorithm (that is, the *learning algorithm*) with training data to learn from. The term *ML model* refers to the model artifact that is created by the training process.

The training data must contain the correct answer, which is known as a *target* or *target attribute*. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer that you want to predict), and it outputs an ML model that captures these patterns.

You can use the ML model to get predictions on new data for which you do not know the target. For example, let's say that you want to train an ML model to predict if an email is spam or not spam. You would provide Amazon ML with training data that contains emails for which you know the target (that is, a label that tells whether an email is spam or not spam). Amazon ML would train an ML model by using this data, resulting in a model that attempts to predict whether new email will be spam or not spam.

For general information about ML models and ML algorithms, see Machine Learning Concepts.

Topics

Types of ML Models

Amazon ML supports three types of ML models: binary classification, multiclass classification, and regression. The type of model you should choose depends on the type of target that you want to predict.

Binary Classification Model

ML models for binary classification problems predict a binary outcome (one of two possible classes). To train binary classification models, Amazon ML uses the industry-standard learning algorithm known as logistic regression.

Examples of Binary Classification Problems

- "Is this email spam or not spam?"
- "Will the customer buy this product?"
- "Is this product a book or a farm animal?"
- "Is this review written by a customer or a robot?"

Multiclass Classification Model

ML models for multiclass classification problems allow you to generate predictions for multiple classes (predict one of more than two outcomes). For training multiclass models, Amazon ML uses the industry-standard learning algorithm known as multinomial logistic regression.

Examples of Multiclass Problems

- "Is this product a book, movie, or clothing?"
- "Is this movie a romantic comedy, documentary, or thriller?"
- "Which category of products is most interesting to this customer?"

Regression Model

ML models for regression problems predict a numeric value. For training regression models, Amazon ML uses the industry-standard learning algorithm known as linear regression.

Examples of Regression Problems

- "What will the temperature be in Seattle tomorrow?"
- "For this product, how many units will sell?"
- "What price will this house sell for?"

Training Process

To train an ML model, you need to specify the following:

- Input training datasource
- Name of the data attribute that contains the target to be predicted
- Required data transformation instructions

- Training parameters to control the learning algorithm

During the training process, Amazon ML automatically selects the correct learning algorithm for you, based on the type of target that you specified in the training datasource.

Training Parameters

Typically, machine learning algorithms accept parameters that can be used to control certain properties of the training process and of the resulting ML model. In Amazon Machine Learning, these are called *training parameters*. You can set these parameters using the Amazon ML console, API, or command line interface (CLI). If you do not set any parameters, Amazon ML will use default values that are known to work well for a large range of machine learning tasks.

You can specify values for the following training parameters:

- Maximum model size
- Maximum number of passes over training data
- Shuffle type
- Regularization type
- Regularization amount

In the Amazon ML console, the training parameters are set by default. The default settings are adequate for most ML problems, but you can choose other values to fine-tune the performance. Certain other training parameters, such as the learning rate, are configured for you based on your data.

The following sections provide more information about the training parameters.

Maximum Model Size

The maximum model size is the total size, in units of bytes, of patterns that Amazon ML creates during the training of an ML model.

By default, Amazon ML creates a 100 MB model. You can instruct Amazon ML to create a smaller or larger model by specifying a different size. For the range of available sizes, see Types of ML Models

If Amazon ML can't find enough patterns to fill the model size, it creates a smaller model. For example, if you specify a maximum model size of 100 MB, but Amazon ML finds patterns that total

only 50 MB, the resulting model will be 50 MB. If Amazon ML finds more patterns than will fit into the specified size, it enforces a maximum cut-off by trimming the patterns that least affect the quality of the learned model.

Choosing the model size allows you to control the trade-off between a model's predictive quality and the cost of use. Smaller models can cause Amazon ML to remove many patterns to fit within the maximum size limit, affecting the quality of predictions. Larger models, on the other hand, cost more to query for real-time predictions.

> **Note**
>
> If you use an ML model to generate real-time predictions, you will incur a small capacity reservation charge that is determined by the model's size. For more information, see Pricing for Amazon ML.

Larger input data sets do not necessarily result in larger models because models store patterns, not input data; if the patterns are few and simple, the resulting model will be small. Input data that has a large number of raw attributes (input columns) or derived features (outputs of the Amazon ML data transformations) will likely have more patterns found and stored during the training process. Picking the correct model size for your data and problem is best approached with a few experiments. The Amazon ML model training log (which you can download from the console or through the API) contains messages about how much model trimming (if any) occurred during the training process, allowing you to estimate the potential hit-to-prediction quality.

Maximum Number of Passes over the Data

For best results, Amazon ML may need to make multiple passes over your data to discover patterns. By default, Amazon ML makes 10 passes, but you can change the default by setting a number up to 100. Amazon ML keeps track of the quality of patterns (model convergence) as it goes along, and automatically stops the training when there are no more data points or patterns to discover. For example, if you set the number of passes to 20, but Amazon ML discovers that no new patterns can be found by the end of 15 passes, then it will stop the training at 15 passes.

In general, data sets with only a few observations typically require more passes over the data to obtain higher model quality. Larger data sets often contain many similar data points, which eliminates the need for a large number of passes. The impact of choosing more data passes over your data is two-fold: model training takes longer, and it costs more.

Shuffle Type for Training Data

In Amazon ML, you must shuffle your training data. Shuffling mixes up the order of your data so that the SGD algorithm doesn't encounter one type of data for too many observations in succession. For example, if you are training an ML model to predict a product type, and your training data includes movie, toy, and video game product types, if you sorted the data by the product type column before uploading it, the algorithm sees the data alphabetically by product type. The algorithm sees all of your data for movies first, and your ML model begins to learn patterns for movies. Then, when your model encounters data on toys, every update that the algorithm makes would fit the model to the toy product type, even if those updates degrade the patterns that fit movies. This sudden switch from movie to toy type can produce a model that doesn't learn how to predict product types accurately.

You must shuffle your training data even if you chose the random split option when you split the input datasource into training and evaluation portions. The random split strategy chooses a random subset of the data for each datasource, but it doesn't change the order of the rows in the datasource. For more information about splitting your data, see Splitting Your Data.

When you create an ML model using the console, Amazon ML defaults to shuffling the data with a pseudo-random shuffling technique. Regardless of the number of passes requested, Amazon ML shuffles the data only once before training the ML model. If you shuffled your data before providing it to Amazon ML and don't want Amazon ML to shuffle your data again, you can set the **Shuffle type** to `none`. For example, if you randomly shuffled the records in your .csv file before uploading it to Amazon S3, used the `rand()` function in your MySQL SQL query when creating your datasource from Amazon RDS, or used the `random()` function in your Amazon Redshift SQL query when creating your datasource from Amazon Redshift, setting **Shuffle type** to `none` won't impact the predictive accuracy of your ML model. Shuffling your data only once reduces the run-time and cost for creating an ML model.

> **Important**
>
> When you create an ML model using the Amazon ML API, Amazon ML doesn't shuffle your data by default. If you use the API instead of the console to create your ML model, we strongly recommend that you shuffle your data by setting the `sgd.shuffleType` parameter to `auto`.

Regularization Type and Amount

The predictive performance of complex ML models (those having many input attributes) suffers when the data contains too many patterns. As the number of patterns increases, so does the likelihood that the model learns unintentional data artifacts, rather than true data patterns. In such a case, the model does very well on the training data, but can't generalize well on new data. This phenomenon is known as *overfitting* the training data.

Regularization helps prevent linear models from overfitting training data examples by penalizing extreme weight values. L1 regularization reduces the number of features used in the model by pushing the weight of features that would otherwise have very small weights to zero. L1 regularization produces sparse models and reduces the amount of noise in the model. L2 regularization results in smaller overall weight values, which stabilizes the weights when there is high correlation between the features. You can control the amount of L1 or L2 regularization by using the `Regularization amount` parameter. Specifying an extremely large `Regularization amount` value can cause all features to have zero weight.

Selecting and tuning the optimal regularization value is an active subject in machine learning research. You will probably benefit from selecting a moderate amount of L2 regularization, which is the default in the Amazon ML console. Advanced users can choose between three types of regularization (none, L1, or L2) and amount. For more information about regularization, go to Regularization (mathematics).

Training Parameters: Types and Default Values

The following table lists the Amazon ML training parameters, along with the default values and the allowable range for each.

Training Parameter	Type	Default Value	Description
maxMLModelSizeInBytes	Integer	100,000,000 bytes (100 MiB)	Allowable range: 100,000 (100 KiB) to 2,147,483,648 (2 GiB) Depending on the input data, the model size might affect the performance.
sgd.maxPasses	Integer	10	Allowable range: 1-100

Training Parameter	Type	Default Value	Description
sgd.shuffleType	String	auto	Allowable values: `auto` or none
sgd.l1Regularizationamount	Double	0 (By default, L1 isn't used)	Allowable range: 0 to MAX_DOUBLE L1 values between 1E-4 and 1E-8 have been found to produce good results. Larger values are likely to produce models that aren't very useful. You can't set both L1 and L2. You must choose one or the other.
sgd.l2Regularizationamount	Double	1E-6 (By default, L2 is used with this amount of regularization)	Allowable range: 0 to MAX_DOUBLE L2 values between 1E-2 and 1E-6 have been found to produce good results. Larger values are likely to produce models that aren't very useful. You can't set both L1 and L2. You must choose one or the other.

Creating an ML Model

After you've created a datasource, you are ready to create an ML model. If you use the Amazon Machine Learning console to create a model, you can choose to use the default settings or you customize your model by applying custom options.

Custom options include:

- Evaluation settings: You can choose to have Amazon ML reserve a portion of the input data to evaluate the predictive quality of the ML model. For information about evaluations, see Evaluating ML Models.

- A recipe: A recipe tells Amazon ML which attributes and attribute transformations are available for model training. For information about Amazon ML recipes, see Feature Transformations with Data Recipes.
- Training parameters: Parameters control certain properties of the training process and of the resulting ML model. For more information about training parameters, see Training Parameters.

To select or specify values for these settings, choose the **Custom** option when you use the Create ML Model wizard. If you want Amazon ML to apply the default settings, choose **Default**.

When you create an ML model, Amazon ML selects the type of learning algorithm it will use based on the attribute type of your target attribute. (The target attribute is the attribute that contains the "correct" answers.) If your target attribute is Binary, Amazon ML creates a binary classification model, which uses the logistic regression algorithm. If your target attribute is Categorical, Amazon ML creates a multiclass model, which uses a multinomial logistic regression algorithm. If your target attribute is Numeric, Amazon ML creates a regression model, which uses a linear regression algorithm.

Topics

Prerequisites

Before using the Amazon ML console to create an ML model, you need to create two datasources, one for training the model and one for evaluating the model. If you haven't created two datasources, see Step 2: Create a Training Datasource in the tutorial.

Creating an ML Model with Default Options

Choose the **Default** options, if you want Amazon ML to:

- Split the input data to use the first 70 percent for training and use the remaining 30 percent for evaluation
- Suggest a recipe based on statistics collected on the training datasource, which is 70 percent of the input datasource

- Choose default training parameters

To choose default options

1. In the Amazon ML console, choose **Amazon Machine Learning**, and then choose **ML models**.
2. On the **ML models** summary page, choose **Create a new ML model**.
3. On the **Input data** page, make sure that **I already created a datasource pointing to my S3 data** is selected.
4. In the table, choose your datasource, and then choose **Continue**.
5. On the **ML model settings** page, for **ML model name**, type a name for your ML model.
6. For **Training and evaluation settings**, make sure that **Default** is selected.
7. For **Name this evaluation**, type a name for the evaluation, and then choose **Review**. Amazon ML bypasses the rest of the wizard and takes you to the **Review** page.
8. Review your data, delete any tags copied from the datasource that you don't want applied to your model and evaluations, and then choose **Finish**.

Creating an ML Model with Custom Options

Customizing your ML model allows you to:

- Provide your own recipe. For information about how to provide your own recipe, see Recipe Format Reference.
- Choose training parameters. For more information about training parameters, see Training Parameters.
- Choose a training/evaluation splitting ratio other than the default 70/30 ratio or provide another datasource that you have already prepared for evaluation. For information about splitting strategies, see Splitting Your Data.

You can also choose the default values for any of these settings.

If you've already created a model using the default options and want to improve your model's predictive performance, use the **Custom** option to create a new model with some customized settings. For example, you might add more feature transformations to the recipe or increase the number of passes in the training parameter.

To create a model with custom options

1. In the Amazon ML console, choose **Amazon Machine Learning**, and then choose **ML models**.
2. On the **ML models** summary page, choose **Create a new ML model**.
3. If you have already created a datasource, on the **Input data** page, choose **I already created a datasource pointing to my S3 data**. In the table, choose your datasource, and then choose **Continue**.

   If you need to create a datasource, choose **My data is in S3, and I need to create a datasource**, choose **Continue**. You are redirected to the **Create a Datasource** wizard. Specify whether your data is in **S3** or **Redshift**, then choose **Verify**. Complete the procedure for creating a datasource.

   After you have created a datasource, you are redirected to the next step in the **Create ML Model** wizard.
4. On the **ML model settings** page, for **ML model name**, type a name for your ML model.
5. In **Select training and evaluation settings**, choose **Custom**, and then choose **Continue**.
6. On the **Recipe** page, you can customize a recipe. If you don't want to customize a recipe, Amazon ML suggests one for you. Choose **Continue**.
7. On the **Advanced settings** page, specify the **Maximum ML model Size**, the **Maximum number of data passes**, the **Shuffle type for training data**, the **Regularization type**, and the **Regularization amount**. If you don't specify these, Amazon ML uses the default training parameters.

   For more information about these parameters and their defaults, see Training Parameters.

   Choose **Continue**.
8. On the **Evaluation** page, specify whether you want to evaluate the ML model immediately. If you don't want to evaluate the ML model now, choose **Review**.

   If you want to evaluate the ML model now:

   a. For **Name this evaluation**, type a name for the evaluation.

   b. For **Select evaluation data**, choose whether you want Amazon ML to reserve a portion of the input data for evaluation and, if you do, how you want to split the datasource, or choose to provide a different datasource for evaluation.

   c. Choose **Review**.

9. On the **Review** page, edit your selections, delete any tags copied from the datasource that you don't want applied to your model and evaluations, and then choose **Finish**.

After you have created the model, see Step 4: Review the ML Model's Predictive Performance and Set a Score Threshold.

Data Transformations for Machine Learning

Machine learning models are only as good as the data that is used to train them. A key characteristic of good training data is that it is provided in a way that is optimized for learning and generalization. The process of putting together the data in this optimal format is known in the industry as *feature transformation*.

Topics

Importance of Feature Transformation

Consider a machine learning model whose task is to decide whether a credit card transaction is fraudulent or not. Based on your application background knowledge and data analysis, you might decide which data fields (or features) are important to include in the input data. For example, transaction amount, merchant name, address, and credit card owner's address are important to provide to the learning process. On the other hand, a randomly generated transaction ID carries no information (if we know that it really is random), and is not useful.

Once you have decided on which fields to include, you transform these features to help the learning process. Transformations add background experience to the input data, enabling the machine learning model to benefit from this experience. For example, the following merchant address is represented as a string:

"123 Main Street, Seattle, WA 98101"

By itself, the address has limited expressive power – it is useful only for learning patterns associated with that exact address. Breaking it up into constituent parts, however, can create additional features like "Address" (123 Main Street), "City" (Seattle), "State" (WA) and "Zip" (98101). Now, the learning algorithm can group more disparate transactions together, and discover broader patterns – perhaps some merchant zip codes experience more fraudulent activity than others.

For more information about the feature transformation approach and process, see Machine Learning Concepts.

Feature Transformations with Data Recipes

There are two ways to transform features before creating ML models with Amazon ML: you can transform your input data directly before showing it to Amazon ML, or you can use the built-in data transformations of Amazon ML. You can use Amazon ML recipes, which are pre-formatted instructions for common transformations. With recipes, you can do the following:

- Choose from a list of built-in common machine learning transformations, and apply these to individual variables or groups of variables
- Select which of the input variables and transformations are made available to the machine learning process

Using Amazon ML recipes offers several advantages. Amazon ML performs the data transformations for you, so you do not need to implement them yourself. In addition, they are fast because Amazon ML applies the transformations while reading input data, and provides results to the learning process without the intermediate step of saving results to disk.

Recipe Format Reference

Amazon ML recipes contain instructions for transforming your data as a part of the machine learning process. Recipes are defined using a JSON-like syntax, but they have additional restrictions beyond the normal JSON restrictions. Recipes have the following sections, which must appear in the order shown here:

- **Groups** enable grouping of multiple variables, for ease of applying transformations. For example, you can create a group of all variables having to do with free-text parts of a web page (title, body), and then perform a transformation on all of these parts at once.
- **Assignments** enable the creation of intermediate named variables that can be reused in processing.
- **Outputs** define which variables will be used in the learning process, and what transformations (if any) apply to these variables.

Groups

You can define groups of variables in order to collectively transform all variables within the groups, or to use these variables for machine learning without transforming them. By default, Amazon ML creates the following groups for you:

ALL_TEXT, ALL_NUMERIC, ALL_CATEGORICAL, ALL_BINARY –Type-specific groups based on variables defined in the datasource schema.

> **Note**
>
> You cannot create a group with `ALL_INPUTS`.

These variables can be used in the outputs section of your recipe without being defined. You can also create custom groups by adding to or subtracting variables from existing groups, or directly from a collection of variables. In the following example, we demonstrate all three approaches, and the syntax for the grouping assignment:

```
"groups": {

"Custom_Group": "group(var1, var2)",
"All_Categorical_plus_one_other": "group(ALL_CATEGORICAL, var2)"

}
```

Group names need to start with an alphabetical character and can be between 1 and 64 characters long. If the group name does not start with an alphabetical character or if it contains special characters (, ' " \t \r \n ( ) \), then the name needs to be quoted to be included in the recipe.

Assignments

You can assign one or more transformations to an intermediate variable, for convenience and readability. For example, if you have a text variable named email_subject, and you apply the lowercase transformation to it, you can name the resulting variable email_subject_lowercase, making it easy to keep track of it elsewhere in the recipe. Assignments can also be chained, enabling you to apply multiple transformations in a specified order. The following example shows single and chained assignments in recipe syntax:

```
"assignments": {

"email_subject_lowercase": "lowercase(email_subject)",

"email_subject_lowercase_ngram":"ngram(lowercase(email_subject), 2)"

}
```

Intermediate variable names need to start with an alphabet character and can be between 1 and 64 characters long. If the name does not start with an alphabet or if it contains special characters (, ' " \t \r \n ( ) \), then the name needs to be quoted to be included in the recipe.

Outputs

The outputs section controls which input variables will be used for the learning process, and which transformations apply to them. An empty or non-existent output section is an error, because no data will be passed to the learning process.

The simplest outputs section simply includes the predefined **ALL_INPUTS** group, instructing Amazon ML to use all of the variables defined in the datasource for learning:

```
"outputs": [

"ALL_INPUTS"

]
```

The output section can also refer to the other predefined groups by instructing Amazon ML to use all the variables in these groups:

```
"outputs": [

"ALL_NUMERIC",

"ALL_CATEGORICAL"

]
```

The output section can also refer to custom groups. In the following example, only one of the custom groups defined in the grouping assignments section in the preceding example will be used for machine learning. All other variables will be dropped:

```
"outputs": [

"All_Categorical_plus_one_other"

]
```

The outputs section can also refer to variable assignments defined in the assignment section:

```
"outputs": [

"email_subject_lowercase"

]
```

And input variables or transformations can be defined directly in the outputs section:

```

"outputs": [

"var1",

"lowercase(var2)"

]
```

Output needs to explicitly specify all variables and transformed variables that are expected to be available to the learning process. Say, for example, that you include in the output a Cartesian product of var1 and var2. If you would like to include both the raw variables var1 and var2 as well, then you need to add the raw variables in the output section:

```

"outputs": [

"cartesian(var1,var2)",

"var1",
```

```
"var2"

]
```

Outputs can include comments for readability by adding the comment text along with the variable:

```
"outputs": [

"quantile_bin(age, 10) //quantile bin age",

"age // explicitly include the original numeric variable along with the
binned version"

]
```

You can mix and match all of these approaches within the outputs section.

> **Note**
>
> Comments are not allowed in the Amazon ML console when adding a recipe.

Complete Recipe Example

The following example refers to several built-in data processors that were introduced in preceding examples:

```
{

"groups": {

"LONGTEXT": "group_remove(ALL_TEXT, title, subject)",

"SPECIALTEXT": "group(title, subject)",

"BINCAT": "group(ALL_CATEGORICAL, ALL_BINARY)"

},
```

```
"assignments": {

"binned_age" : "quantile_bin(age,30)",

"country_gender_interaction" : "cartesian(country, gender)"

},

"outputs": [

"lowercase(no_punct(LONGTEXT))",

"ngram(lowercase(no_punct(SPECIALTEXT)),3)",

"quantile_bin(hours-per-week, 10)",

"hours-per-week // explicitly include the original numeric variable
along with the binned version",

"cartesian(binned_age, quantile_bin(hours-per-week,10)) // this one is
critical",

"country_gender_interaction",

"BINCAT"

]

}
```

Suggested Recipes

When you create a new datasource in Amazon ML and statistics are computed for that datasource, Amazon ML will also create a suggested recipe that can be used to create a new ML model from the datasource. The suggested datasource is based on the data and target attribute present in the data, and provides a useful starting point for creating and fine-tuning your ML models.

To use the suggested recipe on the Amazon ML console, choose **Datasource** or **Datasource and ML model** from the **Create new** drop down list. For ML model settings, you will have a choice of Default or Custom Training and Evaluation settings in the **ML Model Setting**s step of the **Create ML Model** wizard. If you pick the Default option, Amazon ML will automatically use the suggested

recipe. If you pick the Custom option, the recipe editor in the next step will display the suggested recipe, and you will be able to verify or modify it as needed.

> **Note**
>
> Amazon ML allows you to create a datasource and then immediately use it to create an ML model, before statistics computation is completed. In this case, you will not be able to see the suggested recipe in the Custom option, but you will still be able to proceed past that step and have Amazon ML use the default recipe for model training.

To use the suggested recipe with the Amazon ML API, you can pass an empty string in both Recipe and RecipeUri API parameters. It is not possible to retrieve the suggested recipe using the Amazon ML API.

Data Transformations Reference

Topics

N-gram Transformation

The n-gram transformation takes a text variable as input and produces strings corresponding to sliding a window of (user-configurable) n words, generating outputs in the process. For example, consider the text string "I really enjoyed reading this book".

Specifying the n-gram transformation with window size=1 simply gives you all the individual words in that string:

```
{"I", "really", "enjoyed", "reading", "this", "book"}
```

Specifying the n-gram transformation with window size =2 gives you all the two-word combinations as well as the one-word combinations:

```
{"I really", "really enjoyed", "enjoyed reading", "reading this", "this
book", "I", "really", "enjoyed", "reading", "this", "book"}
```

Specifying the n-gram transformation with window size = 3 will add the three-word combinations to this list, yielding the following:

```
{"I really enjoyed", "really enjoyed reading", "enjoyed reading this",
"reading this book", "I really", "really enjoyed", "enjoyed reading",
"reading this", "this book", "I", "really", "enjoyed", "reading",
"this", "book"}
```

You can request n-grams with a size ranging from 2-10 words. N-grams with size 1 are generated implicitly for all inputs whose type is marked as text in the data schema, so you do not have to ask for them. Finally, keep in mind that n-grams are generated by breaking the input data on whitespace characters. That means that, for example, punctuation characters will be considered a part of the word tokens: generating n-grams with a window of 2 for string "red, green, blue" will yield {"red,", "green,", "blue,", "red, green", "green, blue"}. You can use the punctuation remover processor (described later in this document) to remove the punctuation symbols if this is not what you want.

To compute n-grams of window size 3 for variable var1:

```
"ngram(var1, 3)"
```

Orthogonal Sparse Bigram (OSB) Transformation

The OSB transformation is intended to aid in text string analysis and is an alternative to the bi-gram transformation (n-gram with window size 2). OSBs are generated by sliding the window of size n over the text, and outputting every pair of words that includes the first word in the window.

To build each OSB, its constituent words are joined by the "_" (underscore) character, and every skipped token is indicated by adding another underscore into the OSB. Thus, the OSB encodes not

just the tokens seen within a window, but also an indication of number of tokens skipped within that same window.

To illustrate, consider the string "The quick brown fox jumps over the lazy dog", and OSBs of size 4. The six four-word windows, and the last two shorter windows from the end of the string are shown in the following example, as well OSBs generated from each:

Window, {OSBs generated}

```
"The quick brown fox", {The_quick, The__brown, The___fox}

"quick brown fox jumps", {quick_brown, quick__fox, quick___jumps}

"brown fox jumps over", {brown_fox, brown__jumps, brown___over}

"fox jumps over the", {fox_jumps, fox__over, fox___the}

"jumps over the lazy", {jumps_over, jumps__the, jumps___lazy}

"over the lazy dog", {over_the, over__lazy, over___dog}

"the lazy dog", {the_lazy, the__dog}

"lazy dog", {lazy_dog}
```

Orthogonal sparse bigrams are an alternative for n-grams that might work better in some situations. If your data has large text fields (10 or more words), experiment to see which works better. Note that what constitutes a large text field may vary depending on the situation. However, with larger text fields, OSBs have been empirically shown to uniquely represent the text due to the special *skip* symbol (the underscore).

You can request a window size of 2 to 10 for OSB transformations on input text variables.

To compute OSBs with window size 5 for variable var1:

"osb(var1, 5)"

Lowercase Transformation

The lowercase transformation processor converts text inputs to lowercase. For example, given the input "The Quick Brown Fox Jumps Over the Lazy Dog", the processor will output "the quick brown fox jumps over the lazy dog".

To apply lowercase transformation to the variable var1:

"lowercase(var1)"

Remove Punctuation Transformation

Amazon ML implicitly splits inputs marked as text in the data schema on whitespace. Punctuation in the string ends up either adjoining word tokens, or as separate tokens entirely, depending on the whitespace surrounding it. If this is undesirable, the punctuation remover transformation may be used to remove punctuation symbols from generated features. For example, given the string "Welcome to AML - please fasten your seat-belts!", the following set of tokens is implicitly generated:

```
{"Welcome", "to", "Amazon", "ML", "-", "please", "fasten", "your", "seat-belts!"}
```

Applying the punctuation remover processor to this string results in this set:

```
{"Welcome", "to", "Amazon", "ML", "please", "fasten", "your", "seat-belts"}
```

Note that only the prefix and suffix punctuation marks are removed. Punctuations that appear in the middle of a token, e.g. the hyphen in "seat-belts", are not removed.

To apply punctuation removal to the variable var1:

"no_punct(var1)"

Quantile Binning Transformation

The quantile binning processor takes two inputs, a numerical variable and a parameter called *bin number*, and outputs a categorical variable. The purpose is to discover non-linearity in the variable's distribution by grouping observed values together.

In many cases, the relationship between a numeric variable and the target is not linear (the numeric variable value does not increase or decrease monotonically with the target). In such cases,

it might be useful to bin the numeric feature into a categorical feature representing different ranges of the numeric feature. Each categorical feature value (bin) can then be modeled as having its own linear relationship with the target. For example, let's say you know that the continuous numeric feature *account_age* is not linearly correlated with likelihood to purchase a book. You can bin age into categorical features that might be able to capture the relationship with the target more accurately.

The quantile binning processor can be used to instruct Amazon ML to establish n bins of equal size based on the distribution of all input values of the age variable, and then to substitute each number with a text token containing the bin. The optimum number of bins for a numeric variable is dependent on characteristics of the variable and its relationship to the target, and this is best determined through experimentation. Amazon ML suggests the optimal bin number for a numeric feature based on data statistics in the Suggested Recipe.

You can request between 5 and 1000 quantile bins to be computed for any numeric input variable.

To following example shows how to compute and use 50 bins in place of numeric variable var1:

"quantile_bin(var1, 50)"

Normalization Transformation

The normalization transformer normalizes numeric variables to have a mean of zero and variance of one. Normalization of numeric variables can help the learning process if there are very large range differences between numeric variables because variables with the highest magnitude could dominate the ML model, no matter if the feature is informative with respect to the target or not.

To apply this transformation to numeric variable var1, add this to the recipe:

normalize(var1)

This transformer can also take a user defined group of numeric variables or the pre-defined group for all numeric variables (ALL_NUMERIC) as input:

normalize(ALL_NUMERIC)

Note

It is *not* mandatory to use the normalization processor for numeric variables.

Cartesian Product Transformation

The Cartesian transformation generates permutations of two or more text or categorical input variables. This transformation is used when an interaction between variables is suspected. For example, consider the bank marketing dataset that is used in Tutorial: Using Amazon ML to Predict Responses to a Marketing Offer. Using this dataset, we would like to predict whether a person would respond positively to a bank promotion, based on the economic and demographic information. We might suspect that the person's job type is somewhat important (perhaps there is a correlation between being employed in certain fields and having the money available), and the highest level of education attained is also important. We might also have a deeper intuition that there is a strong signal in the interaction of these two variables—for example, that the promotion is particularly well-suited to customers who are entrepreneurs who earned a university degree.

The Cartesian product transformation takes categorical variables or text as input, and produces new features that capture the interaction between these input variables. Specifically, for each training example, it will create a combination of features, and add them as a standalone feature. For example, let's say our simplified input rows look like this:

target, education, job

0, university.degree, technician

0, high.school, services

1, university.degree, admin

If we specify that the Cartesian transformation is to be applied to the categorical variables education and job fields, the resultant feature education_job_interaction will look like this:

target, education_job_interaction

0, university.degree_technician

0, high.school_services

1, university.degree_admin

The Cartesian transformation is even more powerful when it comes to working on sequences of tokens, as is the case when one of its arguments is a text variable that is implicitly or explicitly split into tokens. For example, consider the task of classifying a book as being a textbook or not. Intuitively, we might think that there is something about the book's title that can tell us it is a textbook (certain words might occur more frequently in textbooks' titles), and we might also think

that there is something about the book's binding that is predictive (textbooks are more likely to be hardcover), but it's really the combination of some words in the title and binding that is most predictive. For a real-world example, the following table shows the results of applying the Cartesian processor to the input variables binding and title:

Text	Title	Binding	Cartesian product of no_punct(Title) and Binding
1	Economics : Principles, Problems, Policies	Hardcover	{"Economics_Hardcover", "Principles_Hardcover", "Problems_Hardcover", "Policies_Hardcover"}
0	The Invisible Heart: An Economics Romance	Softcover	{"The_Softcover", "Invisible_Softcover", "Heart_Softcover", "An_Softcover", "Economics_Softcover", "Romance_Softcover"}
0	Fun With Problems	Softcover	{"Fun_Softcover", "With_Softcover", "Problems_Softcover"}

The following example shows how to apply the Cartesian transformer to var1 and var2:

cartesian(var1, var2)

Data Rearrangement

The data rearrangement functionality enables you to create a datasource that is based on only a portion of the input data that it points to. For example, when you create an ML Model using the **Create ML Model** wizard in the Amazon ML console, and choose the default evaluation option, Amazon ML automatically reserves 30% of your data for ML model evaluation, and uses the other 70% for training. This functionality is enabled by the Data Rearrangement feature of Amazon ML.

If you are using the Amazon ML API to create datasources, you can specify which part of the input data a new datasource will be based. You do this by passing instructions in the `DataRearrangement` parameter to the `CreateDataSourceFromS3`, `CreateDataSourceFromRedshift` or `CreateDataSourceFromRDS` APIs. The contents of the DataRearrangement string are a JSON string containing the beginning and end locations of your data, expressed as percentages, a complement flag, and a splitting strategy. For example, the following DataRearrangement string specifies that the first 70% of the data will be used to create the datasource:

```
{
    "splitting": {
        "percentBegin": 0,
        "percentEnd": 70,
        "complement": false,
        "strategy": "sequential"
    }
}
```

DataRearrangement Parameters

To change how Amazon ML creates a datasource, use the follow parameters.

PercentBegin (Optional)

Use `percentBegin` to indicate where the data for the datasource starts. If you do not include `percentBegin` and `percentEnd`, Amazon ML includes all of the data when creating the datasource.

Valid values are `0` to `100`, inclusive.

PercentEnd (Optional)

Use `percentEnd` to indicate where the data for the datasource ends. If you do not include `percentBegin` and `percentEnd`, Amazon ML includes all of the data when creating the datasource.

Valid values are `0` to `100`, inclusive.

Complement (Optional)

The `complement` parameter tells Amazon ML to use the data that is not included in the range of `percentBegin` to `percentEnd` to create a datasource. The `complement` parameter is useful if you need to create complementary datasources for training and evaluation. To create a complementary datasource, use the same values for `percentBegin` and `percentEnd`, along with the `complement` parameter.

For example, the following two datasources do not share any data, and can be used to train and evaluate a model. The first datasource has 25 percent of the data, and the second one has 75 percent of the data.

Datasource for evaluation:

```
{
    "splitting":{
        "percentBegin":0,
        "percentEnd":25
    }
}
```

Datasource for training:

```
{
    "splitting":{
        "percentBegin":0,
        "percentEnd":25,
        "complement":"true"
    }
}
```

Valid values are `true` and `false`.

Strategy (Optional)

To change how Amazon ML splits the data for a datasource, use the `strategy` parameter.

The default value for the `strategy` parameter is `sequential`, meaning that Amazon ML takes all of the data records between the `percentBegin` and `percentEnd` parameters for the datasource, in the order that the records appear in the input data

The following two `DataRearrangement` lines are examples of sequentially ordered training and evaluation datasources:

Datasource for evaluation: `{"splitting":{"percentBegin":70, "percentEnd":100, "strategy":"sequential"}}`

Datasource for training: `{"splitting":{"percentBegin":70, "percentEnd":100, "strategy":"sequential", "complement":"true"}}`

To create a datasource from a random selection of the data, set the `strategy` parameter to `random` and provide a string that is used as the seed value for the random data splitting (for example, you can use the S3 path to your data as the random seed string). If you choose the random split strategy, Amazon ML assigns each row of data a pseudo-random number, and then selects the rows that have an assigned number between `percentBegin` and `percentEnd`.

Pseudo-random numbers are assigned using the byte offset as a seed, so changing the data results in a different split. Any existing ordering is preserved. The random splitting strategy ensures that variables in the training and evaluation data are distributed similarly. It is useful in the cases where the input data may have an implicit sort order, which would otherwise result in training and evaluation datasources containing non-similar data records.

The following two `DataRearrangement` lines are examples of non-sequentially ordered training and evaluation datasources:

Datasource for evaluation:

```
{
    "splitting":{
        "percentBegin":70,
        "percentEnd":100,
        "strategy":"random",
        "strategyParams": {
            "randomSeed":"RANDOMSEED"
        }
    }
}
```

Datasource for training:

```
{
    "splitting":{
        "percentBegin":70,
        "percentEnd":100,
        "strategy":"random",
        "strategyParams": {
            "randomSeed":"RANDOMSEED"
        }
        "complement":"true"
    }
}
```

Valid values are `sequential` and `random`.

(Optional) Strategy:RandomSeed

Amazon ML uses the **randomSeed** to split the data. The default seed for the API is an empty string. To specify a seed for the random split strategy, pass in a string. For more information

about random seeds, see Randomly Splitting Your Data in the *Amazon Machine Learning Developer Guide*.

For sample code that demonstrates how to use cross-validation with Amazon ML, go to Github Machine Learning Samples.

Evaluating ML Models

You should always *evaluate a model* to determine if it will do a good job of predicting the target on new and future data. Because future instances have unknown target values, you need to check the accuracy metric of the ML model on data for which you already know the target answer, and use this assessment as a proxy for predictive accuracy on future data.

To properly evaluate a model, you hold out a sample of data that has been labeled with the target (ground truth) from the training datasource. Evaluating the predictive accuracy of an ML model with the same data that was used for training is not useful, because it rewards models that can "remember" the training data, as opposed to generalizing from it. Once you have finished training the ML model, you send the model the held-out observations for which you know the target values. You then compare the predictions returned by the ML model against the known target value. Finally, you compute a summary metric that tells you how well the predicted and true values match.

In Amazon ML, you evaluate an ML model by *creating an evaluation*. To create an evaluation for an ML model, you need an ML model that you want to evaluate, and you need labeled data that was not used for training. First, create a datasource for evaluation by creating an Amazon ML datasource with the held-out data. The data used in the evaluation must have the same schema as the data used in training and include actual values for the target variable.

If all your data is in a single file or directory, you can use the Amazon ML console to split the data. The default path in the Create ML model wizard splits the input datasource and uses the first 70% for a training datasource and the remaining 30% for an evaluation datasource. You can also customize the split ratio by using the **Custom** option in the Create ML model wizard, where you can choose to select a random 70% sample for training and use the remaining 30% for evaluation. To further specify custom split ratios, use the data rearrangement string in the Create Datasource API. Once you have an evaluation datasource and an ML model, you can create an evaluation and review the results of the evaluation.

Topics

- Preventing Overfitting
- Cross-Validation
- Evaluation Alerts

ML Model Insights

When you evaluate an ML model, Amazon ML provides an industry-standard metric and a number of insights to review the predictive accuracy of your model. In Amazon ML, the outcome of an evaluation contains the following:

- A prediction accuracy metric to report on the overall success of the model
- Visualizations to help explore the accuracy of your model beyond the prediction accuracy metric
- The ability to review the impact of setting a score threshold (only for binary classification)
- Alerts on criteria to check the validity of the evaluation

The choice of the metric and visualization depends on the type of ML model that you are evaluating. It is important to review these visualizations to decide if your model is performing well enough to match your business requirements.

Binary Model Insights

Interpreting the Predictions

The actual output of many binary classification algorithms is a prediction *score*. The score indicates the system's certainty that the given observation belongs to the positive class (the actual target value is 1). Binary classification models in Amazon ML output a score that ranges from 0 to 1. As a consumer of this score, to make the decision about whether the observation should be classified as 1 or 0, you interpret the score by picking a classification threshold, or *cut-off,* and compare the score against it. Any observations with scores higher than the cut-off are predicted as target= 1, and scores lower than the cut-off are predicted as target= 0.

In Amazon ML, the default score cut-off is 0.5. You can choose to update this cut-off to match your business needs. You can use the visualizations in the console to understand how the choice of cut-off will affect your application.

Measuring ML Model Accuracy

Amazon ML provides an industry-standard accuracy metric for binary classification models called Area Under the (Receiver Operating Characteristic) Curve (AUC). AUC measures the ability of the model to predict a higher score for positive examples as compared to negative examples. Because it is independent of the score cut-off, you can get a sense of the prediction accuracy of your model from the AUC metric without picking a threshold.

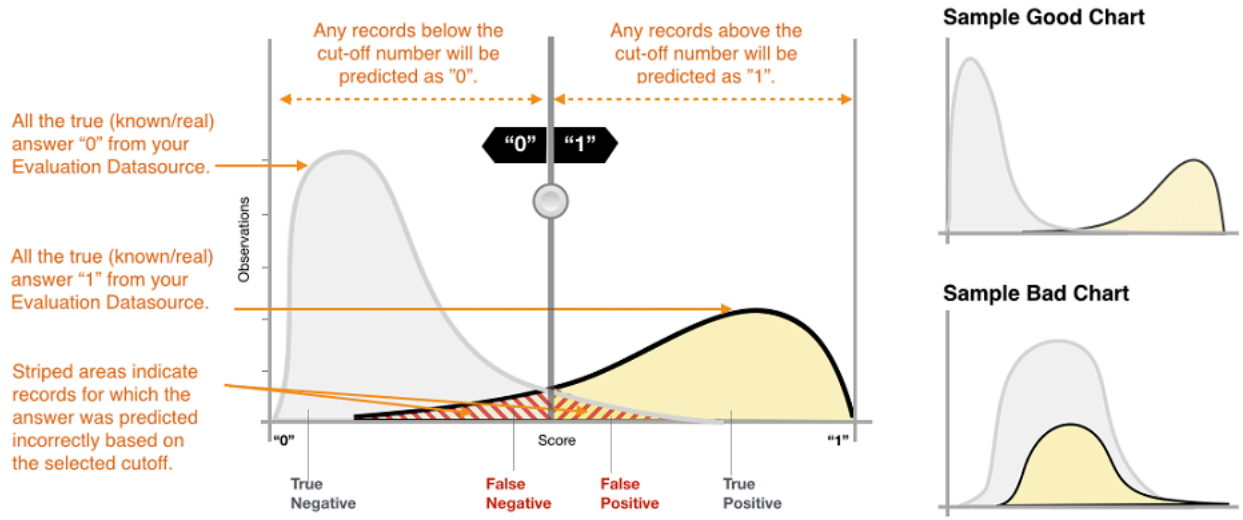
The AUC metric returns a decimal value from 0 to 1. AUC values near 1 indicate an ML model that is highly accurate. Values near 0.5 indicate an ML model that is no better than guessing at random. Values near 0 are unusual to see, and typically indicate a problem with the data. Essentially, an AUC near 0 says that the ML model has learned the correct patterns, but is using them to make predictions that are flipped from reality ('0's are predicted as '1's and vice versa). For more information about AUC, go to the Receiver operating characteristic page on Wikipedia.

The baseline AUC metric for a binary model is 0.5. It is the value for a hypothetical ML model that randomly predicts a 1 or 0 answer. Your binary ML model should perform better than this value to begin to be valuable.

Using the Performance Visualization

To explore the accuracy of the ML model, you can review the graphs on the **Evaluation** page on the Amazon ML console. This page shows you two histograms: a) a histogram of the scores for the actual positives (the target is 1) and b) a histogram of scores for the actual negatives (the target is 0) in the evaluation data.

An ML model that has good predictive accuracy will predict higher scores to the actual 1s and lower scores to the actual 0s. A perfect model will have the two histograms at two different ends of the x-axis showing that actual positives all got high scores and actual negatives all got low scores. However, ML models make mistakes, and a typical graph will show that the two histograms overlap at certain scores. An extremely poor performing model will be unable to distinguish between the positive and negative classes, and both classes will have mostly overlapping histograms.



Using the visualizations, you can identify the number of predictions that fall into the two types of correct predictions and the two types of incorrect predictions.

Correct Predictions

- True positive (TP): Amazon ML predicted the value as 1, and the true value is 1.
- True negative (TN): Amazon ML predicted the value as 0, and the true value is 0.

Erroneous Predictions

- False positive (FP): Amazon ML predicted the value as 1, but the true value is 0.
- False negative (FN): Amazon ML predicted the value as 0, but the true value is 1.

> **Note**
>
> The number of TP, TN, FP, and FN depends on the selected score threshold, and optimizing for any of one of these numbers would mean making a tradeoff on the others. A high number of TPs typically results in a high number of FPs and a low number of TNs.

Adjusting the Score Cut-off

ML models work by generating numeric prediction scores, and then applying a cut-off to convert these scores into binary 0/1 labels. By changing the score cut-off, you can adjust the model's behavior when it makes a mistake. On the **Evaluation** page in the Amazon ML console, you can

review the impact of various score cut-offs, and you can save the score cut-off that you would like to use for your model.

When you adjust the score cut-off threshold, observe the trade-off between the two types of errors. Moving the cut-off to the left captures more true positives, but the trade-off is an increase in the number of false positive errors. Moving it to the right captures less of the false positive errors, but the trade-off is that it will miss some true positives. For your predictive application, you make the decision which kind of error is more tolerable by selecting an appropriate cut-off score.

Reviewing Advanced Metrics

Amazon ML provides the following additional metrics to measure the predictive accuracy of the ML model: accuracy, precision, recall, and false positive rate.

Accuracy

Accuracy (ACC) measures the fraction of correct predictions. The range is 0 to 1. A larger value indicates better predictive accuracy:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Precision

Precision measures the fraction of actual positives among those examples that are predicted as positive. The range is 0 to 1. A larger value indicates better predictive accuracy:

$$Precision = \frac{TP}{TP + FP}$$

Recall

Recall measures the fraction of actual positives that are predicted as positive. The range is 0 to 1. A larger value indicates better predictive accuracy:

$$Recall = \frac{TP}{TP + FN}$$

False Positive Rate

The *false positive rate* (FPR) measures the false alarm rate or the fraction of actual negatives that are predicted as positive. The range is 0 to 1. A smaller value indicates better predictive accuracy:

$$FPR = \frac{FP}{FP + TN}$$

Depending on your business problem, you might be more interested in a model that performs well for a specific subset of these metrics. For example, two business applications might have very different requirements for their ML model:

- One application might need to be extremely sure about the positive predictions actually being positive (high precision), and be able to afford to misclassify some positive examples as negative (moderate recall).
- Another application might need to correctly predict as many positive examples as possible (high recall), and will accept some negative examples being misclassified as positive (moderate precision).

Amazon ML allows you to choose a score cut-off that corresponds to a particular value of any of the preceding advanced metrics. It also shows the tradeoffs incurred with optimizing for any one metric. For example, if you select a cut-off that corresponds to a high precision, you typically will have to trade that off with a lower recall.

> **Note**
>
> You have to save the score cut-off for it to take effect on classifying any future predictions by your ML model.

Multiclass Model Insights

Interpreting the Predictions

The actual output of a multiclass classification algorithm is a set of prediction *scores*. The scores indicate the model's certainty that the given observation belongs to each of the classes. Unlike for binary classification problems, you do not need to choose a score cut-off to make predictions. The predicted answer is the class (for example, label) with the highest predicted score.

Measuring ML Model Accuracy

Typical metrics used in multiclass are the same as the metrics used in the binary classification case after averaging them over all classes. In Amazon ML, the macro-average F1 score is used to evaluate the predictive accuracy of a multiclass metric.

Macro Average F1 Score

F1 score is a binary classification metric that considers both binary metrics precision and recall. It is the harmonic mean between precision and recall. The range is 0 to 1. A larger value indicates better predictive accuracy:

$$F1\ score = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

The macro average F1 score is the unweighted average of the F1-score over all the classes in the multiclass case. It does not take into account the frequency of occurrence of the classes in the evaluation dataset. A larger value indicates better predictive accuracy. The following example shows K classes in the evaluation datasource:

$$Macro\ average\ F1\ score = \frac{1}{K}\sum_{k=1}^{K} F1\ score\ for\ class\ k$$

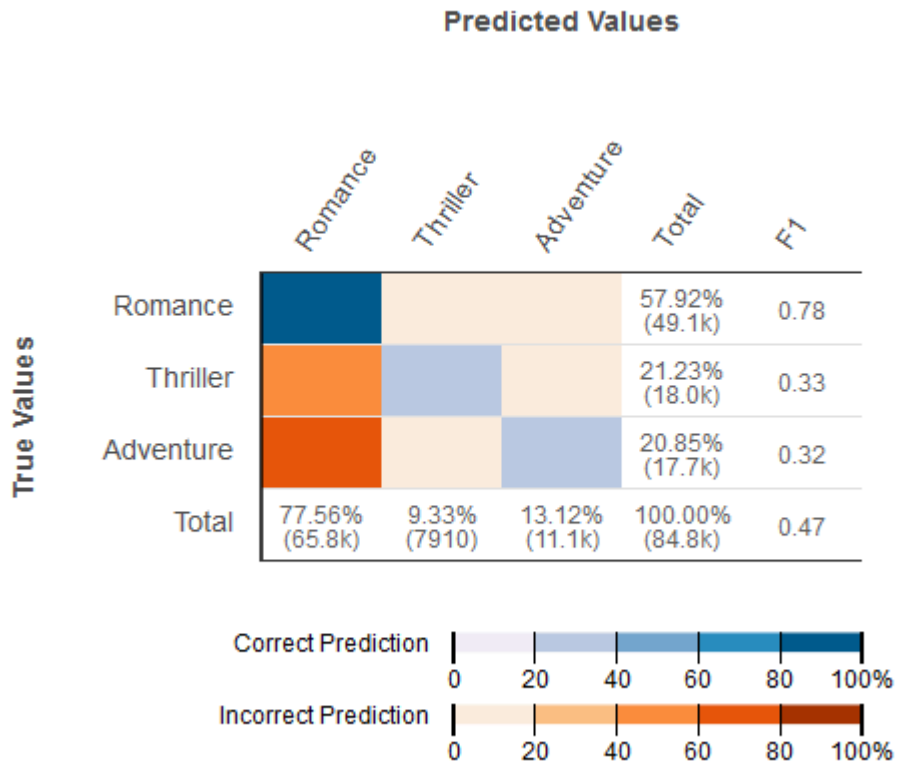
Baseline Macro Average F1 Score

Amazon ML provides a baseline metric for multiclass models. It is the macro average F1 score for a hypothetical multiclass model that would always predict the most frequent class as the answer. For example, if you were predicting the genre of a movie and the most common genre in your training data was Romance, then the baseline model would always predict the genre as Romance. You would compare your ML model against this baseline to validate if your ML model is better than an ML model that predicts this constant answer.

Using the Performance Visualization

Amazon ML provides a *confusion matrix* as a way to visualize the accuracy of multiclass classification predictive models. The confusion matrix illustrates in a table the number or percentage of correct and incorrect predictions for each class by comparing an observation's predicted class and its true class.

For example, if you are trying to classify a movie into a genre, the predictive model might predict that its genre (class) is Romance. However, its true genre actually might be Thriller. When you evaluate the accuracy of a multiclass classification ML model, Amazon ML identifies these misclassifications and displays the results in the confusion matrix, as shown in the following illustration.



The following information is displayed in a confusion matrix:

- **Number of correct and incorrect predictions for each class:** Each row in the confusion matrix corresponds to the metrics for one of the true classes. For example, the first row shows that for movies that are actually in the Romance genre, the multiclass ML model gets the predictions right for over 80% of the cases. It incorrectly predicts the genre as Thriller for less than 20% of the cases, and Adventure for less than 20% of the cases.
- **Class-wise F1-score**: The last column shows the F1-score for each of the classes.
- **True class-frequencies in the evaluation data**: The second to last column shows that in the evaluation dataset, 57.92% of the observations in the evaluation data is Romance, 21.23% is Thriller, and 20.85% is Adventure.
- **Predicted class-frequencies for the evaluation data**: The last row shows the frequency of each class in the predictions. 77.56% of the observations is predicted as Romance, 9.33% is predicted as Thriller, and 13.12% is predicted as Adventure.

The Amazon ML console provides a visual display that accommodates up to 10 classes in the confusion matrix, listed in order of most frequent to least frequent class in the evaluation data. If

your evaluation data has more than 10 classes, you will see the top 9 most frequently occurring classes in the confusion matrix, and all other classes will be collapsed into a class called "others." Amazon ML also provides the ability to download the full confusion matrix through a link on the multiclass visualizations page.

Regression Model Insights

Interpreting the Predictions

The output of a regression ML model is a numeric value for the model's prediction of the target. For example, if you are predicting housing prices, the prediction of the model could be a value such as 254,013.

> **Note**
>
> The range of predictions can differ from the range of the target in the training data. For example, let's say you are predicting housing prices, and the target in the training data had values in a range from 0 to 450,000. The predicted target need not be in the same range, and might take any positive value (greater than 450,000) or negative value (less than zero). It is important to plan how to address prediction values that fall outside a range that is acceptable for your application.

Measuring ML Model Accuracy

For regression tasks, Amazon ML uses the industry standard root mean square error (RMSE) metric. It is a distance measure between the predicted numeric target and the actual numeric answer (ground truth). The smaller the value of the RMSE, the better is the predictive accuracy of the model. A model with perfectly correct predictions would have an RMSE of 0. The following example shows evaluation data that contains N records:

$$RMSE = \sqrt{1/N \sum_{i=1}^{N} (actual\ target - predicted\ target)^2}$$

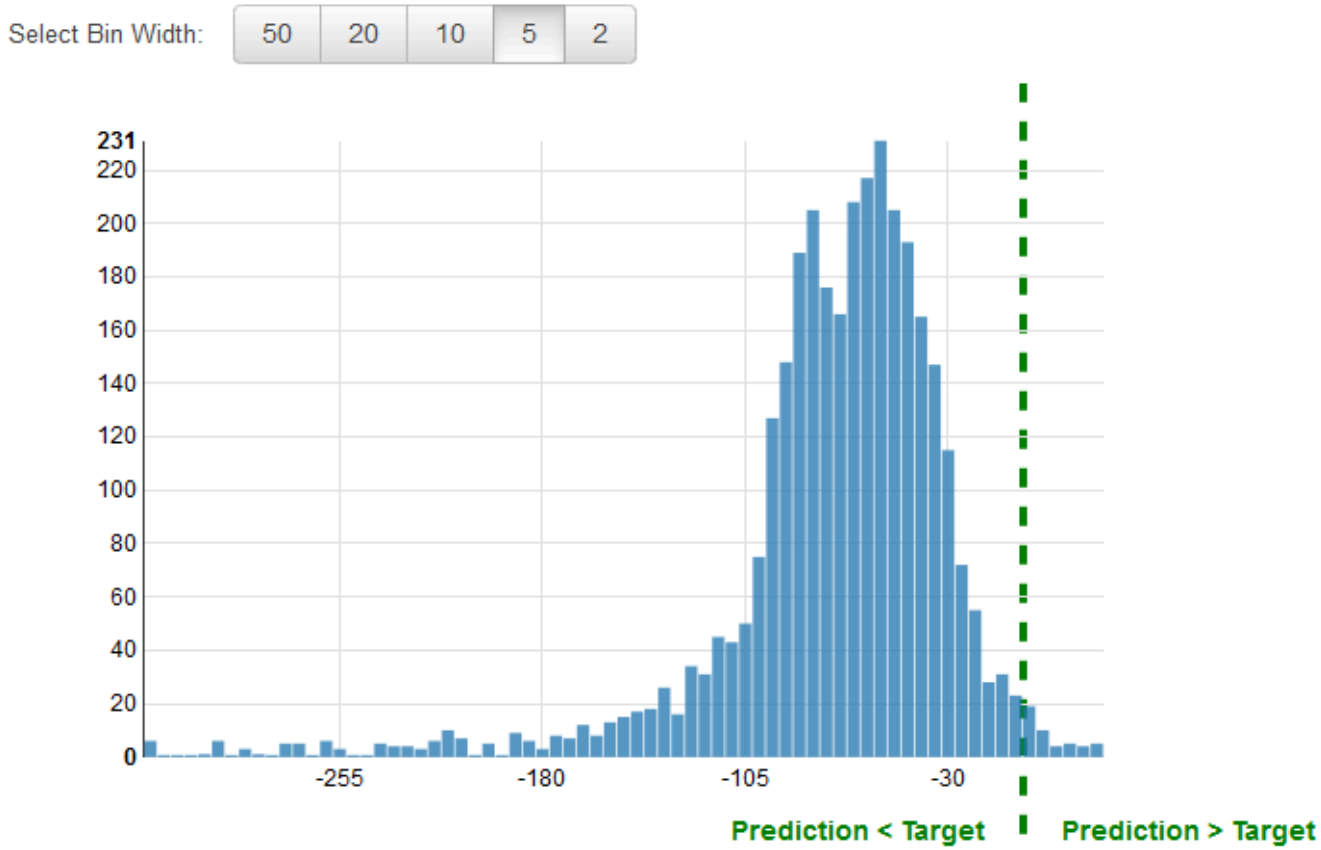
Baseline RMSE

Amazon ML provides a baseline metric for regression models. It is the RMSE for a hypothetical regression model that would always predict the mean of the target as the answer. For example,

if you were predicting the age of a house buyer and the mean age for the observations in your training data was 35, the baseline model would always predict the answer as 35. You would compare your ML model against this baseline to validate if your ML model is better than a ML model that predicts this constant answer.

Using the Performance Visualization

It is common practice to review the *residuals* for regression problems. A residual for an observation in the evaluation data is the difference between the true target and the predicted target. Residuals represent the portion of the target that the model is unable to predict. A positive residual indicates that the model is underestimating the target (the actual target is larger than the predicted target). A negative residual indicates an overestimation (the actual target is smaller than the predicted target). The histogram of the residuals on the evaluation data when distributed in a bell shape and centered at zero indicates that the model makes mistakes in a random manner and does not systematically over or under predict any particular range of target values. If the residuals do not form a zero-centered bell shape, there is some structure in the model's prediction error. Adding more variables to the model might help the model capture the pattern that is not captured by the current model. The following illustration shows residuals that are not centered around zero.



Preventing Overfitting

When creating and training an ML model, the goal is to select the model that makes the best predictions, which means selecting the model with the best settings (ML model settings or hyperparameters). In Amazon Machine Learning, there are four hyperparameters that you can set: number of passes, regularization, model size, and shuffle type. However, if you select model parameter settings that produce the "best" predictive performance on the evaluation data, you might overfit your model. Overfitting occurs when a model has memorized patterns that occur in the training and evaluation datasources, but has failed to generalize the patterns in the data. It often occurs when the training data includes all of the data used in the evaluation. An overfitted model does well during evaluations, but fails to make accurate predictions on unseen data.

To avoid selecting an overfitted model as the best model, you can reserve additional data to validate the performance of the ML model. For example, you can divide your data into 60 percent for training, 20 percent for evaluation, and an additional 20 percent for validation. After selecting the model parameters that work well for the evaluation data, you run a second evaluation with the validation data to see how well the ML model performs on the validation data. If the model meets your expectations on the validation data, then the model is not overfitting the data.

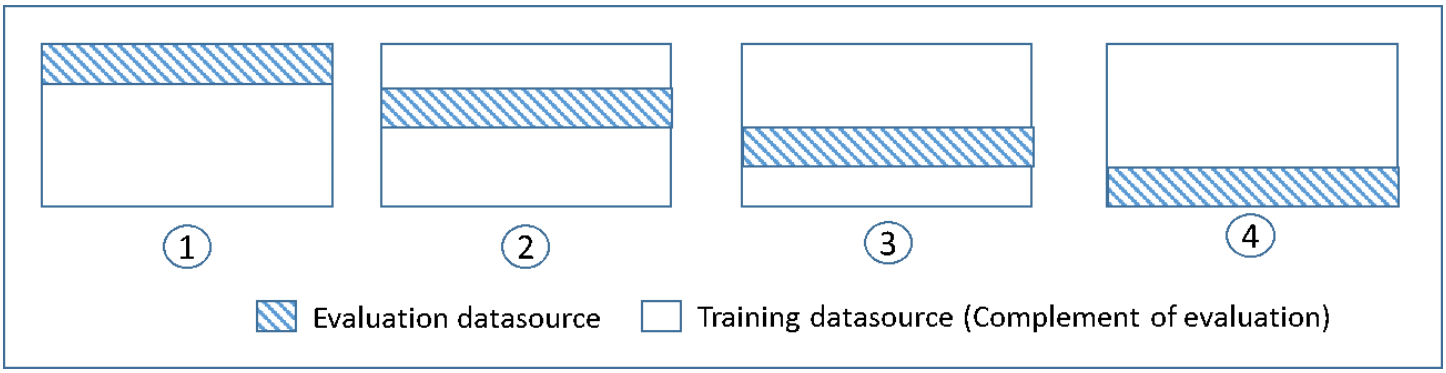
Using a third set of data for validation helps you select appropriate ML model parameters to prevent overfitting. However, holding out data from the training process for both evaluation and validation makes less data available for training. This is especially a problem with small data sets because it's always best to use as much data as possible for training. To solve this problem, you can perform cross-validation. For information on cross-validation, see Cross-Validation.

Cross-Validation

Cross-validation is a technique for evaluating ML models by training several ML models on subsets of the available input data and evaluating them on the complementary subset of the data. Use cross-validation to detect overfitting, ie, failing to generalize a pattern.

In Amazon ML, you can use the k-fold cross-validation method to perform cross-validation. In k-fold cross-validation, you split the input data into k subsets of data (also known as folds). You train an ML model on all but one (k-1) of the subsets, and then evaluate the model on the subset that was not used for training. This process is repeated k times, with a different subset reserved for evaluation (and excluded from training) each time.

The following diagram shows an example of the training subsets and complementary evaluation subsets generated for each of the four models that are created and trained during a 4-fold cross-validation. Model one uses the first 25 percent of data for evaluation, and the remaining 75 percent for training. Model two uses the second subset of 25 percent (25 percent to 50 percent) for evaluation, and the remaining three subsets of the data for training, and so on.



Each model is trained and evaluated using complementary datasources - the data in the evaluation datasource includes and is limited to all of the data that is not in the training datasource. You create datasources for each of these subsets with the `DataRearrangement` parameter in the `createDatasourceFromS3`, `createDatasourceFromRedShift`, and `createDatasourceFromRDS` APIs. In the `DataRearrangement` parameter, specify which subset of data to include in a datasource by specifying where to begin and end each segment. To create the complementary datasources required for a 4k-fold cross validation, specify the `DataRearrangement` parameter as shown in the following example:

Model one:

Datasource for evaluation:

```
{"splitting":{"percentBegin":0, "percentEnd":25}}
```

Datasource for training:

```
{"splitting":{"percentBegin":0, "percentEnd":25, "complement":"true"}}
```

Model two:

Datasource for evaluation:

```
{"splitting":{"percentBegin":25, "percentEnd":50}}
```

Datasource for training:

```
{"splitting":{"percentBegin":25, "percentEnd":50, "complement":"true"}}
```

Model three:

Datasource for evaluation:

```
{"splitting":{"percentBegin":50, "percentEnd":75}}
```

Datasource for training:

```
{"splitting":{"percentBegin":50, "percentEnd":75, "complement":"true"}}
```

Model four:

Datasource for evaluation:

```
{"splitting":{"percentBegin":75, "percentEnd":100}}
```

Datasource for training:

```
{"splitting":{"percentBegin":75, "percentEnd":100, "complement":"true"}}
```

Performing a 4-fold cross-validation generates four models, four datasources to train the models, four datasources to evaluate the models, and four evaluations, one for each model. Amazon ML generates a model performance metric for each evaluation. For example, in a 4-fold cross-validation for a binary classification problem, each of the evaluations reports an area under curve (AUC) metric. You can get the overall performance measure by computing the average of the four AUC metrics. For information about the AUC metric, see Measuring ML Model Accuracy.

For sample code that shows how to create a cross-validation and average the model scores, see the Amazon ML sample code.

Adjusting Your Models

After you have cross-validated the models, you can adjust the settings for the next model if your model does not perform to your standards. For more information about overfitting, see Model Fit:

Underfitting vs. Overfitting. For more information about regularization, see Regularization. For more information on changing the regularization settings, see Creating an ML Model with Custom Options.

Evaluation Alerts

Amazon ML provides insights to help you validate whether you evaluated the model correctly. If any of the validation criteria are not met by the evaluation, the Amazon ML console alerts you by displaying the validation criterion that has been violated, as follows.

- **Evaluation of ML model is done on held-out data**

  Amazon ML alerts you if you use the same datasource for training and evaluation. If you use Amazon ML to split your data, you will meet this validity criterion. If you do not use Amazon ML to split your data, make sure to evaluate your ML model with a datasource other than the training datasource.

- **Sufficient data was used for the evaluation of the predictive model**

  Amazon ML alerts you if the number of observations/records in your evaluation data is less than 10% the number of observations you have in your training datasource. To properly evaluate your model, it is important to provide a sufficiently large data sample. This criterion provides a check to let you know if you are using too little data. The amount of data required to evaluate your ML model is subjective. 10% is selected here as a stop gap in the absence of a better measure.

- **Schema matched**

  Amazon ML alerts you if the schema for the training and evaluation datasource are not the same. If you have certain attributes that do not exist in the evaluation datasource or if you have additional attributes, Amazon ML displays this alert.

- **All records from evaluation files were used for predictive model performance evaluation**

  It is important to know if all the records provided for evaluation were actually used for evaluating the model. Amazon ML alerts you if some records in the evaluation datasource were invalid and were not included in the accuracy metric computation. For example, if the target variable is missing for some of the observations in the evaluation datasource, Amazon ML is unable to check if the ML model's predictions for these observations are correct. In this case, the records with missing target values are considered invalid.

- **Distribution of target variable**

Amazon ML shows you the distribution of the target attribute from the training and evaluation datasources so that you can review whether the target is distributed similarly in both datasources. If the model was trained on training data with a target distribution that differs from the distribution of the target on the evaluation data, then the quality of the evaluation could suffer because it is being computed on data with very different statistics. It is best to have the data distributed similarly over training and evaluation data, and have these datasets mimic as much as possible the data that the model will encounter when making predictions.

If this alert triggers, try using the random split strategy to split the data into training and evaluation datasources. In rare cases, this alert might erroneously warn you about target distribution differences even though you split your data randomly. Amazon ML uses approximate data statistics to evaluate the data distributions, occasionally triggering this alert in error.

Generating and Interpreting Predictions

Amazon ML provides two mechanisms for generating predictions: asynchronous (batch-based) and synchronous (one-at-a-time).

Use asynchronous predictions, or *batch predictions*, when you have a number of observations and would like to obtain predictions for the observations all at once. The process uses a datasource as input, and outputs predictions into a .csv file stored in an S3 bucket of your choice. You need to wait until the batch prediction process completes before you can access the prediction results. The maximum size of a datasource that Amazon ML can process in a batch file is 1 TB (approximately 100 million records). If your datasource is larger than 1 TB, your job will fail and Amazon ML will return an error code. To prevent this, divide your data into multiple batches. If your records are typically longer, you will reach the 1 TB limit before 100 million records are processed. In this case, we recommend that you contact AWS support to increase the job size for your batch prediction.

Use synchronous, or *real-time predictions*, when you want to obtain predictions at low latency. The real-time prediction API accepts a single input observation serialized as a JSON string, and synchronously returns the prediction and associated metadata as part of the API response. You can simultaneously invoke the API more than once to obtain synchronous predictions in parallel. For more information about throughput limits of the real-time prediction API, see real-time prediction limits in Amazon ML API reference.

Topics

Creating a Batch Prediction

To create a batch prediction, you create a `BatchPrediction` object using either the Amazon Machine Learning (Amazon ML) console or API. A `BatchPrediction` object describes a set of predictions that Amazon ML generates by using your ML model and a set of input observations. When you create a `BatchPrediction` object, Amazon ML starts an asynchronous workflow that computes the predictions.

You must use the same schema for the datasource that you use to obtain batch predictions and the datasource that you used to train the ML model that you query for predictions. The one exception is that the datasource for a batch prediction doesn't need to include the target attribute because Amazon ML predicts the target. If you provide the target attribute, Amazon ML ignores its value.

Creating a Batch Prediction (Console)

To create a batch prediction using the Amazon ML console, use the Create Batch Prediction wizard.

To create a batch prediction (console)

1. Sign in to the AWS Management Console and open the Amazon Machine Learning console at https://console.aws.amazon.com/machinelearning/.
2. On the Amazon ML dashboard, under **Objects**, choose **Create new...**, and then choose **Batch prediction**.
3. Choose the Amazon ML model that you want to use to create the batch prediction.
4. To confirm that you want to use this model, choose **Continue**.
5. Choose the datasource that you want to create predictions for. The datasource must have the same schema as your model, although it doesn't need to include the target attribute.
6. Choose **Continue**.
7. For **S3 destination**, type the name of your S3 bucket.
8. Choose **Review**.
9. Review your settings and choose **Create batch prediction**.

Creating a Batch Prediction (API)

To create a `BatchPrediction` object using the Amazon ML API, you must provide the following parameters:

Datasource ID

The ID of the datasource that points to the observations for which you want predictions. For example, if you want predictions for data in a file called `s3://examplebucket/input.csv`, you would create a datasource object that points to the data file, and then pass in the ID of that datasource with this parameter.

BatchPrediction ID

The ID to assign to the batch prediction.

ML Model ID

The ID of the ML model that Amazon ML should query for the predictions.

Output Uri

The URI of the S3 bucket in which to store the output of the prediction. Amazon ML must have permissions to write data to this bucket.

The `OutputUri` parameter must refer to an S3 path that ends with a forward slash ('/') character, as shown in the following example:

s3://examplebucket/examplepath/

For information about configuring S3 permissions , see Granting Amazon ML Permissions to Output Predictions to Amazon S3.

(Optional) BatchPrediction Name

(Optional) A human-readable name for your batch prediction.

Reviewing Batch Prediction Metrics

After Amazon Machine Learning (Amazon ML) creates a batch prediction, it provides two metrics: `Records seen` and `Records failed to process`. `Records seen` tells you how many records Amazon ML looked at when it ran your batch prediction. `Records failed to process` tells you how many records Amazon ML could not process.

To allow Amazon ML to process failed records, check the formatting of the records in the data used to create your datasource, and make sure that all of the required attributes are present and all of the data is correct. After fixing your data, you can either recreate your batch prediction, or create a new datasource with the failed records, and then create a new batch prediction using the new datasource.

Reviewing Batch Prediction Metrics (Console)

To see the metrics in the Amazon ML console, open the **Batch prediction summary** page and look in the **Processed Info** section.

Reviewing Batch Prediction Metrics and Details (API)

You can use the Amazon ML APIs to retrieve details about `BatchPrediction` objects, including the record metrics. Amazon ML provides the following batch prediction API calls:

- `CreateBatchPrediction`
- `UpdateBatchPrediction`
- `DeleteBatchPrediction`
- `GetBatchPrediction`
- `DescribeBatchPredictions`

For more information, see the Amazon ML API Reference.

Reading the Batch Prediction Output Files

Perform the following steps to retrieve the batch prediction output files:

1. Locate the batch prediction manifest file.
2. Read the manifest file to determine the locations of output files.
3. Retrieve the output files that contain the predictions.
4. Interpret the contents of the output files. Contents will vary based on the type of ML model that was used to generate predictions.

The following sections describe the steps in greater detail.

Locating the Batch Prediction Manifest File

The manifest files of the batch prediction contain the information that maps your input files to the prediction output files.

To locate the manifest file, start with the output location that you specified when you created the batch prediction object. You can query a completed batch prediction object to retrieve the S3 location of this file by using either the Amazon ML API or the https://console.aws.amazon.com/machinelearning/.

The manifest file is located in the output location in a path that consists of the static string `/batch-prediction/` appended to the output location and the name of the manifest file, which is the ID of the batch prediction, with the extension `.manifest` appended to that.

For example, if you create a batch prediction object with the ID `bp-example`, and you specify the S3 location `s3://examplebucket/output/` as the output location, you will find your manifest file here:

`s3://examplebucket/output/batch-prediction/bp-example.manifest`

Reading the Manifest File

The contents of the .manifest file are encoded as a JSON map, where the key is a string of the name of an S3 input data file, and the value is a string of the associated batch prediction result file. There is one mapping line for each input/output file pair. Continuing with our example, if the input for the creation of the `BatchPrediction` object consists of a single file called data.csv that is located in `s3://examplebucket/input/`, you might see a mapping string that looks like this:

```
{"s3://examplebucket/input/data.csv":"
s3://examplebucket/output/batch-prediction/result/bp-example-data.csv.gz"}
```

If the input to the creation of the `BatchPrediction` object consists of three files called data1.csv, data2.csv, and data3.csv, and they are all stored in the S3 location `s3://examplebucket/input/`, you might see a mapping string that looks like this:

```
{"s3://examplebucket/input/data1.csv":"s3://examplebucket/output/batch-prediction/
result/bp-example-data1.csv.gz",

"s3://examplebucket/input/data2.csv":"
s3://examplebucket/output/batch-prediction/result/bp-example-data2.csv.gz",

"s3://examplebucket/input/data3.csv":"
s3://examplebucket/output/batch-prediction/result/bp-example-data3.csv.gz"}
```

Retrieving the Batch Prediction Output Files

You can download each batch prediction file obtained from the manifest mapping and process it locally. The file format is CSV, compressed with the gzip algorithm. Within that file, there is one line per input observation in the corresponding input file.

To join the predictions with the input file of the batch prediction, you can perform a simple record-by-record merge of the two files. The output file of the batch prediction always contains the same number of records as the prediction input file, in the same order. If an input observation fails in processing, and no prediction can be generated, the output file of the batch prediction will have a blank line in the corresponding location.

Interpreting the Contents of Batch Prediction Files for a Binary Classification ML model

The columns of the batch prediction file for a binary classification model are named **bestAnswer** and **score**.

The **bestAnswer** column contains the prediction label ("1" or "0") that is obtained by evaluating the prediction score against the cut-off score. For more information about cut-off scores, see Adjusting the Score Cut-off. You set a cut-off score for the ML model by using either the Amazon ML API or the model evaluation functionality on the Amazon ML console. If you don't set a cut-off score,Amazon ML uses the default value of 0.5.

The **score** column contains the raw prediction score assigned by the ML model for this prediction. Amazon ML uses logistic regression models, so this score attempts to model the probability of the observation that corresponds to a true ("1") value. Note that the **score** is reported in scientific notation, so in the first row of the following example, the value 8.7642E-3 is equal to 0.0087642.

For example, if the cut-off score for the ML model is 0.75, the contents of the batch prediction output file for a binary classification model might look like this:

```
bestAnswer,score

0,8.7642E-3

1,7.899012E-1


0,6.323061E-3

0,2.143189E-2


1,8.944209E-1
```

The second and fifth observations in the input file have received prediction scores above 0.75, so the bestAnswer column for these observations indicates value "1", while other observations have the value "0".

Interpreting the Contents of Batch Prediction Files for a Multiclass Classification ML Model

The batch prediction file for a multiclass model contains one column for each class found in the training data. Column names appear in the header line of the batch prediction file.

When you request predictions from a multiclass model, Amazon ML computes several prediction scores for each observation in the input file, one for each of the classes defined in the input dataset. It is equivalent to asking "What is the probability (measured between 0 and 1) that this observation will fall into this class, as opposed to any of the other classes?" Each score can be interpreted as a "probability that the observation belongs to this class." Because prediction scores model the underlying probabilities of the observation belonging to one class or another, the sum of all the prediction scores across a row is 1. You need to pick one class as the predicted class for the model. Most commonly, you would pick the class that has the highest probability as the best answer.

For example, consider attempting to predict a customer's rating of a product, based on a 1-to-5 star scale. If the classes are named `1_star`, `2_stars`, `3_stars`, `4_stars`, and `5_stars`, the multiclass prediction output file might look like this:

```
1_star, 2_stars, 3_stars, 4_stars, 5_stars

8.7642E-3, 2.7195E-1, 4.77781E-1, 1.75411E-1, 6.6094E-2

5.59931E-1, 3.10E-4, 2.48E-4, 1.99871E-1, 2.39640E-1

7.19022E-1, 7.366E-3, 1.95411E-1, 8.78E-4, 7.7323E-2

1.89813E-1, 2.18956E-1, 2.48910E-1, 2.26103E-1, 1.16218E-1

3.129E-3, 8.944209E-1, 3.902E-3, 7.2191E-2, 2.6357E-2
```

In this example, the first observation has the highest prediction score for the `3_stars` class (prediction score = 4.77781E-1), so you would interpret the results as showing that class `3_stars` is the best answer for this observation. Note that prediction scores are reported in scientific notation, so a prediction score of 4.77781E-1 is equal to 0.477781.

There may be circumstances when you do not want to choose the class with the highest probability. For example, you might want to establish a minimum threshold below which you won't consider a class as the best answer even if it has the highest prediction score. Suppose you are classifying movies into genres, and you want the prediction score to be at least 5E-1 before you declare the genre to be your best answer. You get a prediction score of 3E-1 for comedies, 2.5E-1 for dramas, 2.5E-1 for documentaries, and 2E-1 for action movies. In this case, the ML model predicts that comedy is your most likely choice, but you decide not to choose it as the best answer. Because none of the prediction scores exceeded your baseline prediction score of 5E-1, you decide that the prediction is insufficient to confidently predict the genre and you decide to choose something else. Your application might then treat the genre field for this movie as "unknown."

Interpreting the Contents of Batch Prediction Files for a Regression ML Model

The batch prediction file for a regression model contains a single column named **score**. This column contains the raw numeric prediction for each observation in the input data. The values are reported in scientific notation, so the **score** value of -1.526385E1 is equal to -15.26835 in the first row in the following example.

This example shows an output file for a batch prediction performed on a regression model:

```
score

-1.526385E1

-6.188034E0

-1.271108E1

-2.200578E1

8.359159E0
```

Requesting Real-time Predictions

A real-time prediction is a synchronous call to Amazon Machine Learning (Amazon ML). The prediction is made when Amazon ML gets the request, and the response is returned immediately. Real-time predictions are commonly used to enable predictive capabilities within interactive

web, mobile, or desktop applications. You can query an ML model created with Amazon ML for predictions in real time by using the low-latency `Predict` API. The `Predict` operation accepts a single input observation in the request payload, and returns the prediction synchronously in the response. This sets it apart from the batch prediction API, which is invoked with the ID of an Amazon ML datasource object that points to the location of the input observations, and asynchronously returns a URI to a file that contains predictions for all these observations. Amazon ML responds to most real-time prediction requests within 100 milliseconds.

You can try real-time predictions without incurring charges in the Amazon ML console. If you then decide to use real-time predictions, you must first create an endpoint for real-time prediction generation. You can do this in the Amazon ML console or by using the `CreateRealtimeEndpoint` API. After you have an endpoint, use the real-time prediction API to generate real-time predictions.

> **Note**
>
> After you create a real-time endpoint for your model, you will start incurring a capacity reservation charge that is based on the model's size. For more information, see Pricing. If you create the real-time endpoint in the console, the console displays a breakdown of the estimated charges that the endpoint will accrue on an ongoing basis. To stop incurring the charge when you no longer need to obtain real-time predictions from that model, remove the real-time endpoint by using the console or the `DeleteRealtimeEndpoint` operation.

For examples of `Predict` requests and responses, see Predict in the *Amazon Machine Learning API Reference*. To see an example of the exact response format that uses your model, see Trying Real-Time Predictions.

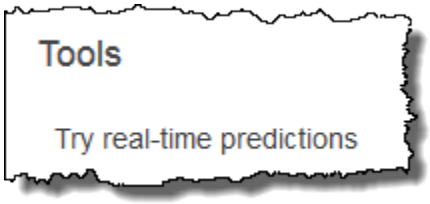
Topics

Trying Real-Time Predictions

To help you decide whether to enable real-time prediction, Amazon ML allows you to try generating predictions on single data records without incurring the additional charges associated with setting up a real-time prediction endpoint. To try real-time prediction, you must have an ML model. To create real-time predictions on a larger scale, use the Predict API in the *Amazon Machine Learning API Reference*.

To try real-time predictions

1. Sign in to the AWS Management Console and open the Amazon Machine Learning console at https://console.aws.amazon.com/machinelearning/.
2. In the navigation bar, in the **Amazon Machine Learning** drop down, choose **ML models**.
3. Choose the model that you want to use to try real-time predictions, such as the `Subscription propensity model` from the tutorial.
4. On the ML model report page, under **Predictions**, choose **Summary**, and then choose **Try real-time predictions**.



   Amazon ML shows a list of the variables that made up the data records that Amazon ML used to train your model.
5. You can proceed by entering data in each of the fields in the form or by pasting a single data record, in CSV format, into the text box.

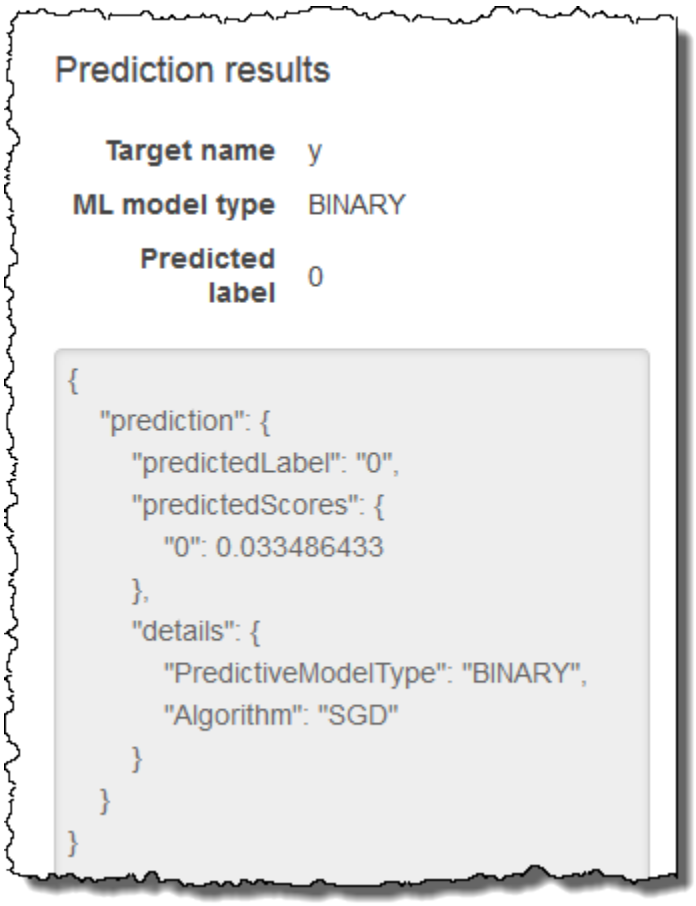
   To use the form, for each **Value** field, enter the data that you want to use to test your real-time predictions. If the data record you are entering does not contain values for one or more data attributes, leave the entry fields blank.

   To provide a data record, choose **Paste a record**. Paste a single CSV-formatted row of data into the text field, and choose **Submit**. Amazon ML auto-populates the **Value** fields for you.

> **Note**
>
> The data in the data record must have the same number of columns as the training data, and be arranged in the same order. The only exception is that you should omit the target value. If you include a target value, Amazon ML ignores it.

6. At the bottom of the page, choose **Create prediction**. Amazon ML returns the prediction immediately.

   In the **Prediction results** pane, you see the prediction object that the `Predict` API call returns, along with the ML model type, the name of the target variable, and the predicted class or value. For information about interpreting the results, see Interpreting the Contents of Batch Prediction Files for a Binary Classification ML model.



Creating a Real-Time Endpoint

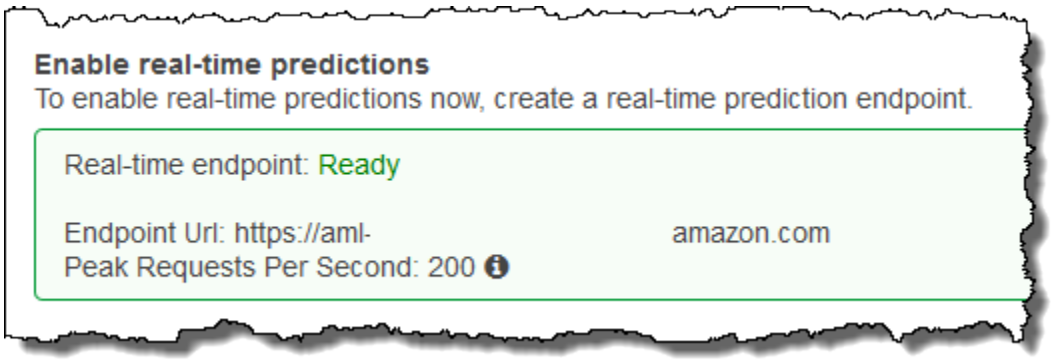
To generate real-time predictions, you need to create a real-time endpoint. To create a real-time endpoint, you must already have an ML model for which you want to generate real-time predictions. You can create a real-time endpoint by using the Amazon ML console or by calling the `CreateRealtimeEndpoint` API. For more information on using the `CreateRealtimeEndpoint` API, see https://docs.aws.amazon.com/machine-learning/latest/APIReference/API_CreateRealtimeEndpoint.html in the Amazon Machine Learning API Reference.

To create a real-time endpoint

1. Sign in to the AWS Management Console and open the Amazon Machine Learning console at https://console.aws.amazon.com/machinelearning/.
2. In the navigation bar, in the **Amazon Machine Learning** drop down, choose **ML models**.
3. Choose the model for which you want to generate real-time predictions.
4. On the **ML model summary** page, under **Predictions**, choose **Create real-time endpoint**.

   A dialog box that explains how real-time predictions are priced appears.
5. Choose **Create**. The real-time endpoint request is sent to Amazon ML and entered into a queue. The status of the real-time endpoint is **Updating**.

6. When the real-time endpoint is ready, the status changes to **Ready**, and Amazon ML displays the endpoint URL. Use the endpoint URL to create real-time prediction requests with the `Predict` API. For more information about using the `Predict` API, see https://docs.aws.amazon.com/machine-learning/latest/APIReference/API_Predict.html in the Amazon Machine Learning API Reference.



Locating the Real-time Prediction Endpoint (Console)

To use the Amazon ML console to find the endpoint URL for an ML model navigate to the model's **ML model summary** page.

To locate a real-time endpoint URL

1. Sign in to the AWS Management Console and open the Amazon Machine Learning console at https://console.aws.amazon.com/machinelearning/.
2. In the navigation bar, in the **Amazon Machine Learning** drop down, choose **ML models**.
3. Choose the model for which you want to generate real-time predictions.
4. On the **ML model summary** page, scroll down until you see the **Predictions** section.
5. The endpoint URL for the model is listed in **Real-time prediction**. Use the URL as the **Endpoint Url** URL for your real-time prediction calls. For information on how to use the endpoint to generate predictions, see https://docs.aws.amazon.com/machine-learning/latest/APIReference/API_Predict.html in the Amazon Machine Learning API Reference.

Locating the Real-time Prediction Endpoint (API)

When you create a real-time endpoint by using the `CreateRealtimeEndpoint` operation, the URL and status of the endpoint is returned to you in the response. If you created the real-time endpoint by using the console or if you want to retrieve the URL and status of an endpoint that you created earlier, call the `GetMLModel` operation with the ID of the model that you want to query for real-time predictions. The endpoint information is contained in the `EndpointInfo` section of the response. For a model that has a real-time endpoint associated with it, the `EndpointInfo` might look like this:

```
"EndpointInfo":{
    "CreatedAt": 1427864874.227,
    "EndpointStatus": "READY",
    "EndpointUrl": "https://endpointUrl",
    "PeakRequestsPerSecond": 200
}
```

A model without a real-time endpoint would return the following:

```
EndpointInfo":{
    "EndpointStatus": "NONE",
    "PeakRequestsPerSecond": 0
}
```

Creating a Real-time Prediction Request

A sample `Predict` request payload might look like this:

```
{
    "MLModelId": "model-id",
    "Record":{
        "key1": "value1",
        "key2": "value2"
    },
    "PredictEndpoint": "https://endpointUrl"
}
```

The `PredictEndpoint` field must correspond to the `EndpointUrl` field of the `EndpointInfo` structure. Amazon ML uses this field to route the request to the appropriate servers in the real-time prediction fleet.

The `MLModelId` is the identifier of a previously trained model with a real-time endpoint.

A `Record` is a map of variable names to variable values. Each pair represents an observation. The `Record` map contains the inputs to your Amazon ML model. It is analogous to a single row of data in your training data set, without the target variable. Regardless of the type of values in the training data, `Record` contains a string-to-string mapping.

> **Note**
>
> You can omit variables for which you do not have a value, although this might reduce the accuracy of your prediction. The more variables you can include, the more accurate your model is.

The format of the response returned by `Predict` requests depends on the type of model that is being queried for prediction. In all cases, the `details` field contains information about the prediction request, notably including the `PredictiveModelType` field with the model type.

The following example shows a response for a binary model:

```
{
    "Prediction":{
        "details":{
            "PredictiveModelType": "BINARY"
        },
        "predictedLabel": "0",
        "predictedScores":{
            "0": 0.47380468249320984
        }
    }
}
```

Notice the `predictedLabel` field that contains the predicted label, in this case 0. Amazon ML computes the predicted label by comparing the prediction score against the classification cut-off:

- You can obtain the classification cut-off that is currently associated with an ML model by inspecting the `ScoreThreshold` field in the response of the `GetMLModel` operation, or by viewing the model information in the Amazon ML console. If you do not set a score threshold, Amazon ML uses the default value of 0.5.
- You can obtain the exact prediction score for a binary classification model by inspecting the `predictedScores` map. Within this map, the predicted label is paired with the exact prediction score.

For more information about binary predictions, see Interpreting the Predictions.

The following example shows a response for a regression model. Notice that the predicted numeric value is found in the `predictedValue` field:

```
{
    "Prediction":{
        "details":{
            "PredictiveModelType": "REGRESSION"
        },
        "predictedValue": 15.508452415466309
    }
}
```

The following example shows a response for a multiclass model:

```
{
    "Prediction":{
        "details":{
            "PredictiveModelType": "MULTICLASS"
        },
        "predictedLabel": "red",
        "predictedScores":{
            "red": 0.12923571467399597,
            "green": 0.08416014909744263,
            "orange": 0.22713537514209747,
            "blue": 0.1438363939523697,
            "pink": 0.184102863073349,
            "violet": 0.12816807627677917,
            "brown": 0.10336143523454666
        }
    }
}
```

Similar to binary classification models, the predicted label/class is found in the `predictedLabel` field. You can further understand how strongly the prediction is related to each class by looking at the `predictedScores` map. The higher the score of a class within this map, the more strongly the prediction is related to the class, with the highest value ultimately being selected as the `predictedLabel`.

For more information about multiclass predictions, see Multiclass Model Insights.

Deleting a Real-Time Endpoint

When you've completed your real-time predictions, delete the real-time endpoint to avoid incurring additional charges. Charges stop accruing as soon as you delete your endpoint.

To delete a real-time endpoint

1. Sign in to the AWS Management Console and open the Amazon Machine Learning console at https://console.aws.amazon.com/machinelearning/.
2. In the navigation bar, in the **Amazon Machine Learning** drop down, choose **ML models**.
3. Choose the model that no longer requires real-time predictions.
4. On the ML model report page, under **Predictions**, choose **Summary**.
5. Choose **Delete real-time endpoint**.
6. In the **Delete real-time endpoint** dialog box, choose **Delete**.

Managing Amazon ML Objects

Amazon ML provides four objects that you can manage through the Amazon ML console or the Amazon ML API:

- Datasources
- ML models
- Evaluations
- Batch predictions

Each object serves a different purpose in the lifecycle of building a machine learning application, and each object has specific attributes and functionality that apply only to that object. Despite these differences, you manage the objects in similar ways. For example, you use almost identical processes for listing objects, retrieving their descriptions, and updating or deleting them.

The following sections describe the management operations that are common to all four objects and notes any differences.

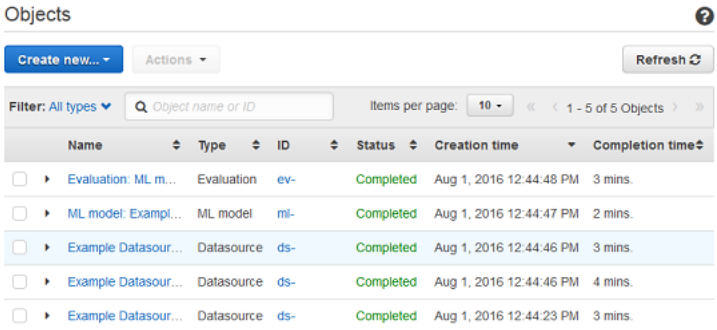
Topics

Listing Objects

For in-depth information about your Amazon Machine Learning (Amazon ML) datasources, ML models, evaluations, and batch predictions, list them. For each object, you will see its name, type, ID, status code, and creation time. You can also see details that are specific to a particular object type. For example, you can see the Data insights for a datasource.

Listing Objects (Console)

To see a list of the last 1,000 objects that you've created, in the Amazon ML console, open the **Objects** dashboard. To display the **Objects** dashboard, log into the Amazon ML console.



To see more details about an object, including details that are specific to that object type, choose the object's name or ID. For example, to see the **Data insights** for a datasource, choose the datasource name.

The columns on the **Objects** dashboard show the following information about each object.

Name

The name of the object.

Type

The type of object. Valid values include **Datasource**, **ML model**, **Evaluation**, and **Batch prediction**.

> **Note**
>
> To see whether a model is set up to support real-time predictions, go to the **ML model summary** page by choosing the name or model ID.

ID

The ID of the object.

Status

The status of the object. Values include **Pending**, **In Progress**, **Completed**, and **Failed**. If the status is **Failed**, check your data and try again.

Creation time

The date and time when Amazon ML finished creating this object.

Completion time

The length of time it took Amazon ML to create this object. You can use the completion time of a model to estimate the training time for a new model.

Datasource ID

For objects that were created using a datasource, such as models and evaluations, the ID of the datasource. If you delete the datasource, you can no longer use ML models created with that datasource to create predictions.

Sort by any column by choosing the double-triangle icon next to the column header.

Listing Objects (API)

In the Amazon ML API, you can list objects, by type, by using the following operations:

- `DescribeDataSources`
- `DescribeMLModels`
- `DescribeEvaluations`
- `DescribeBatchPredictions`

Each operation includes parameters for filtering, sorting, and paginating through a long list of objects. There is no limit to the number of objects that you can access through the API. To limit the size of the list, use the `Limit` parameter, which can take a maximum value of 100.

The API response to a `Describe*` command includes a pagination token (`nextPageToken`), if appropriate, and brief descriptions of each object. The object descriptions include the same information for each of the object types that is displayed in the console, including details that are specific to an object type.

> **Note**
>
> Even if the response includes fewer objects than the specified limit, it might include a `nextPageToken` that indicates that more results are available. Even a response that contains 0 items might contain a `nextPageToken`.

For more information, see the Amazon ML API Reference.

Retrieving Object Descriptions

You can view detailed descriptions of any object through the console or through the API.

Detailed Descriptions in the Console

To see descriptions on the console, navigate to a list for a specific type of object (datasource, ML model, evaluation, or batch prediction). Next, locate the row in the table that corresponds to the object, either by browsing through the list or by searching for its name or ID.

Detailed Descriptions from the API

Each object type has an operation that retrieves the full details of an Amazon ML object:

- GetDataSource
- GetMLModel
- GetEvaluation
- GetBatchPrediction

Each operation takes exactly two parameters: the object ID and a Boolean flag called Verbose. Calls with Verbose set to true will include extra details about the object, resulting in higher latencies and larger responses. To learn which fields are included by setting the Verbose flag, see the Amazon ML API Reference.

Updating Objects

Each object type has an operation that updates the details of an Amazon ML object (See Amazon ML API Reference):

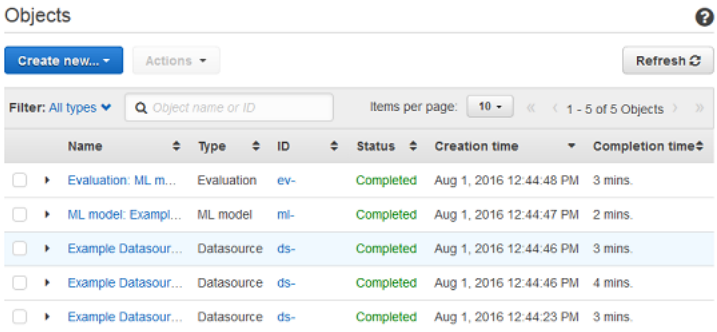
- UpdateDataSource
- UpdateMLModel
- UpdateEvaluation
- UpdateBatchPrediction

Each operation requires the object's ID to specify which object is being updated. You can update the names of all objects. You can't update any other properties of objects for datasources, evaluations, and batch predictions. For ML Models, you can update the ScoreThreshold field, as long as the ML model does not have a real-time prediction endpoint associated with it.

Deleting Objects

When you no longer need your datasources, ML models, evaluations, and batch predictions, you can delete them. Although there is no additional cost for keeping Amazon ML objects other than batch predictions after you are done with them, deleting objects keeps your workspace uncluttered and easier to manage. You can delete single or multiple objects using the Amazon Machine Learning (Amazon ML) console or the API.

> **Warning**
>
> When you delete Amazon ML objects, the effect is immediate, permanent, and irreversible.



Deleting Objects (Console)

You can use the Amazon ML console to delete objects, including models. The procedure that you use to delete a model depends on whether you're using the model to generate real-time predictions or not. To delete a model used to generate real-time predictions, first delete the real-time endpoint.

To delete Amazon ML objects (console)

1. Sign in to the AWS Management Console and open the Amazon Machine Learning console at https://console.aws.amazon.com/machinelearning/.

2. Select the Amazon ML objects that you want to delete. To select more than one object, use the SHIFT key. To deselect all selected objects, use the

   or

   buttons.
3. For **Actions**, choose **Delete**.
4. In the dialog box, choose **Delete** to delete the model.

To delete an Amazon ML model with a real-time endpoint (console)

1. Sign in to the AWS Management Console and open the Amazon Machine Learning console at https://console.aws.amazon.com/machinelearning/.
2. Select the model that you want to delete.
3. For **Actions**, choose **Delete real-time endpoint**.
4. Choose **Delete** to delete the endpoint.
5. Select the model again.
6. For **Actions**, choose **Delete**.
7. Choose **Delete** to delete the model.

Deleting Objects (API)

You can delete Amazon ML objects using the following API calls:

- `DeleteDataSource` - Takes the parameter `DataSourceId`.
- `DeleteMLModel` - Takes the parameter `MLModelId`.
- `DeleteEvaluation` - Takes the parameter `EvaluationId`.
- `DeleteBatchPrediction` - Takes the parameter `BatchPredictionId`.

For more information, see the Amazon Machine Learning API Reference.

Monitoring Amazon ML with Amazon CloudWatch Metrics

Amazon ML automatically sends metrics to Amazon CloudWatch so that you can gather and analyze usage statistics for your ML models. For example, to keep track of batch and real-time predictions, you can monitor the PredictCount metric according to the RequestMode dimension. The metrics are automatically collected and sent to Amazon CloudWatch every five minutes. You can monitor these metrics by using the Amazon CloudWatch console, AWS CLI, or AWS SDKs.

There is no charge for the Amazon ML metrics that are reported through CloudWatch. If you set alarms on the metrics, you will be billed at standard CloudWatch rates.

For more information, see the Amazon ML list of metrics in Amazon CloudWatch Namespaces, Dimensions, and Metrics Reference in the Amazon CloudWatch Developer Guide.

Logging Amazon ML API Calls with AWS CloudTrail

Amazon Machine Learning (Amazon ML) is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon ML. CloudTrail captures all API calls for Amazon ML as events. The calls captured include calls from the Amazon ML console and code calls to the Amazon ML API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon ML. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to Amazon ML, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, including how to configure and enable it, see the AWS CloudTrail User Guide.

Amazon ML Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When supported event activity occurs in Amazon ML, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for Amazon ML, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

Amazon ML supports logging the following actions as events in CloudTrail log files:

- AddTags
- CreateBatchPrediction
- CreateDataSourceFromRDS
- CreateDataSourceFromRedshift
- CreateDataSourceFromS3
- CreateEvaluation
- CreateMLModel
- CreateRealtimeEndpoint
- DeleteBatchPrediction
- DeleteDataSource
- DeleteEvaluation
- DeleteMLModel
- DeleteRealtimeEndpoint
- DeleteTags
- DescribeTags
- UpdateBatchPrediction
- UpdateDataSource
- UpdateEvaluation
- UpdateMLModel

The following Amazon ML operations use request parameters that contain credentials. Before these requests are sent to CloudTrail, the credentials are replaced with three asterisks ("***"):

- CreateDataSourceFromRDS
- CreateDataSourceFromRedshift

When the following Amazon ML operations are performed with the Amazon ML console, the attribute `ComputeStatistics` is not included in the `RequestParameters` component of the CloudTrail log:

- CreateDataSourceFromRedshift
- CreateDataSourceFromS3

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity Element.

Example: Amazon ML Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the action.

```
{
    "Records": [
        {
            "eventVersion": "1.03",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::012345678910:user/Alice",
                "accountId": "012345678910",
                "accessKeyId": "EXAMPLE_KEY_ID",
                "userName": "Alice"
            },
            "eventTime": "2015-11-12T15:04:02Z",
            "eventSource": "machinelearning.amazonaws.com",
            "eventName": "CreateDataSourceFromS3",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "console.amazonaws.com",
            "requestParameters": {
```

```
                "data": {
                    "dataLocationS3": "s3://aml-sample-data/banking-batch.csv",
                    "dataSchema": "{\"version\":\"1.0\",\"rowId\":null,\"rowWeight
\":null,
                        \"targetAttributeName\":null,\"dataFormat\":\"CSV\",
                        \"dataFileContainsHeader\":false,\"attributes\":[
                          {\"attributeName\":\"age\",\"attributeType\":\"NUMERIC\"},
                          {\"attributeName\":\"job\",\"attributeType\":\"CATEGORICAL
\"},
                          {\"attributeName\":\"marital\",\"attributeType\":
\"CATEGORICAL\"},
                          {\"attributeName\":\"education\",\"attributeType\":
\"CATEGORICAL\"},
                          {\"attributeName\":\"default\",\"attributeType\":
\"CATEGORICAL\"},
                          {\"attributeName\":\"housing\",\"attributeType\":
\"CATEGORICAL\"},
                          {\"attributeName\":\"loan\",\"attributeType\":\"CATEGORICAL
\"},
                          {\"attributeName\":\"contact\",\"attributeType\":
\"CATEGORICAL\"},
                          {\"attributeName\":\"month\",\"attributeType\":\"CATEGORICAL
\"},
                          {\"attributeName\":\"day_of_week\",\"attributeType\":
\"CATEGORICAL\"},
                          {\"attributeName\":\"duration\",\"attributeType\":\"NUMERIC
\"},
                          {\"attributeName\":\"campaign\",\"attributeType\":\"NUMERIC
\"},
                          {\"attributeName\":\"pdays\",\"attributeType\":\"NUMERIC\"},
                          {\"attributeName\":\"previous\",\"attributeType\":\"NUMERIC
\"},
                          {\"attributeName\":\"poutcome\",\"attributeType\":
\"CATEGORICAL\"},
                          {\"attributeName\":\"emp_var_rate\",\"attributeType\":
\"NUMERIC\"},
                          {\"attributeName\":\"cons_price_idx\",\"attributeType\":
\"NUMERIC\"},
                          {\"attributeName\":\"cons_conf_idx\",\"attributeType\":
\"NUMERIC\"},
                          {\"attributeName\":\"euribor3m\",\"attributeType\":\"NUMERIC
\"},
                          {\"attributeName\":\"nr_employed\",\"attributeType\":
\"NUMERIC\"}
```

```
                    ],\"excludedAttributeNames\":[]}"
            },
            "dataSourceId": "exampleDataSourceId",
            "dataSourceName": "Banking sample for batch prediction"
        },
        "responseElements": {
            "dataSourceId": "exampleDataSourceId"
        },
        "requestID": "9b14bc94-894e-11e5-a84d-2d2deb28fdec",
        "eventID": "f1d47f93-c708-495b-bff1-cb935a6064b2",
        "eventType": "AwsApiCall",
        "recipientAccountId": "012345678910"
    },
    {
        "eventVersion": "1.03",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::012345678910:user/Alice",
            "accountId": "012345678910",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
        "eventTime": "2015-11-11T15:24:05Z",
        "eventSource": "machinelearning.amazonaws.com",
        "eventName": "CreateBatchPrediction",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "console.amazonaws.com",
        "requestParameters": {
            "batchPredictionName": "Batch prediction: ML model: Banking sample",
            "batchPredictionId": "exampleBatchPredictionId",
            "batchPredictionDataSourceId": "exampleDataSourceId",
            "outputUri": "s3://EXAMPLE_BUCKET/BatchPredictionOutput/",
            "mLModelId": "exampleModelId"
        },
        "responseElements": {
            "batchPredictionId": "exampleBatchPredictionId"
        },
        "requestID": "3e18f252-8888-11e5-b6ca-c9da3c0f3955",
        "eventID": "db27a771-7a2e-4e9d-bfa0-59deee9d936d",
        "eventType": "AwsApiCall",
        "recipientAccountId": "012345678910"
    }
```

```
    ]
}
```

Tagging Your Amazon ML Objects

Organize and manage your Amazon Machine Learning (Amazon ML) objects by assigning metadata to them with tags. A *tag* is a key-value pair that you define for an object.

In addition to using tags to organize and manage your Amazon ML objects, you can use them to categorize and track your AWS costs. When you apply tags to your AWS objects, including ML models, your AWS cost allocation report includes usage and costs aggregated by tags. By applying tags that represent business categories (such as cost centers, application names, or owners), you can organize your costs across multiple services. For more information, see Use Cost Allocation Tags for Custom Billing Reports in the *AWS Billing User Guide*.

Contents

Tag Basics

Use tags to categorize your objects to make it easier to manage them. For example, you could categorize objects by purpose, owner, or environment. Then, you might define a set of tags that helps you track models by owner and associated application. Here are several examples:

- Project: Project name
- Owner: Name
- Purpose: Marketing predictions
- Application: Application name
- Environment: Production

You use the Amazon ML console or API to complete the following tasks:

- Add tags to an object
- View the tags for your objects
- Edit the tags for your objects

- Delete tags from an object

By default, tags applied to an Amazon ML object are copied to objects created using that object. For example, if an Amazon Simple Storage Service (Amazon S3) datasource has the "Marketing cost: Targeted marketing campaign" tag, a model created using that datasource would also have the "Marketing cost: Targeted marketing campaign" tag, as would the evaluation for the model. This allows you to use tags to track related objects, such as all of the objects used for a marketing campaign. If there is a conflict between tag sources, such as a model with the tag "Marketing cost: Targeted marketing campaign" and a datasource with the tag "Marketing cost: Target marketing customers", Amazon ML applies the tag from the model.

Tag Restrictions

The following restrictions apply to tags.

Basic restrictions:

- The maximum number of tags per object is 50.
- Tag keys and values are case sensitive.
- You can't change or edit tags for a deleted object.

Tag key restrictions:

- Each tag key must be unique. If you add a tag with a key that's already in use, your new tag overwrites the existing key-value pair for that object.
- You can't start a tag key with `aws:` because this prefix is reserved for use by AWS. AWS creates tags that begin with this prefix on your behalf, but you can't edit or delete them.
- Tag keys must be between 1 and 128 Unicode characters in length.
- Tag keys must consist of the following characters: Unicode letters, digits, white space, and the following special characters: `_ . / = + - @`.

Tag value restrictions:

- Tag values must be between 0 and 255 Unicode characters in length.
- Tag values can be blank. Otherwise, they must consist of the following characters: Unicode letters, digits, white space, and any of the following special characters: `_ . / = + - @`.

Tagging Amazon ML Objects (Console)

You can view, add, edit, and delete tags using the Amazon ML console.

To view the tags for an object (console)

1. Sign in to the AWS Management Console and open the Amazon Machine Learning console at https://console.aws.amazon.com/machinelearning/.
2. In the navigation bar, expand the region selector and choose a region.
3. On the **Objects** page, choose an object.
4. Scroll to the **Tags** section of the chosen object. The tags for that object are listed at the bottom of the section.

To add a tag to an object (console)

1. Sign in to the AWS Management Console and open the Amazon Machine Learning console at https://console.aws.amazon.com/machinelearning/.
2. In the navigation bar, expand the region selector and choose a region.
3. On the **Objects** page, choose an object.
4. Scroll to the **Tags** section of the chosen object. The tags for that object are listed at the bottom of the section.
5. Choose **Add or edit tags**.
6. Under **Add Tag**, specify the tag key in the **Key** field, optionally specify a tag value in the **Value** field, and then choose **Apply changes**.

   If the **Apply changes** button isn't enabled, either the tag key or tag value that you specified doesn't meet the tag restrictions. For more information, see Tag Restrictions.
7. To view your new tag in the list in the **Tags** section, refresh the page.

To edit a tag (console)

1. Sign in to the AWS Management Console and open the Amazon Machine Learning console at https://console.aws.amazon.com/machinelearning/.
2. In the navigation bar, expand the region selector and select a region.
3. On the **Objects** page, choose an object.

4. Scroll to the **Tags** section of the chosen object. The tags for that object are listed at the bottom of the section.
5. Choose **Add or edit tags**.
6. Under **Applied tags**, edit the a tag value in the **Value** field, and then choose **Apply changes**.

   If the **Apply changes** button is not enabled, the tag value that you specified doesn't meet the tag restrictions. For more information, see Tag Restrictions.
7. To view your updated tag in the list in the **Tags** section, refresh the page.

To delete a tag from an object (console)

1. Sign in to the AWS Management Console and open the Amazon Machine Learning console at https://console.aws.amazon.com/machinelearning/.
2. In the navigation bar, expand the region selector and choose a region.
3. On the **Objects** page, choose an object.
4. Scroll to the **Tags** section of the chosen object. The tags for that object are listed at the bottom of the section.
5. Choose **Add or edit tags**.
6. Under **Applied tags**, choose the tag that you want to delete, and then choose **Apply changes**.

Tagging Amazon ML Objects (API)

You can add, list, and delete tags using the Amazon ML API. For examples, see the following documentation:

AddTags

Adds or edits tags for the specified object.

DescribeTags

Lists the tags for the specified object.

DeleteTags

Deletes tags from the specified object.

Amazon Machine Learning Reference

Topics

Granting Amazon ML Permissions to Read Your Data from Amazon S3

To create a datasource object from your input data in Amazon S3, you must grant Amazon ML the following permissions to the S3 location where your input data is stored:

- **GetObject** permission on the S3 bucket and prefix.
- **ListBucket** permission on the S3 bucket. Unlike other actions, **ListBucket** must be granted bucket-wide permissions (rather than on the prefix). However, you can scope the permission to a specific prefix by using a **Condition** clause.

If you use the Amazon ML console to create the datasource, these permissions can be added to the bucket for you. You will be prompted to confirm whether you want to add them as you complete the steps in the wizard.The following example policy shows how to grant permission for Amazon ML to read data from the sample location s3://*examplebucket*/*exampleprefix*, while scoping the **ListBucket** permission to only the *exampleprefix* input path.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "machinelearning.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::examplebucket/exampleprefix/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:machinelearning:us-
east-1:123456789012:*"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "machinelearning.amazonaws.com"
            },
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::examplebucket",
            "Condition": {
                "StringLike": {
                    "s3:prefix": "exampleprefix/*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:machinelearning:us-
east-1:123456789012:*"
                }
            }
        }
    ]
}
```

To apply this policy to your data, you must edit the policy statement associated with the S3 bucket where your data is stored.

To edit the permissions policy for an S3 bucket (using the old console)

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Select the bucket name where your data resides.
3. Choose **Properties**.
4. Choose **Edit bucket policy**
5. Enter the policy shown above, customizing it to fit your needs, and then choose **Save**.
6. Choose **Save**.

To edit the permissions policy for an S3 bucket (using the new console)

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the bucket name and then choose **Permissions**.
3. Choose **Bucket Policy**.
4. Enter the policy shown above, customizing it to fit your needs.
5. Choose **Save**.

Granting Amazon ML Permissions to Output Predictions to Amazon S3

To output the results of the batch prediction operation to Amazon S3, you must grant Amazon ML the following permissions to the output location, which is provided as input to the Create Batch Prediction operation:

- **GetObject** permission on your S3 bucket and prefix.
- **PutObject** permission on your S3 bucket and prefix.
- **PutObjectAcl** on your S3 bucket and prefix.
  - Amazon ML needs this permission to ensure that it can grant the canned ACL bucket-owner-full-control permission to your AWS account, after objects are created.

- **ListBucket** permission on the S3 bucket. Unlike other actions, **ListBucket** must be granted bucket-wide permissions (rather than on the prefix). You can, however, scope the permission to a specific prefix by using a **Condition** clause.

If you use the Amazon ML console to create the batch prediction request, these permissions can be added to the bucket for you. You will be prompted to confirm whether you want to add them as you complete the steps in the wizard.

The following example policy shows how to grant permission for Amazon ML to write data to the sample location s3://examplebucket/exampleprefix, while scoping the **ListBucket** permission to only the exampleprefix input path, and granting the permission for Amazon ML to set put object ACLs on the output prefix:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "machinelearning.amazonaws.com"
            },
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::examplebucket/exampleprefix/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:machinelearning:us-
east-1:123456789012:*"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
```

```
                "Service": "machinelearning.amazonaws.com"
            },
            "Action": "s3:PutObjectAcl",
            "Resource": "arn:aws:s3:::examplebucket/exampleprefix/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:machinelearning:us-
east-1:123456789012:*"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "machinelearning.amazonaws.com"
            },
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::examplebucket",
            "Condition": {
                "StringLike": {
                    "s3:prefix": "exampleprefix/*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:machinelearning:us-
east-1:123456789012:*"
                }
            }
        }
    ]
}
```

To apply this policy to your data, you must edit the policy statement associated with the S3 bucket where your data is stored.

To edit the permissions policy for an S3 bucket (using the old console)

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Select the bucket name where your data resides.
3. Choose **Properties**.
4. Choose **Edit bucket policy**
5. Enter the policy shown above, customizing it to fit your needs, and then choose **Save**.
6. Choose **Save**.

To edit the permissions policy for an S3 bucket (using the new console)

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the bucket name and then choose **Permissions**.
3. Choose **Bucket Policy**.
4. Enter the policy shown above, customizing it to fit your needs.
5. Choose **Save**.

Controlling Access to Amazon ML Resources -with IAM

AWS Identity and Access Management (IAM) enables you to securely control access to AWS services and resources for your users. Using IAM, you can create and manage AWS users, groups, and roles, and use permissions to allow and deny their access to AWS resources.By using IAM with Amazon Machine Learning (Amazon ML), you can control whether users in your organization can use specific AWS resources and whether they can perform a task using specific Amazon ML API actions.

IAM enables you to:

- Create users and groups under your AWS account.
- Assign unique security credentials to each user under your AWS account
- Control each user's permissions to perform tasks using AWS resources
- Easily share your AWS resources with the users in your AWS account
- Create roles for your AWS account and manage permissions to them to define the users or services that can assume them

- You can create roles in IAM and manage permissions to control which operations can be performed by the entity, or AWS service, that assumes the role. You can also define which entity is allowed to assume the role.

If your organization already has IAM identities, you can use them to grant permissions to perform tasks using AWS resources.

For more information about IAM, see the IAM User Guide.

IAM Policy Syntax

An IAM policy is a JSON document that consists of one or more statements. Each statement has the following structure:

```
{
    "Statement":[{
        "Effect":"effect",
        "Action":"action",
        "Resource":"arn",
        "Condition":{
            "condition operator":{
                "key":"value"
            }
        }
    }]
}
```

A policy statement includes the following elements:

- **Effect:** Controls permission to use the resources and API actions that you will specify later in the statement. Valid values are `Allow` and `Deny`. By default, IAM users don't have permission to use resources and API actions, so all requests are denied. An explicit `Allow` overrides the default. An explicit `Deny` overrides any `Allows`.
- **Action**: The specific API action or actions for which you are granting or denying permission.
- **Resource**: The resource that's affected by the action. To specify a resource in the statement, you use its Amazon Resource Name (ARN).
- **Condition (optional)**: Controls when your policy will be in effect.

To simplify creating and managing IAM policies, you can use the AWS Policy Generator and the IAM Policy Simulator.

Specifying IAM Policy Actions for Amazon MLAmazon ML

In an IAM policy statement, you can specify an API action for any service that supports IAM. When you create a policy statement for Amazon ML API actions, prepend `machinelearning:` to the name of the API action, as shown in the following examples:

- `machinelearning:CreateDataSourceFromS3`
- `machinelearning:DescribeDataSources`
- `machinelearning:DeleteDataSource`
- `machinelearning:GetDataSource`

To specify multiple actions in a single statement, separate them with commas:

```
"Action": ["machinelearning:action1", "machinelearning:action2"]
```

You can also specify multiple actions using wildcards. For example, you can specify all actions whose name begins with the word "Get":

```
"Action": "machinelearning:Get*"
```

To specify all Amazon ML actions, use the * wildcard:

```
"Action": "machinelearning:*"
```

For the complete list of Amazon ML API actions, see the Amazon Machine Learning API Reference.

Specifying ARNs for Amazon ML Resources in IAM Policies

IAM policy statements apply to one or more resources. You specify resources for your policies by their ARNs.

To specify the ARNs for Amazon ML resources, use the following format:

"Resource": `arn:aws:machinelearning:region:account:resource-type/identifier`

The following examples show how to specify common ARNs.

Datasource ID: `my-s3-datasource-id`

```
"Resource":
arn:aws:machinelearning:<region>:<your-account-id>:datasource/my-s3-datasource-id
```

ML model ID: `my-ml-model-id`

```
"Resource":
arn:aws:machinelearning:<region>:<your-account-id>:mlmodel/my-ml-model-id
```

Batch prediction ID: `my-batchprediction-id`

```
"Resource":
arn:aws:machinelearning:<region>:<your-account-id>:batchprediction/my-batchprediction-
id
```

Evaluation ID: `my-evaluation-id`

```
"Resource": arn:aws:machinelearning:<region>:<your-account-id>:evaluation/my-
evaluation-id
```

Example Policies for Amazon MLs

Example 1: Allow users to read machine learning resources metadata

The following policy allows a user or group read the metadata of datasources, ML models, batch predictions, and evaluations by performing DescribeDataSources, DescribeMLModels, DescribeBatchPredictions, DescribeEvaluations, GetDataSource, GetMLModel, GetBatchPrediction, and GetEvaluation actions on the specified resource(s). The Describe * operations permissions can't be restricted to a particular resource.

JSON

```
{ "Version":"2012-10-17",       "Statement": [ { "Effect": "Allow", "Action": [
    "machinelearning:Get*" ], "Resource": [
        "arn:aws:machinelearning:us-east-1:123456789012:datasource/S3-DS-ID1",
```

```
        "arn:aws:machinelearning:us-east-1:123456789012:datasource/REDSHIFT-DS-
ID1",
        "arn:aws:machinelearning:us-east-1:123456789012:mlmodel/ML-MODEL-ID1",
        "arn:aws:machinelearning:us-east-1:123456789012:batchprediction/BP-ID1",
        "arn:aws:machinelearning:us-east-1:123456789012:evaluation/EV-ID1"
    ] }, { "Effect": "Allow", "Action": [ "machinelearning:Describe*" ],
 "Resource": [ "*" ] } ]
    }
```

Example 2: Allow users to create machine learning resources

The following policy allows a user or group to create machine learning datasources, ML models, batch predictions, and evaluations by performing `CreateDataSourceFromS3`, `CreateDataSourceFromRedshift`, `CreateDataSourceFromRDS`, `CreateMLModel`, `CreateBatchPrediction`, and `CreateEvaluation` actions. You can't restrict the permissions for these actions to a specific resource.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "machinelearning:CreateDataSourceFrom*",
                "machinelearning:CreateMLModel",
                "machinelearning:CreateBatchPrediction",
                "machinelearning:CreateEvaluation"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Example 3: Allow users to create and delete) real-time endpoints and perform real-time predictions on an ML model

The following policy allows users or groups to create and delete real-time endpoints and perform real-time predictions for a specific ML model by performing `CreateRealtimeEndpoint`, `DeleteRealtimeEndpoint`, and `Predict` actions on that model.

JSON

```
{ "Version":"2012-10-17",        "Statement": [ { "Effect": "Allow", "Action": [
    "machinelearning:CreateRealtimeEndpoint",
 "machinelearning:DeleteRealtimeEndpoint",
    "machinelearning:Predict" ], "Resource": [
        "arn:aws:machinelearning:us-east-1:123456789012:mlmodel/ML-MODEL"
    ] } ] }
```

Example 4: Allow users to update and delete specific resources

The following policy allows a user or group to update and delete specific resources in your AWS account by giving them permission to perform `UpdateDataSource`, `UpdateMLModel`, `UpdateBatchPrediction`, `UpdateEvaluation`, `DeleteDataSource`, `DeleteMLModel`, `DeleteBatchPrediction`, and `DeleteEvaluation` actions on those resources in your account.

JSON

```
{ "Version":"2012-10-17",        "Statement": [ { "Effect": "Allow", "Action": [
    "machinelearning:Update*", "machinelearning:DeleteDataSource",
 "machinelearning:DeleteMLModel",
    "machinelearning:DeleteBatchPrediction",
 "machinelearning:DeleteEvaluation" ], "Resource": [
        "arn:aws:machinelearning:us-east-1:123456789012:datasource/S3-DS-ID1",
        "arn:aws:machinelearning:us-east-1:123456789012:datasource/REDSHIFT-DS-
ID1",
        "arn:aws:machinelearning:us-east-1:123456789012:mlmodel/ML-MODEL-ID1",
        "arn:aws:machinelearning:us-east-1:123456789012:batchprediction/BP-ID1",
        "arn:aws:machinelearning:us-east-1:123456789012:evaluation/EV-ID1"
    ] } ] }
```

Example 5: Allow any Amazon MLaction

The following policy allows a user or group to use any Amazon ML action. Because this policy grants full access to all of your machine learning resources, restrict it to administrators only.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "machinelearning:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in resource policies to limit the permissions that Amazon Machine Learning gives another service to the resource. If the `aws:SourceArn` value does not contain the account ID, such as an Amazon S3 bucket ARN, you must use both global condition context keys to limit permissions. If you use both global condition context keys and the `aws:SourceArn` value contains the account ID, the `aws:SourceAccount` value and the account in the `aws:SourceArn` value must use the same account ID when used in the same policy statement. Use `aws:SourceArn` if you want only

one resource to be associated with the cross-service access. Use `aws:SourceAccount` if you want to allow any resource in that account to be associated with the cross-service use.

The most effective way to protect against the confused deputy problem is to use the `aws:SourceArn` global condition context key with the full ARN of the resource. If you don't know the full ARN of the resource or if you are specifying multiple resources, use the `aws:SourceArn` global context condition key with wildcards (*) for the unknown portions of the ARN. For example, `arn:aws:servicename:*:123456789012:*`.

The following example shows how you can use the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in Amazon ML to prevent the confused deputy problem when reading data from an Amazon S3 bucket.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "machinelearning.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::examplebucket/exampleprefix/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:machinelearning:us-
east-1:123456789012:*"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "machinelearning.amazonaws.com"
            },
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::examplebucket",
```

```
            "Condition": {
                "StringLike": {
                    "s3:prefix": "exampleprefix/*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:machinelearning:us-
east-1:123456789012:*"
                }
            }
        }
    ]
}
```

Dependency Management of Asynchronous Operations

Batch operations in Amazon ML depend on other operations in order to complete successfully. To manage these dependencies, Amazon ML identifies requests that have dependencies, and verifies that the operations have completed. If the operations have not completed, Amazon ML sets the initial requests aside until the operations that they depend on have completed.

There are some dependencies between batch operations. For example, before you can create an ML model, you must have created a datasource with which you can train the ML model. Amazon ML cannot train an ML model if there is no datasource available.

However, Amazon ML supports dependency management for asynchronous operations. For example, you do not have to wait until data statistics have been computed before you can send a request to train an ML model on the datasource. Instead, as soon as the datasource is created, you can send a request to train an ML model using the datasource. Amazon ML does not actually start the training operation until the datasource statistics have been computed. The createMLModel request is put into a queue until the statistics have been computed; once that is done, Amazon ML immediately attempts to run the createMLModel operation. Similarly, you can send batch prediction and evaluation requests for ML models that have not finished training.

The following table shows the requirements to proceed with different AmazonML actions

In order to...	You must have...
Create an ML model (createMLModel)	Datasource with computed data statistics
Create a batch prediction (createBatchPrediction)	Datasource ML model
Create a batch evaluation (createBatchEvaluation)	Datasource ML model

Checking Request Status

When you submit a request, you can check its status with the Amazon Machine Learning (Amazon ML) API. For example, if you submit a `createMLModel` request, you can check its status by using the `describeMLModel` call. Amazon ML responds with one of the following statuses.

Status	Definition
PENDING	Amazon ML is validating the request. OR Amazon ML is waiting for computational resources to become available before running the request. This might occur when your account has exceeded the maximum number of concurren t running batch operation requests. If this is the case, the status transitions to **InProgress** when other running requests have completed or get canceled. OR Amazon ML is waiting for a batch operation that your request depends on to complete.
INPROGRESS	Your request is still running.

Status	Definition
COMPLETED	The request has finished, and the object is ready to be used (ML models and datasources) or viewed (batch predictions and evaluations).
FAILED	There is something wrong with the data that you provided, or you have canceled the operation. For example, if you try to compute data statistics on a datasource that failed to complete, you might receive an **Invalid** or **Failed** status message. The error message explains why the operation did not complete successfully.
DELETED	The object has already been deleted.

Amazon ML also provides information about an object, such as when Amazon ML finished creating that object. For more information, see Listing Objects.

System Limits

In order to provide a robust, reliable service, Amazon ML imposes certain limits on the requests you make to the system. Most ML problems fit easily within these constraints. However, if you do find that your use of Amazon ML is being restricted by these limits, you can contact AWS customer service and request to have a limit raised. For example, you might have a limit of five for the number of jobs that you can run simultaneously. If you find that you often have jobs queued that are waiting for resources because of this limit, then it probably makes sense to raise that limit for your account.

The following table shows default per-account limits in Amazon ML. Not all of these limits can be raised by AWS customer service.

Limit Type	System Limit
Size of each observations	100 KB
Size of training data *	100 GB
Size of batch prediction input	1 TB

Limit Type	System Limit
Size of batch prediction input (number of records)	100 million
Number of variables in a data file (schema)	1,000
Recipe complexity (number of processed output variables)	10,000
TPS for each real-time prediction endpoint	200
Total TPS for all real-time prediction endpoints	10,000
Total RAM for all real-time prediction endpoints	10 GB
Number of simultaneous jobs	25
Longest run time for any job	7 days
Number of classes for multiclass ML models	100
ML model size	Minimum of 1 MB, maximum of 2 GB
Number of tags per object	50

- The size of your data files is limited to ensure that jobs finish in a timely manner. Jobs that have been running for more than seven days will be automatically terminated, resulting in a FAILED status.

Names and IDs for all Objects

Every object in Amazon ML must have an identifier, or ID. The Amazon ML console generates ID values for you, but if you use the API you must generate your own. Each ID must be unique among all Amazon ML objects of the same type in your AWS account. That is, you cannot have two evaluations with the same ID. It is possible to have an evaluation and a datasource with the same ID, although it is not recommended.

We recommend that you use randomly generated identifiers for your objects, prefixed with a short string to identify their type. For example, when the Amazon ML console generates a datasource, it

assigns the datasource a random, unique ID like "ds-zScWIuWiOxF". This ID is sufficiently random to avoid collisions for any single user, and it's also compact and readable. The "ds-" prefix is for convenience and clarity, but is not required. If you're not sure what to use for your ID strings, we recommend using hexadecimal UUID values (like 28b1e915-57e5-4e6c-a7bd-6fb4e729cb23), which are readily available in any modern programming environment.

ID strings can contain ASCII letters, numbers, hyphens and underscores, and can be up to 64 characters long. It is possible and perhaps convenient to encode metadata into an ID string. But it is not recommended because once an object has been created, its ID cannot be changed.

Object names provide an easy way for you to associate user-friendly metadata with each object. You can update names after an object has been created. This makes it possible for the object's name to reflect some aspect of your ML workflow. For example, you might initially name an ML model "experiment #3", and then later rename the model "final production model". Names can be any string you want, up to 1,024 characters.

Object Lifetimes

Any datasource, ML model, evaluation, or batch prediction object that you create with Amazon ML will be available for your use for at least two years after creation. Amazon ML might automatically remove objects that have not been accessed or used for over two years.

Resources

The following related resources can help you as you work with this service.

- **Amazon ML product information** – Captures all the pertinent product information about Amazon ML in a central location.
- **Amazon ML FAQs** – Covers the top questions that developers have asked about this product.
- **Amazon ML sample code** – Sample applications that use Amazon ML. You can use the sample code as a starting point to create your own ML applications.
- **Amazon ML API Reference** – Describes all of the API operations for Amazon ML in detail. It also provides sample requests and responses for supported web service protocols.
- **AWS Developer Resource Center** – Provides a central starting point to find documentation, code samples, release notes, and other information to help you build innovative applications with AWS.
- **AWS Training and Courses** – Links to role-based and specialty courses and self-paced labs to help sharpen your AWS skills and gain practical experience.
- **AWS Developer Tools** – Links to developer tools and resources that provide documentation, code samples, release notes, and other information to help you build innovative applications with AWS.
- **AWS Support Center** – The hub for creating and managing your AWS support cases. Also includes links to other helpful resources, such as forums, technical FAQs, service health status, and AWS Trusted Advisor.
- **AWS Support** – The primary web page for information about AWS Support, a one-on-one, fast-response support channel to help you build and run applications in the cloud.
- **Contact Us** – A central contact point for inquiries concerning AWS billing, your account, events, abuse, and other issues.
- **AWS Site Terms** – Detailed information about our copyright and trademark; your account, license, and site access; and other topics.

Document History

The following table describes the important changes to the documentation in this release of Amazon Machine Learning (Amazon ML).

- **API Version:** 2015-04-09
- **Last documentation update:** 2016-08-02

Change	Description	Date Changed
Metrics added	This release of Amazon ML adds new metrics for Amazon ML objects. For more information, see Listing Objects.	August 2nd, 2016
Delete multiple objects	This release of Amazon ML adds the ability to delete multiple Amazon ML objects. For more information, see Deleting Objects.	July 20th, 2016
Tagging added	This release of Amazon ML adds the ability to apply tags to Amazon ML objects. For more information, see Tagging Your Amazon ML Objects.	June 23rd, 2016
Copying Amazon Redshift datasources	This release of Amazon ML adds the ability copy Amazon Redshift datasource settings to a new Amazon Redshift datasource. For more information about copying Amazon Redshift datasource settings, see Copying a Datasource (Console).	April 11th, 2016
Shuffling added	This release of Amazon ML adds the ability to shuffle your input data. For more information about using the **Shuffle type** parameter, see Shuffle Type for Training Data.	April 5th, 2016

Change	Description	Date Changed
Improved datasource creation with Amazon Redshift	This release of Amazon ML adds the ability to test your Amazon Redshift settings when you create an Amazon ML datasource in the console to verify that the connection works. For more information, see Creating a Datasource with Amazon Redshift Data (Console).	March 21st, 2016
Improved Amazon Redshift data schema conversion	This release of Amazon ML improves the conversion of Amazon Redshift (Amazon Redshift) data schemas to Amazon ML data schemas. For more information about using Amazon Redshift with Amazon ML, see Creating an Amazon ML Datasource from Data in Amazon Redshift.	February 9th, 2016
CloudTrail logging added	This release of Amazon ML adds the ability to log requests using AWS CloudTrail (CloudTrail). For more information about using CloudTrail logging, see Logging Amazon ML API Calls with AWS CloudTrail.	December 10th, 2015
Additional DataRearrangement options added	This release of Amazon ML adds the ability to split your input data randomly and create complementary datasources. For more information about using the `DataRearrangement` parameter, see Data Rearrangement . For information on how to use the new options for cross-validation, see Cross-Validation.	December 3rd, 2015
Trying real-time predictions	This release of Amazon ML adds ability to try real-time predictions in the service console. For more information about trying real-time predictions, see Requesting Real-time Predictions in the *Amazon Machine Learning Developer Guide*.	November 19th, 2015

Change	Description	Date Changed
New Region	This release of Amazon ML adds support for the EU (Ireland) region. For more information about Amazon ML in the EU (Ireland) region, see Regions and Endpoints in the *Amazon Machine Learning Developer Guide*.	August 20th, 2015
Initial Release	This is the first release of the *Amazon ML Developer Guide*.	April 9th, 2015