Academy-CUR-TF-200-ACMLFO-1-PROD (EN)

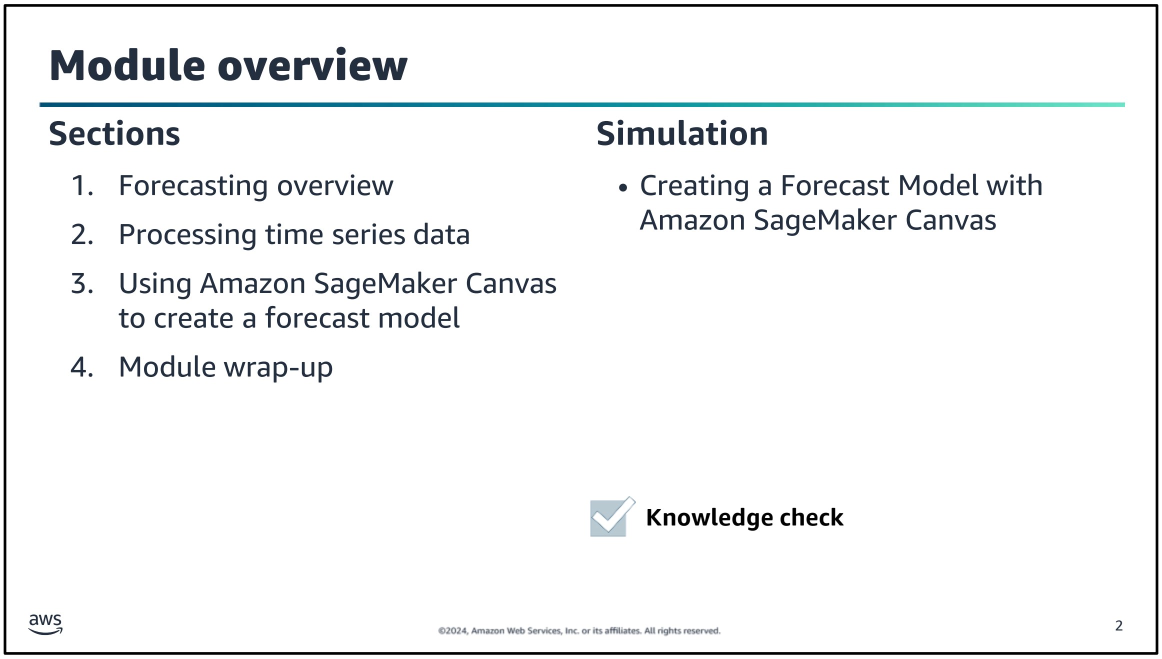
Module 04 Student Guide

Version 1.0.7

200-ACMLFO-10-EN-SG

Contents

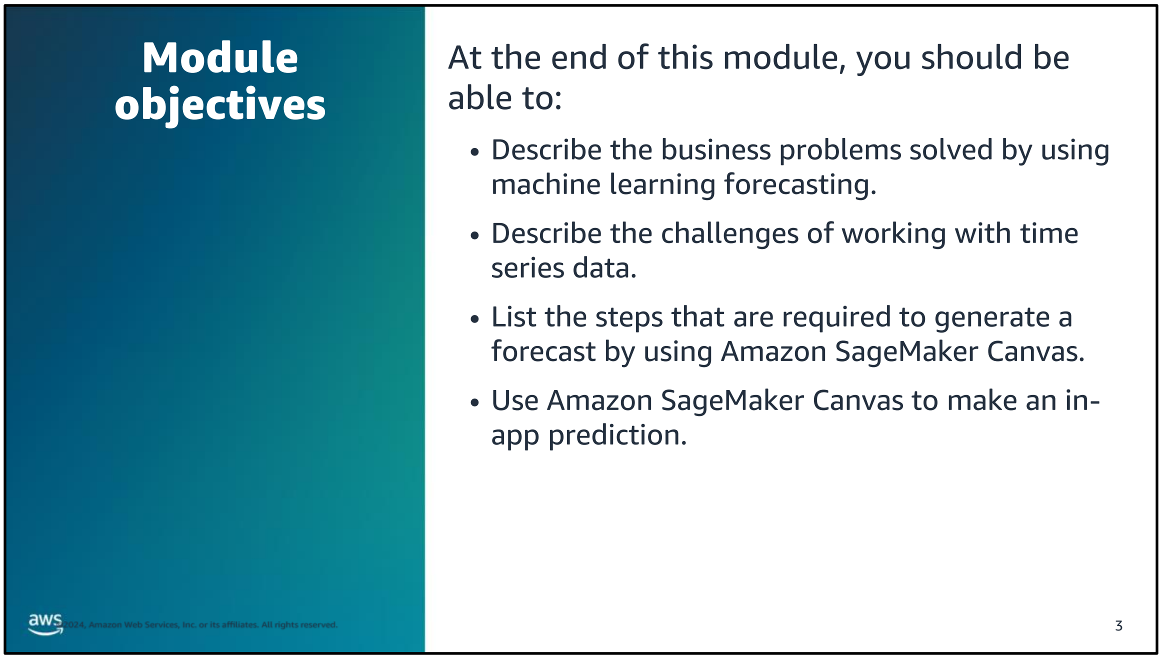
Welcome to Module 4: Introduction to Forecasting.



This module includes the following sections:
- Forecasting overview
- Processing time series data
- Using Amazon SageMaker Canvas to create a forecast model
- Module wrap-up

The module also includes a simulation where you will learn how to use Amazon SageMaker Canvas to work with time series data.

Finally, you will be asked to complete a knowledge check that will test your understanding of key concepts covered in this module.



After completing this module, you should be able to do the following:

- Describe the business problems solved by using machine learning forecasting.
- Describe the challenges of working with time series data.
- List the steps that are required to generate a forecast by using Amazon SageMaker Canvas.
- Use Amazon SageMaker Canvas to make an in-app prediction.

Introducing Section 1: Forecasting overview.

You start with a review of what forecasting means and learn about some use cases for forecasting.

Overview of forecasting

- Predicting future values that are based on historical data
  - Can be either univariate or multivariate
- Common patterns –
  - Trends: Patterns that increase, decrease, or are stagnant
  - Seasonal: Pattern that is based on seasons
  - Cyclical: Other repeating patterns
  - Irregular: Patterns that might appear to be random


Trending data


Seasonal data


Cyclical data


Irregular data

5

Forecasting is an important area of machine learning. It is important because so many opportunities for predicting future outcomes are based on historical data. Many of these opportunities involve a time component. Although the time component adds more information, it also makes time series problems more difficult to handle than other types of predictions.

You can think of time series data as falling into two broad categories. The first type is *univariate*, which means that it has only one variable. The second type is *multivariate*, which means that it has more than one variable. In addition to these two categories, most time series datasets also follow one of the following patterns:

- Trend – A pattern that shows the values as they increase, decrease, or stay the same over time
- Seasonal – A repeating pattern that is based on the seasons in a year
- Cyclical – Some other form of a repeating pattern
- Irregular – Changes in the data over time that appear to be random or that have no discernable pattern

Forecasting use cases

Sales and demand forecasts

Inventory projections

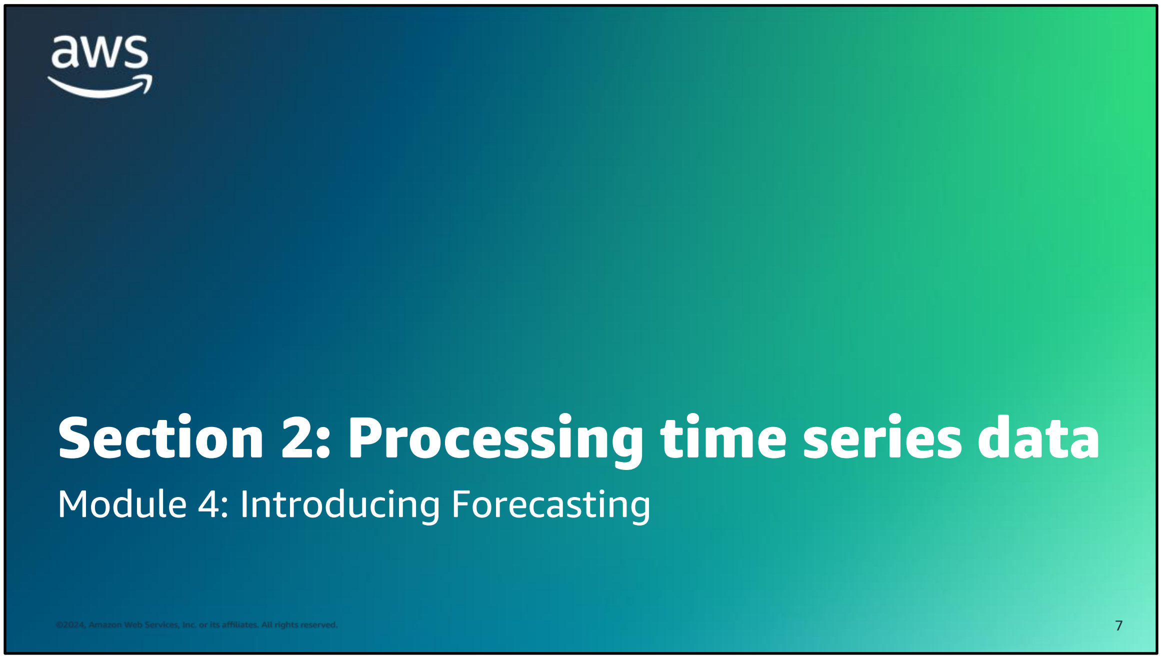
Energy consumption

Weather forecasts

 6

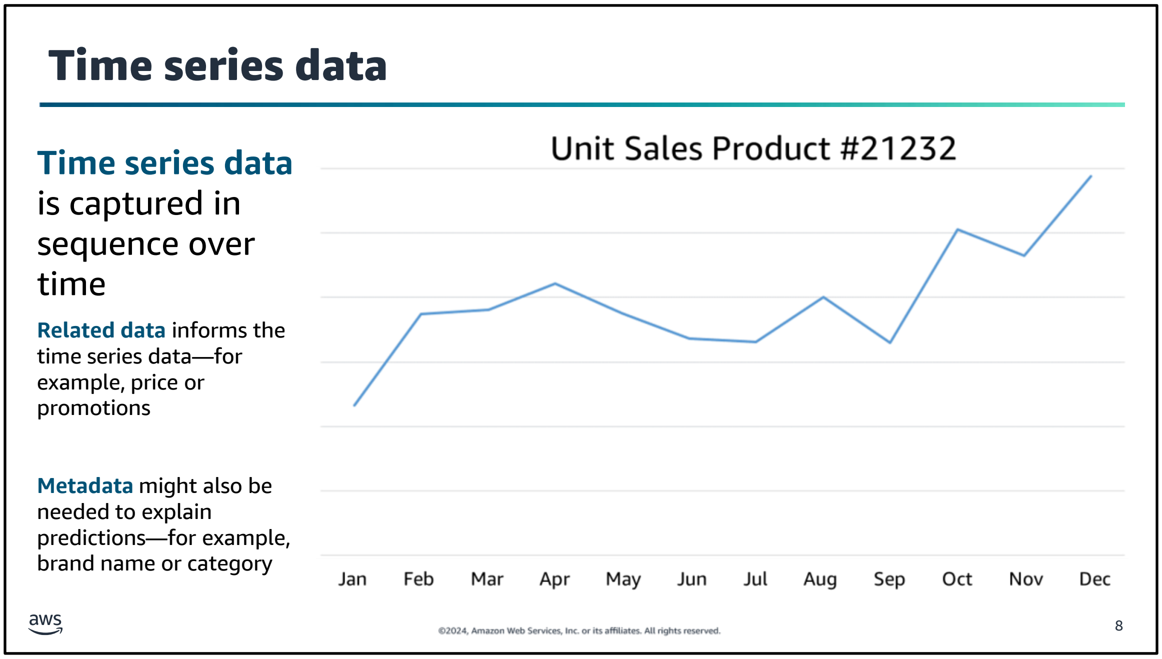
You can use forecasting for a range of domains. Some of the more common applications include:

- Marketing applications, such as sales forecasting or demand projections.
- Inventory management systems to anticipate required inventory levels. Often, this type of forecast includes information about delivery times.
- Energy consumption to determine when and where energy is needed.
- Weather forecasting systems for governments, and commercial applications such as agriculture.



Introducing Section 2: Processing time series data.

Working with time series data presents several unique challenges, which you will now review.

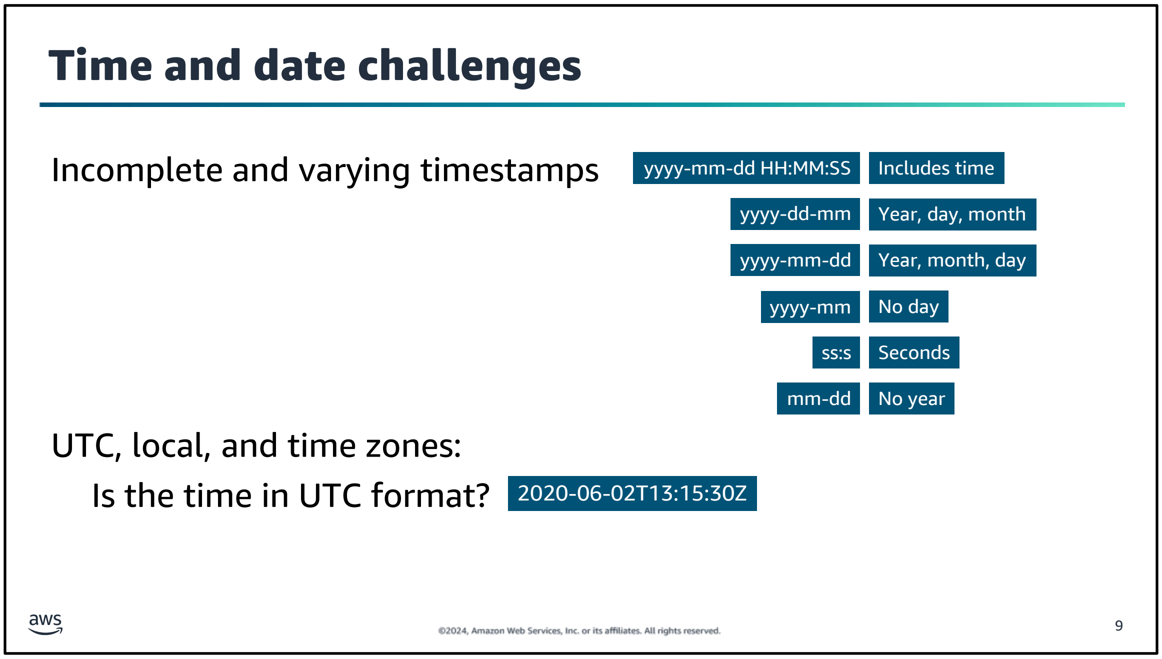


Time series data is captured in chronological sequence over a defined period of time. Introducing time into a machine learning model has a positive impact because the model can derive meaning from change in the data points over time. Time series data tends to be correlated, which means that a dependency exists between data points.

Because you have a regression problem—and because regression assumes independence of data points—you must develop a method for handling data dependence. The purpose of this method is to increase the validity of the predictions.

In addition to the time series data, you can add related data to augment a forecasting model. For example, for a prediction about retail sales, you might include information about the product being sold (such as item identification or sales price). This information is in addition to the number of units that are sold per time period.

The third type of data is metadata about the dataset. For example, for a retail dataset, you might want to include metadata to group results, like a brand name. Another example of metadata could be including a genre for music or videos.



The more data that you have, the better. A challenge that you can expect with multiple data sources is the timestamp of the data. You find differences in the timestamp format, along with other challenges, such as incomplete data. You might be able to infer missing data in some cases. For example, imagine that you have some data that contains both the month and the day, but no year. Suppose that the data appears to sequence through the month numbers in the database, and repeats after 12. In that case, you can add the year if you know when the data started. You can infer future years, based on the order of the data.

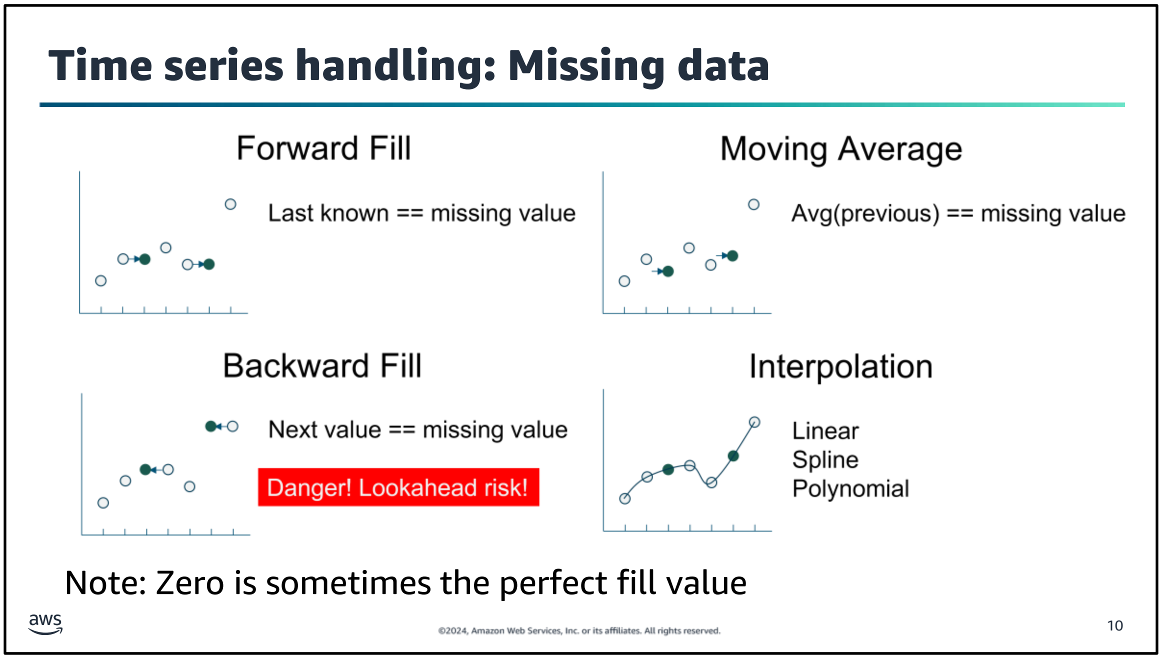
Much data is stored in Universal Coordinated Time (UTC) format, but not all data is in UTC. You should check whether the timestamp is local or universal time. With ML in general, it is good practice to standardize timestamps to UTC for consistency across time zones.

Sometimes the timestamp doesn’t represent the time that you think it does. For example, suppose you have a database of cars that were serviced at a garage. Does the timestamp indicate the time that the car arrived, was completed, or was picked up? Or does it indicate when the final entry was entered into the system?

If you try to model hourly caloric intake of patients, but you have only daily data, then you must adjust your target timescale.

You might not have a timestamp present in your data. You might have other ways to extrapolate a time series, depending on the data and domain. For example, you might have wavelength measurements or vectors in an image.

Daylight savings time is different around the world. Because of daylight savings, some times occur twice a year in their time zones.



A common occurrence in real-world forecasting problems is missing values in the raw data. Missing values makes it harder for a model to generate a forecast. The primary example in retail is an out-of-stock situation in demand forecasting. If an item goes out of stock, the sales for the day will zero. If the forecast is generated based on those zero sales values, the forecast will be incorrect.

Missing values can be marked as missing for various reasons. Missing values can occur because of no transaction, or possibly because of measurement errors. Maybe a service that monitored certain data was not working correctly, or the measurement could not occur correctly.

The missing data can be calculated in several ways:

- Forward fill – Uses the last known value for the missing value.
- Moving average – Uses the average of the last known values to calculate the missing value.
- Backward fill – Uses the next known value after the missing value. Be aware that it is a potential danger to use the future to calculate the past, which is bad in forecasting. This practice is known as *lookahead*, and it should be avoided.
- Interpolation – Essentially uses an equation to calculate the missing value.

You also have the choice to use a zero fill. This choice is often used in retail, a domain where missing sales data shouldn't be calculated. The missing data represents no orders on that day. It would be wise to investigate why, but you don't want to fill in the missing value in this case.

Time series handling: Downsampling

yyyy-mm-dd HH:MM:SS

InvoiceDate	Item	Quantity
2009-12-01 07:45:00	21232	24
2009-12-01 10:06:00	21232	36
2010-12-08 14:21:00	21232	17
2010-12-08 13:11:00	21232	9
2010-12-09 18:28:00	21232	44

→ Mean/ Sum →

yyyy-mm-dd

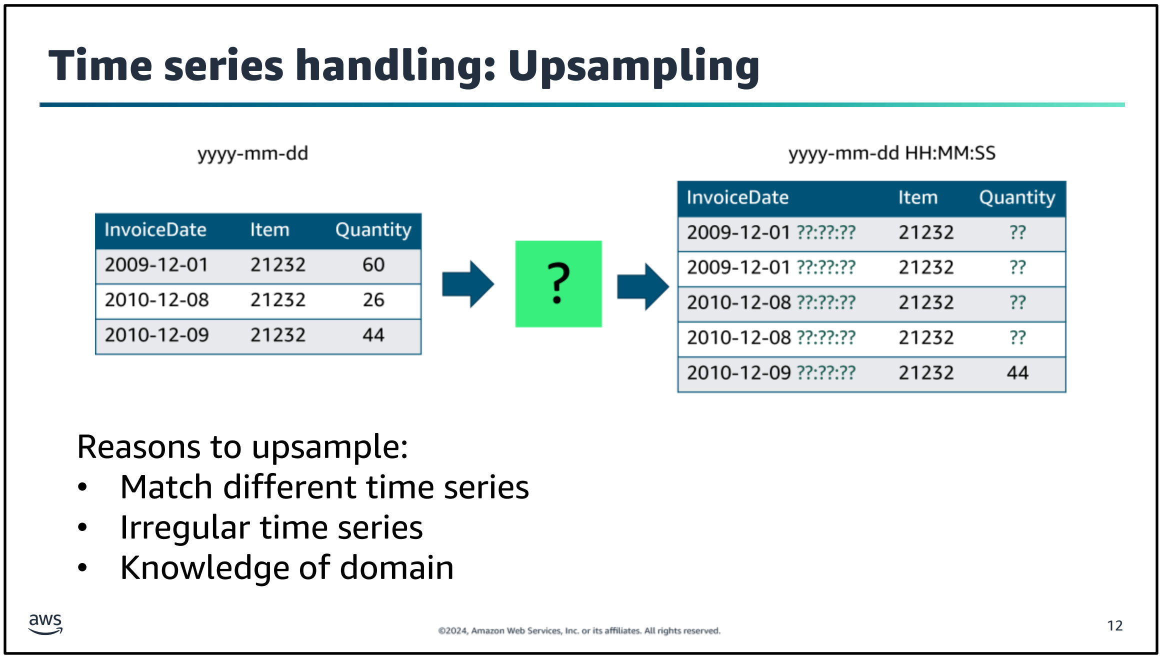
InvoiceDate	Item	Quantity
2009-12-01	21232	60
2010-12-08	21232	26
2010-12-09	21232	44

 11

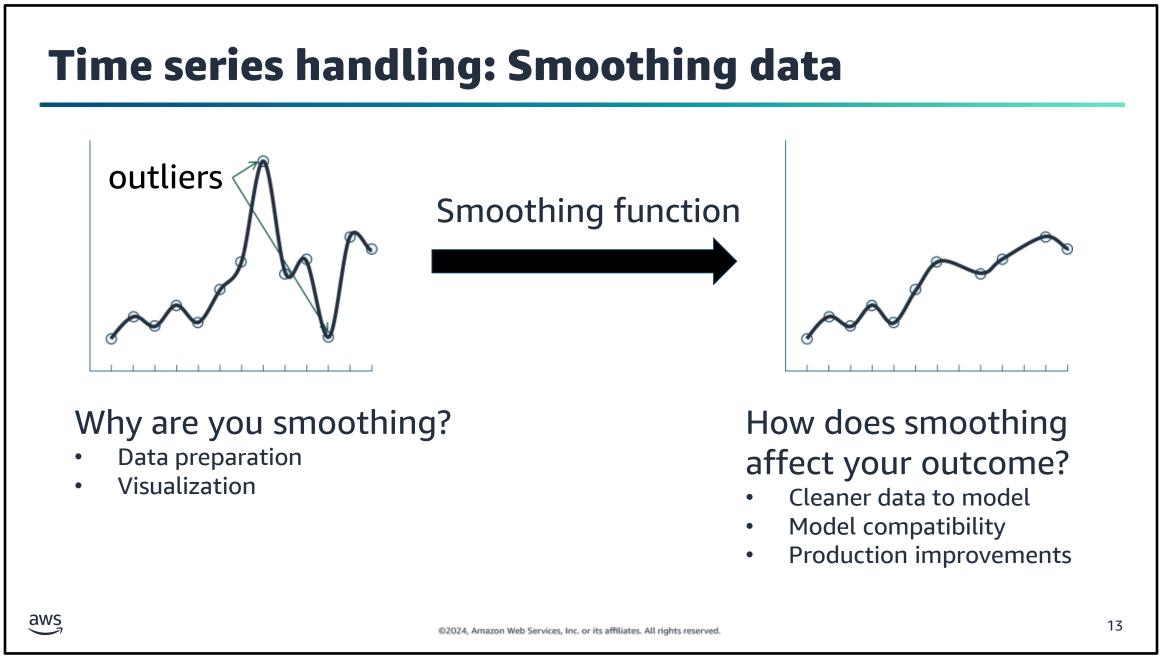
You might obtain data at different frequencies. For example, you might have sales data that includes the exact timestamp that the sale was recorded. However, the inventory data might contain only the year, month, and day of the inventory level. When you have data that is at a different frequency than other datasets, or isn't compatible with your question, you might need to *downsample*.

Downsample means moving from a more finely grained time to a less finely grained time. This example converts an hourly dataset to a daily dataset.

When you downsample, you must decide how to combine the values. In the case of sales data, summing the quantity makes the most sense. If the data is temperature, you might want to find the average. Understanding your data helps you decide what the best course of action is.



The inverse of downsampling is upsampling. The problem with upsampling is that it’s difficult to achieve in most cases. Suppose that you wanted to upsample your sales data from daily sales to hourly sales. Unless you have some other data source to reference, you wouldn’t be able to change from daily to hourly sales. In some cases, you must use additional data or knowledge. For example, if you must match the frequency of another time series, you might have an irregular time series or specific domain knowledge that could help. In those cases, you must be careful of how you convert the data. For the retail example, the best that you can do is create a single order for the day at a specified hour. For temperature, you can copy the daily temperature into each of the hourly slots, or use some formula to calculate a curve.



Outliers can be a problem in data science. The same is true of time series data.

If you examine sales data and you see an order with an unusually large quantity of items, you might not want to include that order in your forecast calculations. The order size might never be repeated. Removing these outliers and anomalies is known as *smoothing*.

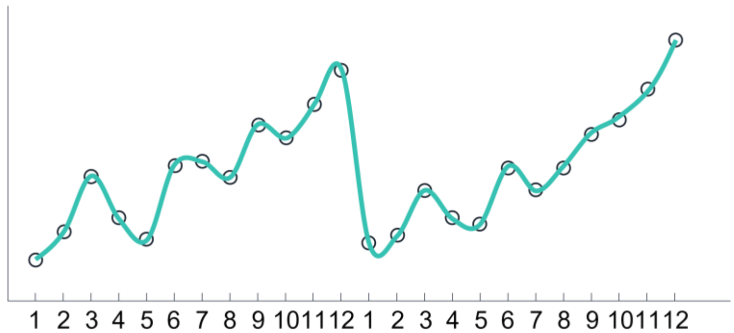
Smoothing your data can help you deal with outliers and other anomalies. You might consider smoothing for the following reasons.

- Data preparation – Removing error values and outliers
- Visualization – Reducing noise in a plot

Understand why you are smoothing the data and the impact that it might have. You might want the outcome to be reduced noise and to create a better model. However, it’s equally important to consider these questions: Could your smoothing compromise the model? Does the model expect noisy data? Can you also smooth the data in production?

Seasonality

- Seasonality frequency
  - Hourly, daily, quarterly, yearly
  - Spring, summer, fall, winter
  - Major holiday sales, winter holiday season
- Incorporating holidays



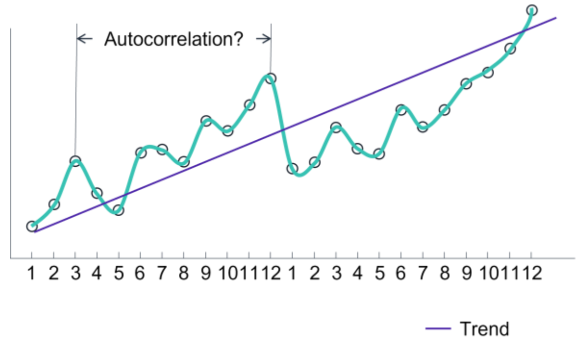
 14

Seasonality in data is any kind of repeating observation where the frequency of the observation is stable. For example, in sales you typically see higher sales at the end of a quarter, and into Q4. Consumer retail exhibits this pattern even more in Q4. Be aware that data can have multiple types of seasonality in the same dataset.

Many times, you might want to incorporate seasonality information into your forecast. Localized holidays are a good example for sales.

Stationarity, trends, and autocorrelation

- Stationarity
  - How stable is the system?
  - Does the past inform the future?
- Trends
  - Correlation issues
- Autocorrelation
  - How points in time are linearly related
- Influences algorithm choice



15

It is important to know how stable a system is. The level of stability, or *stationarity*, can tell you how much you should expect the system's past behavior to inform future behavior. A system with low stability is not good for predicting the future. A stationary time series has constant statistical properties over time, which is important for forecasting.

Often, you will want to determine the trend for a time series. However, adjusting the series for the trend can make it difficult to compare the series with another series that was also adjusted for trend. The trends might dominate the values in the series, which can lead you to overestimate of the correlation between the two series. This phenomenon was shown in the previous topic.

Autocorrelation is one of the special problems that you face with time series data. As you saw in other machine learning problems, the goal of building an ML model is to separate the signal from the noise. Autocorrelation is a form of noise because separate observations are not independent of each other.

A time series with autocorrelation might overstate the accuracy of the model that is produced. Some of the algorithms that you see in this module can help correct for autocorrelation.

Correlations do not mean causation.

Be careful when you interpret your own data, and be cautious about correlations—you don't want to act on correlations that have no real-world meaning. As an experiment, say that you generate two random time series datasets of numbers between 0 and 1. You will find that they have low correlation. However, if you introduce the same slope to both sets of data, you will see a strong correlation.

These factors, along with seasonality, can influence the model that you select to produce your forecast. Some algorithms handle seasonality and autocorrelation, but others do not.

Using pandas for time series data

Time-aware index

```
dataframe['2010-01-04’]
dataframe['2010-02':'2010-03’]
dataframe['weekday_name'] = dataframe.index.weekday_name
```

GroupBy and resampling operations

```
dataframe.groupby('StockCode')
dataframe.groupby('StockCode').resample('D').sum()
```

Autocorrelation

```
dataframe['Quantity'].autocorr()
```


16

The pandas library was developed with financial data analysis in mind. As such, it is good at handling time series data.

You can set the index of your pandas DataFrame to be a *datetime*, which gives you the ability to use the date and time to select your data. You can use ranges that contain partial dates. You can also extract date parts, such as *year*, *month*, *weekday_name*, and more.

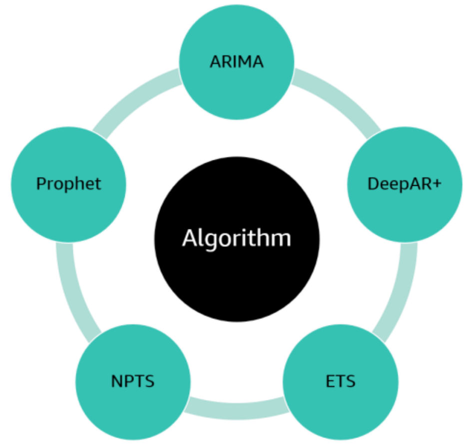
For grouping and resampling tasks, pandas has built-in functions to do both.

Finally, pandas can give you insights into autocorrelation.

For more information about pandas and time series data, see “Time Series/Date Functionality” in the pandas documentation at https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html.

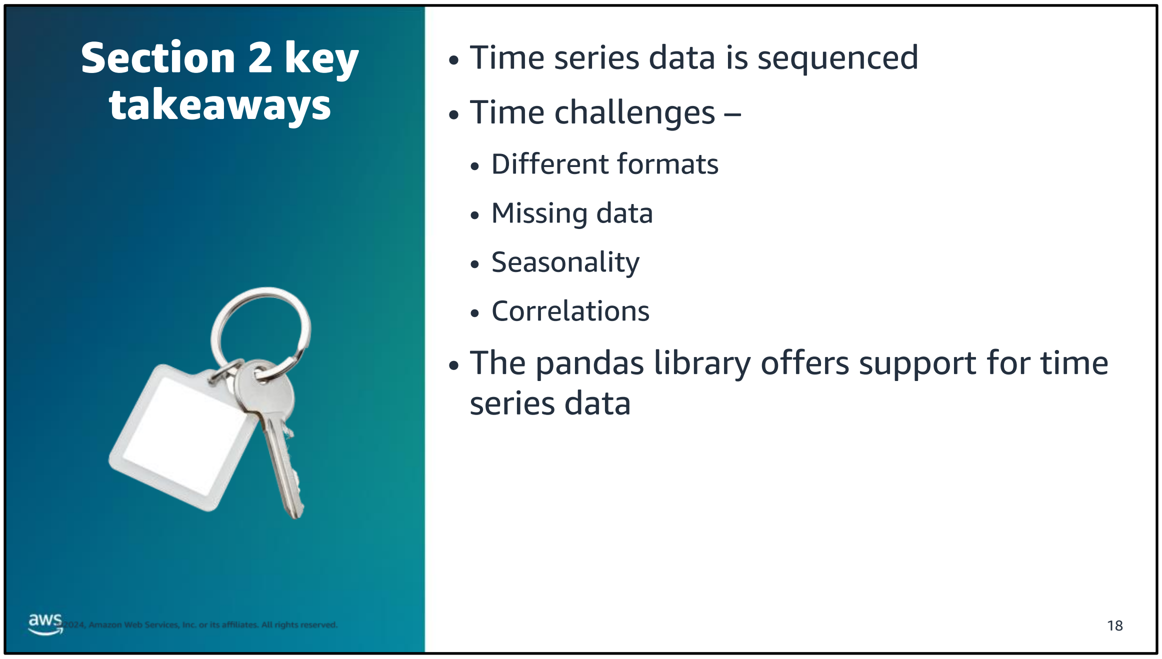
Time series algorithms

- Autoregressive Integrated Moving Average (ARIMA)
- DeepAR+
- Exponential Smoothing (ETS)
- Non-Parametric Time Series (NPTS)
- Prophet



 17

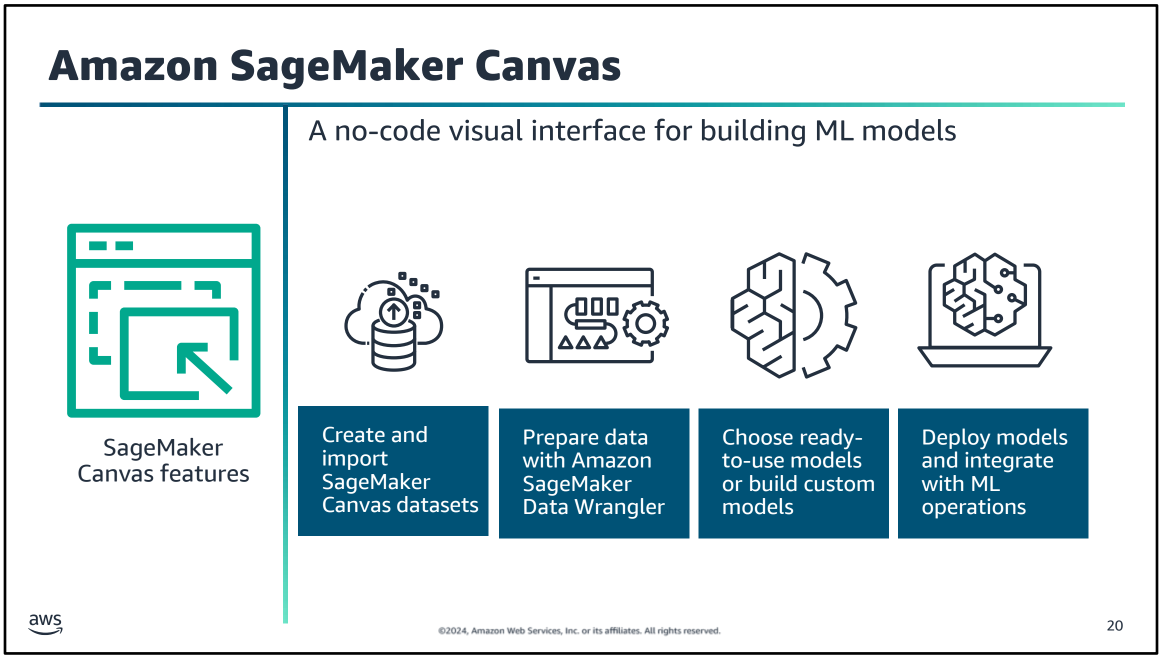
One of the tasks in building a forecasting application is to choose an appropriate algorithm. The type of dataset that you are using and the features of that dataset should determine your choice of algorithm.



Some key takeaways from this section of the module include:

- Time series data is sequenced data that includes a time element, which makes it different from regular datasets
- Some of the time challenges include –
  - Handling different time formats
  - Handling missing data through down sampling, up sampling and smoothing
  - Handling seasonality, such as weekdays and yearly cycles
  - Avoiding bad correlations
- The pandas library offers support for time series data through functions that deal with time

Introducing Section 3: Using Amazon SageMaker Canvas to create a forecast model



If you have ML users who want to create ML models but have limited coding and cloud infrastructure provisioning experience, you can use Amazon SageMaker Canvas. It contains ready-to-use models that you can immediately use to make predictions. Alternatively, you can build your own custom ML model or chat with popular large language models.

SageMaker Canvas gives you the ability to train and deploy custom ML models and generate predictions without the need to write any code. It is a managed service that provisions needed AWS resources on your behalf. SageMaker Canvas contains several features to assist your ML workflow, such as importing and preparing datasets, training ML models, and deploying ML models to make predictions.

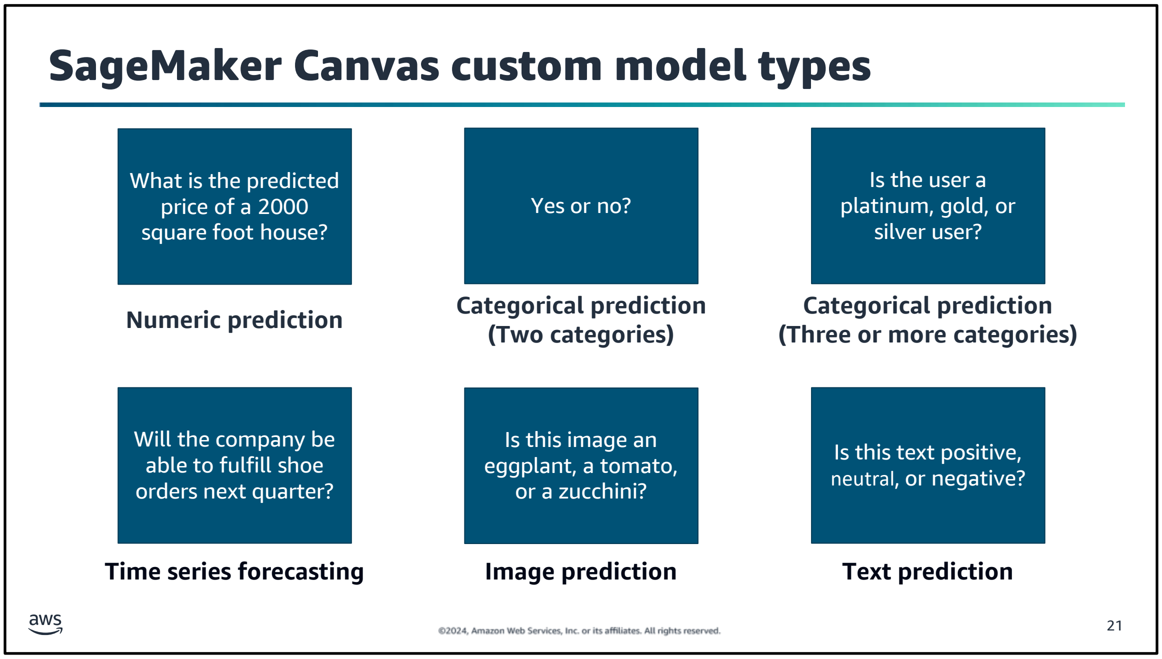
You can import and create SageMaker Canvas datasets. For custom models, you can create datasets for tabular and image data. For ready-to-use models, you can use tabular and image datasets as well as document datasets.

If you're importing datasets larger than 5 GB or you need advanced transformations, use Amazon SageMaker Data Wrangler. With SageMaker Data Wrangler in SageMaker Canvas, you can create a data flow and use various data preparation techniques, such as applying advanced transforms or joining datasets. You can import data into SageMaker Canvas from the following data sources depending on the type of model you want to build.

- Local files on your computer
- Amazon Simple Storage Service (Amazon S3) buckets
- Amazon Redshift provisioned clusters (not Amazon Redshift Serverless)
- AWS Glue Data Catalog through Amazon Athena
- Amazon Aurora
- Amazon Relational Database Service (Amazon RDS)
- Salesforce Data Cloud
- Snowflake
- Databricks, SQLServer, MariaDB, and other databases that use Java Database Connectivity (JDBC) connectors
- More than 40 external software as a service (SaaS) platforms, such as SAP OData

From within SageMaker Canvas, you can choose a ready-to-use model or use the My Models option to train a custom model for your imported dataset. When you build your own model, SageMaker Canvas uses the information in the dataset to build up to 250 models and chooses the one that performs the best.

The ML Operations feature provides model deployment capability and helps you to integrate your model with the machine learning operations (MLOps) processes in your organization.



When you begin building a model, SageMaker Canvas automatically recommends one or more model types depending on the data in your dataset. Model types fall into one of the following categories.

Numeric prediction is known as *regression* in machine learning. Use the numeric prediction model type when you want to make predictions for numeric data. For example, you might want to predict the price of houses based on features such as the house's square footage.

Categorical prediction is known as *classification* in machine learning. The classification known as **two category prediction** (also known as *binary classification* in machine learning) is used when you have two categories that you want to predict for your data. For example, you might want to determine whether a customer is likely to churn or not.

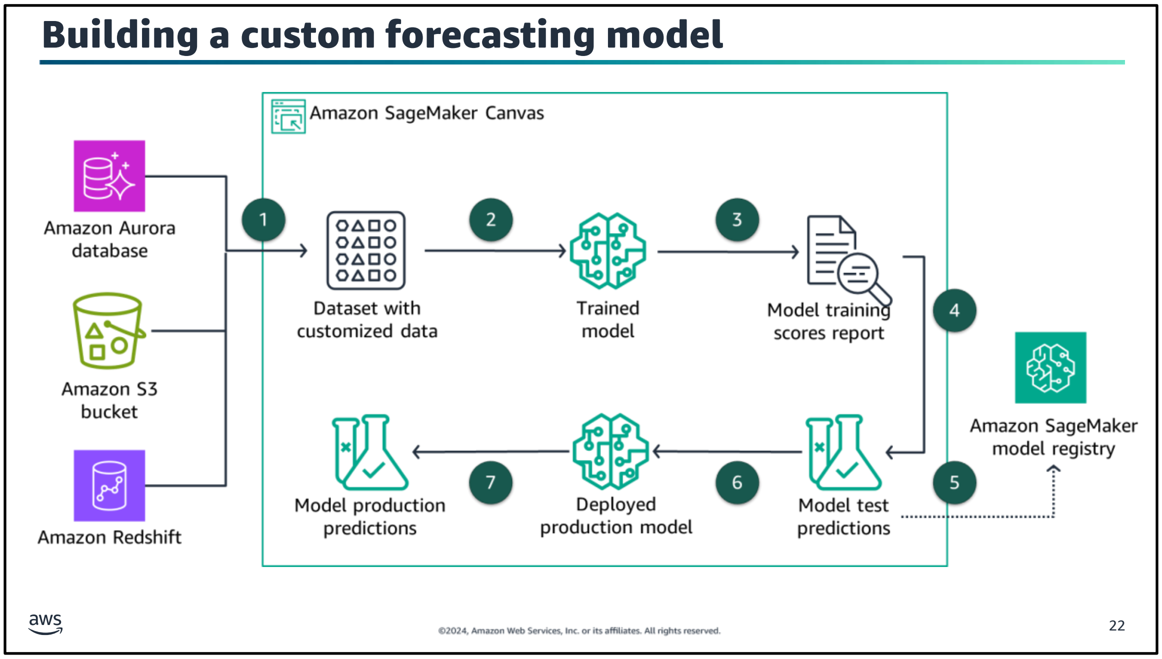
Three plus category prediction (also known as *multi-class classification* in machine learning) is used when you have three or more categories that you want to predict for your data. For example, you might want to predict a customer's loan status based on features such as previous payments or if a customer will fall into a specific user category.

Time series forecasting should be used when you want to make predictions over a period of time. For example, you might want to predict the number of items you'll sell in the next quarter.

Image prediction (also known as *single-label image classification* in machine learning), should be used when you want to assign labels to images. For example, you might want to classify different types of manufacturing defects in images of your product or classify images of vegetables.

Text prediction (also known as *multi-class text classification* in machine learning) should be used when you want to assign labels to passages of text. For example, you might have a dataset of customer reviews for a product, and you want to determine whether customers liked or disliked the product. You might have your model predict whether a given passage of text is positive, negative, or neutral.

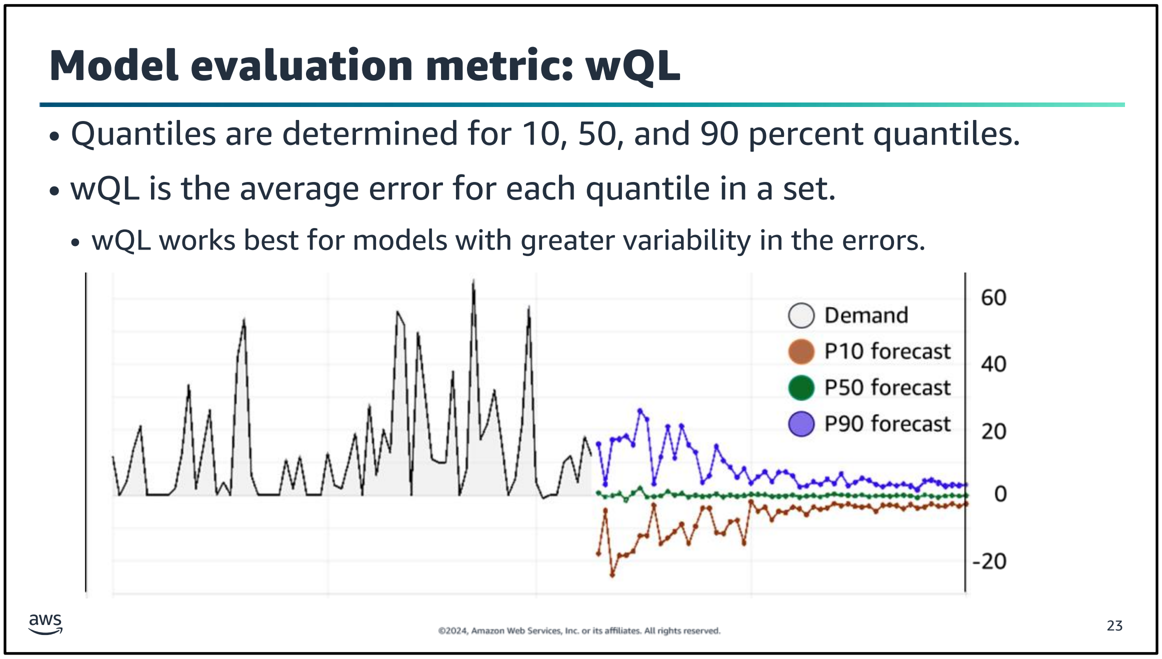
without having to write any code or provision your own resource environments.



If you want a model that is customized to your use case and trained with your data, you can create a new model. By training a custom model on your data, you are able to capture characteristics and trends that are specific and most representative of your data. For example, you might want to create a custom time series forecasting model that you train on inventory data from your warehouse to manage your logistics operations.

You can get predictions customized to your data by implementing the following steps:
1. Import customized data from your data sources. Example data sources include Amazon Aurora, Amazon S3, and Amazon Redshift. You can optionally clean, format, and merge data from different data sources if required. To make a time series forecast, your dataset must have a timestamp column, a target column that has the values that you are using to forecast future values, and an item ID column that contains unique identifiers for each item in your dataset.
2. Build a predictive model by training the model on the customized data. You can choose to do a quick build or a standard build. A quick build will usually take 2–15 minutes and is reasonably accurate. Alternatively, a standard build duration takes 2–4 hours but is the most accurate of the two options.
3. Evaluate the model's training performance and accuracy scores supplied by SageMaker Canvas. For example, for tabular time series models, you can evaluate scores such as Average Weighted Quantile Loss, Mean Absolute Percent Error, Weighted Absolute Percent Error, Root Mean Square Error, and Mean Absolute Scaled Error.
4. If you are satisfied with the model's scores, you can test the model by making batch or single predictions on a deployed test version of the model.
5. When you are satisfied with the prediction tests, you can optionally register the model. A new version of the model will be available in Amazon SageMaker model registry.
6. Your model is now ready for use by applications, and you can deploy the model to a production environment to provide forecast predictions. The deployment configuration requires the model version, a deployment name, and the type and number of Amazon Elastic Compute Cloud (Amazon EC2) instances to use. The deployed model will have a URL to invoke the model for a prediction.
7. You can also use SageMaker Canvas to test single predictions on your production model. This can provide you with latency and prediction results to help ensure your model is working well.

This process can be automated with Amazon SageMaker Canvas automations.

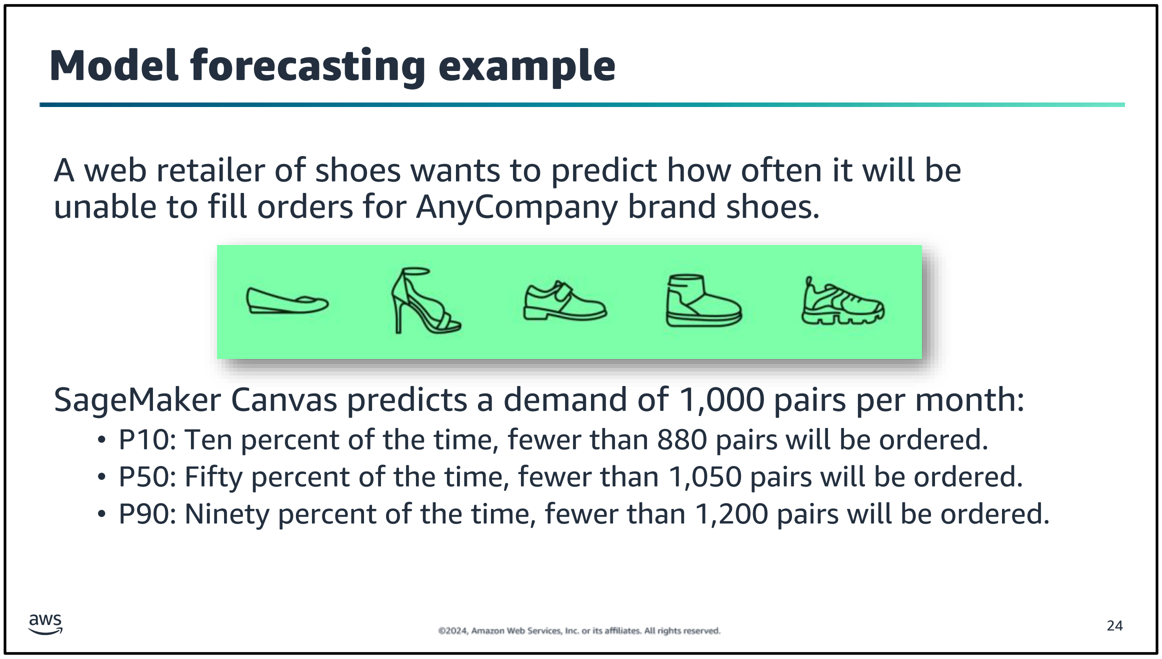


When Amazon SageMaker Canvas creates a forecast, it provides probabilistic predictions at three distinct quantiles: 10 percent, 50 percent, and 90 percent. These prediction quantiles show you how much uncertainty is associated with each forecast.

A P10 quantile predicts that, 10 percent of the time, the true value will be less than the predicted value. For example, suppose that you are a retailer. You want to forecast product demand for winter gloves that sell well only during the fall and winter. Say that you don’t have sufficient storage space and the cost of invested capital is high or that the price of being overstocked on winter gloves concerns you. Then, you might use the P10 quantile to order a relatively low number of winter gloves. You know that the P10 forecast overestimates the demand for your winter gloves only 10 percent of the time, so you will be sold out of your winter gloves 90 percent of the time.

A P50 quantile predicts that 50 percent of the time, the true value will be less than the predicted value. Continuing the winter gloves example, say you know that there will be a moderate amount of demand for the gloves, and you aren’t concerned about being overstocked. Then, you might choose to use the P50 quantile to order gloves.

A P90 quantile predicts that 90 percent of the time, the true value will be less than the predicted value. Suppose you determine that being understocked on gloves will result in large amounts of lost revenue: for example, the cost of not selling gloves is extremely high or the cost of invested capital is low. In this case, you might choose to use the P90 quantile to order gloves.



This example shows how a web retailer might use the accuracy metrics to evaluate a forecast. The retailer wants to predict the demand for sales of a particular brand of shoes. They input the sales records for this brand into SageMaker Canvas to create a prediction.

The prediction provides a forecasted demand of 1,000 pairs with the P10, P50, and P90 values. The wQuantileLoss values indicate that 10 percent of the time (P10), fewer than 880 pairs will be sold. Next, 50 percent of the time (P50), fewer than 1,050 pairs will be sold. Finally, 90 percent of the time (P90), fewer than 1,200 pairs will be sold. The retailer can then use these values to determine which level of inventory to hold. The determination is based on their assessment of the risk that they can’t fulfill orders or that they have excess inventory.

Model evaluation metric: RMSE

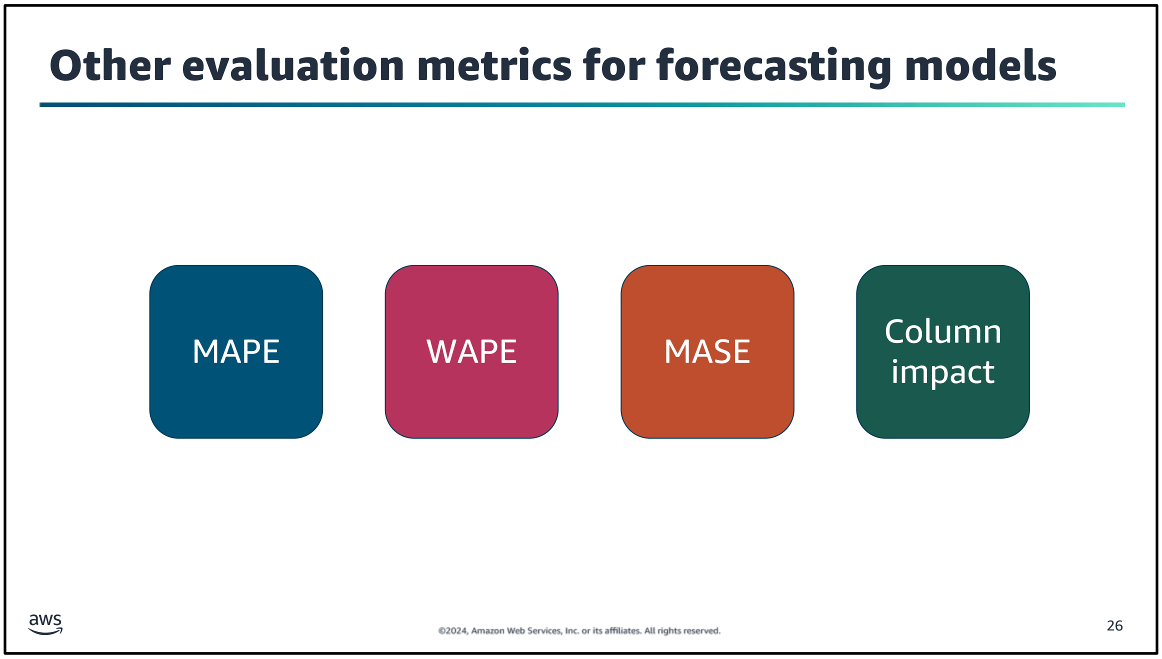
Root mean square error (RMSE) is the root square of prediction errors. The RMSE is the difference between the actual target value and the predicted value.

Test	Actual Result	Prediction	Difference	Difference Squared
1	2	4	2	4
2	4	8	4	16
3	8	13	5	25
4	5	9	4	16

RMSE = root ((sum of squared differences) / number of predictions)
= root ((4 + 16 + 25 + 16) / 4)
= root (61 / 4) = root (15.25) = 3.905

The Root Mean Square Error (RMSE) is another method for evaluating the reliability of your forecasts. Like wQuantileLoss, RMSE calculates how far off the forecasted values were from the actual test data.

The RMSE finds the difference between the actual target value in the dataset and the forecasted value for that time period, and it then squares the differences. The example shows how to calculate RMSE. The RMSE value represents the standard deviation of the prediction errors. This test is good for forecast validity when the errors are mostly of the same size (that is, there aren't many outliers). Lower RMSE metrics indicate that the model's forecasts are more reliable.



In addition to RMSE and wQL, SageMaker Canvas provides three more evaluation metrics. You can also view the column impact that your input data makes on the predictions.

Mean Absolute Percent Error (MAPE) is the percentage error (percent difference of the mean forecasted and actual value) averaged over all time points. A lower value indicates a more accurate model with MAPE equals zero as a perfect model with no errors.

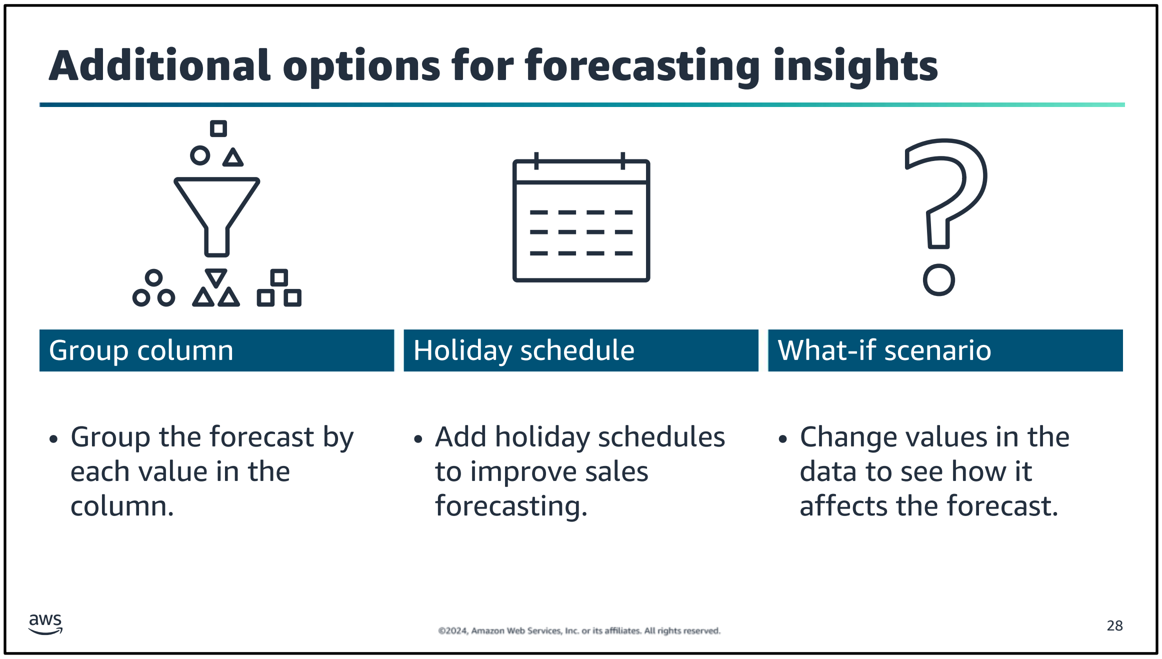
Weighted Absolute Percent Error (WAPE) measures the overall deviation of forecasted values from observed values and is defined by the sum of the absolute error normalized by the sum of the absolute target. A lower value indicates a more accurate model with MAPE equals zero as a perfect model with no errors.

Python example of invoking a time series model

```
import boto3
import pandas as pd

csv_path = './real-time-payload.csv'
data = pd.read_csv(csv_path)

client = boto3.client("runtime.sagemaker")

body = data.to_csv(index=False).encode("utf-8")

response = client.invoke_endpoint(
    EndpointName = "endpoint_name",
    ContentType = "text/csv",
    Body=body,
    Accept = "application/json"
)
```

 27

You can use your SageMaker Canvas models that you've deployed to a SageMaker endpoint in production with your applications. To get a forecast, invoke the endpoint programmatically by calling the runtime SageMaker client.invoke_endpoint function with the endpoint name and data to make the prediction.

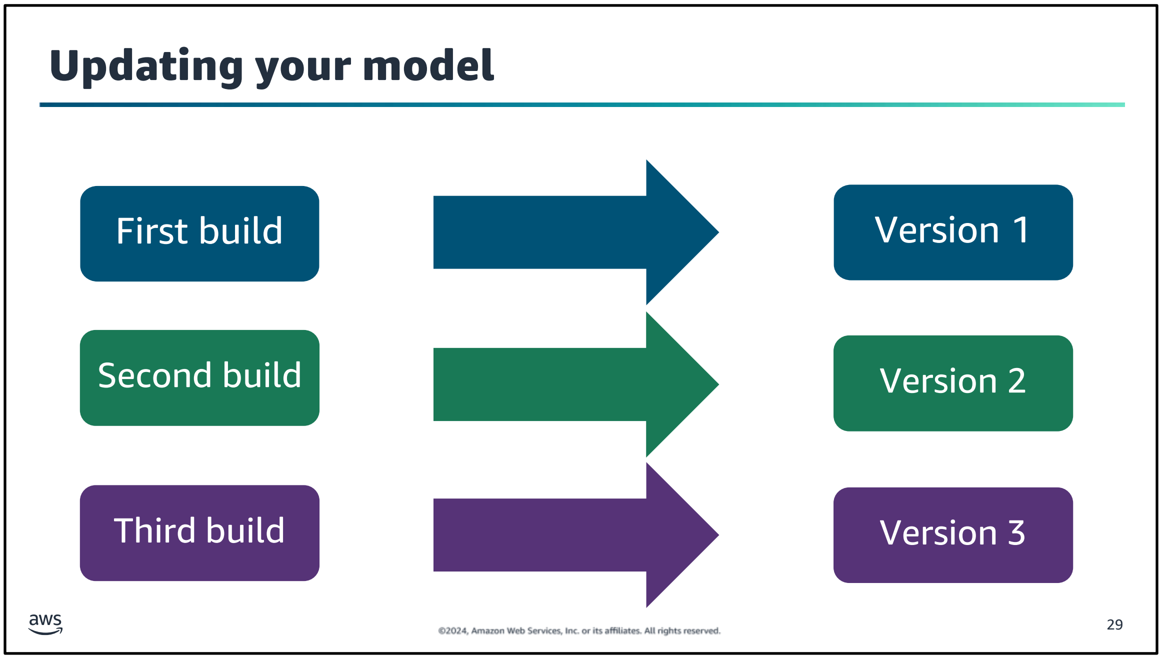


SageMaker Canvas provides you the capability to refine your forecasting insights with group columns, holiday schedules, or what-if scenarios.

You can specify a column in your dataset as a Group column. SageMaker Canvas groups the forecast by each value in the column. For example, you can group the forecast on columns containing price data or unique item identifiers. By grouping a forecast by a column, you can make more specific forecasts. For example, if you group a forecast on a column containing item identifiers, you can see the forecast for each item.

Holidays might impact the overall sales of items. For example, in the US, the number of items sold in both November and December might differ greatly from the number of items sold in January. If you use the data from November and December to forecast the sales in January, your results might be inaccurate. Using a holiday schedule prevents you from getting inaccurate results. You can use a holiday schedule for 251 countries.

For a forecast on a single item in your dataset, you can use what-if scenarios. A what-if scenario gives you the ability to change values in your data and change the forecast. For example, you can answer the following questions by using a what-if scenario: What if I lowered prices? How would that affect the number of items sold?



In SageMaker Canvas, you can update the models that you've built by adding model versions.

Each model that you build has a version number. The first model is version 1 or V1. You can use model versions to see changes in prediction accuracy when you update your data or choose a different build type.

When viewing your model, SageMaker Canvas shows you the model history so that you can compare all of the model versions that you built. You can also delete versions that are no longer useful to you. By creating multiple model versions and evaluating their accuracy, you can iteratively improve your model performance.

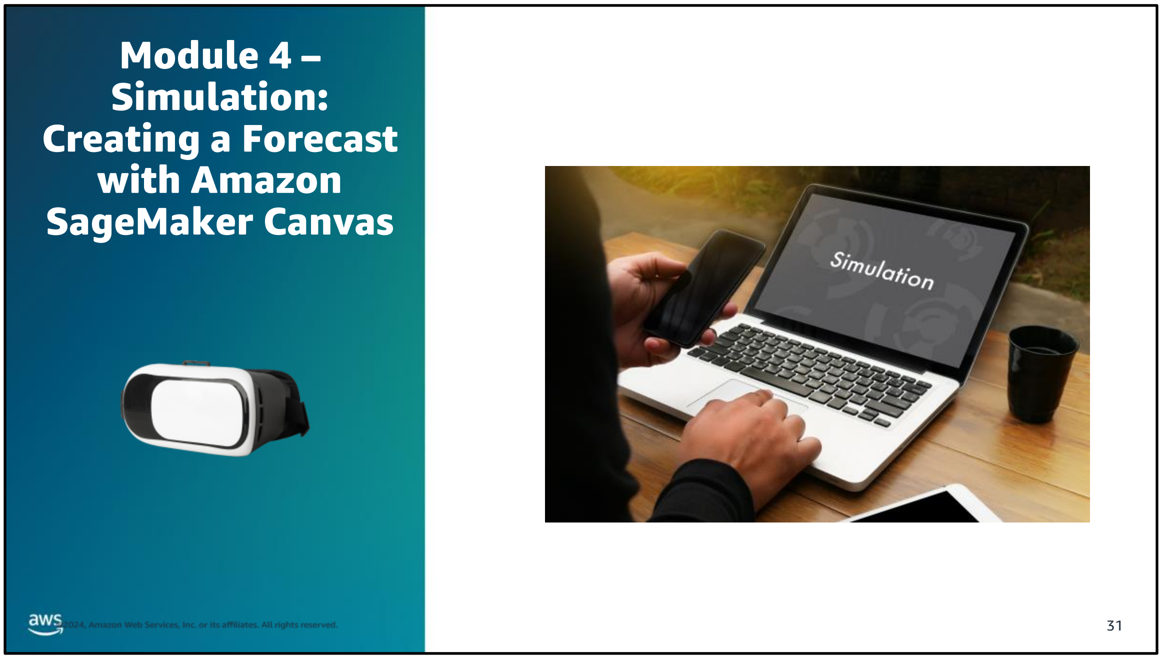
Before you can add a new version, you must successfully build at least one model version. To add a model version, you can either clone an existing version or create a new version.

For example, if you build a model successfully for the first time by using the quick build option, the version number would be version 1. If you then choose the standard build option, SageMaker clones your model, and the new version would be version 2.

Section 3 key takeaways

- Amazon SageMaker Canvas is a no-code visual interface for building ML models.
- SageMaker Canvas provides ready-to-use models and the capability to build your own customized models, such as forecasting models.
- SageMaker Canvas provides evaluation metrics to evaluate your built model.
- You can test your model in SageMaker Canvas test and production environments managed on your behalf.
- SageMaker Canvas provides you the capability to refine your forecasting insights with group columns, holiday schedules, or what-if scenarios.
- Invoke your production model by deploying it to an Amazon SageMaker endpoint.

30



You will now complete the module 4 simulation: Creating a Forecast with Amazon SageMaker Canvas.

It’s now time to review the module and wrap up with a knowledge check.

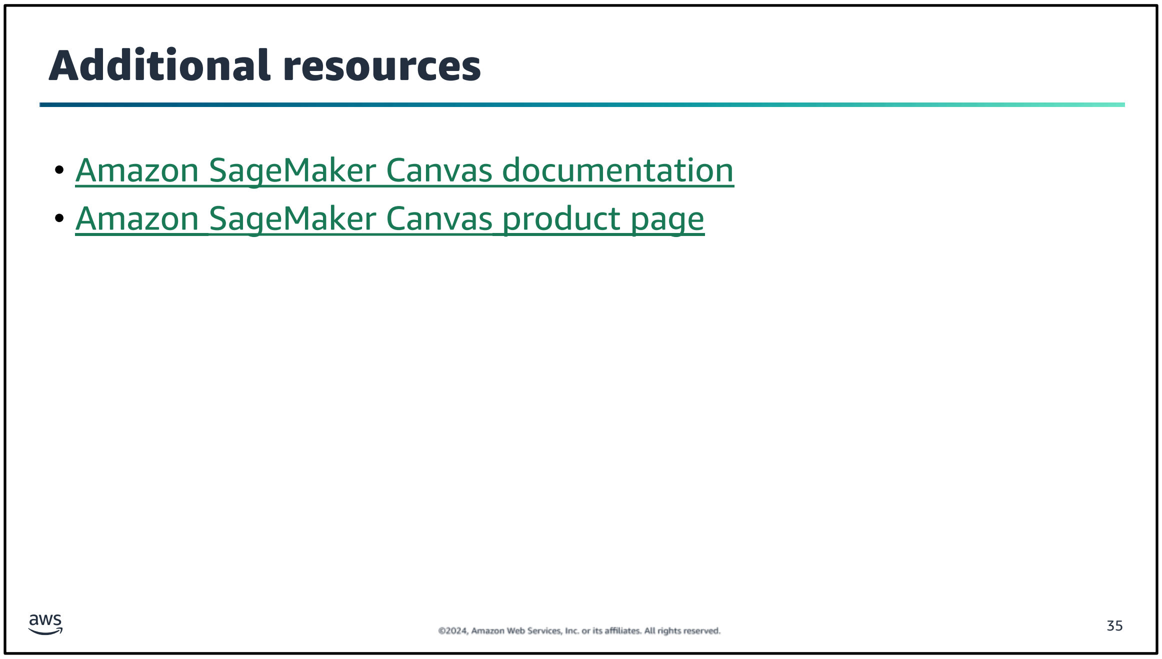
Module summary

In summary, in this module you learned how to:

- Describe the business problems solved by using machine learning forecasting.
- Describe the challenges of working with time series data.
- List the steps that are required to generate a forecast by using Amazon SageMaker Canvas.
- Use Amazon SageMaker Canvas to make an in-app prediction.

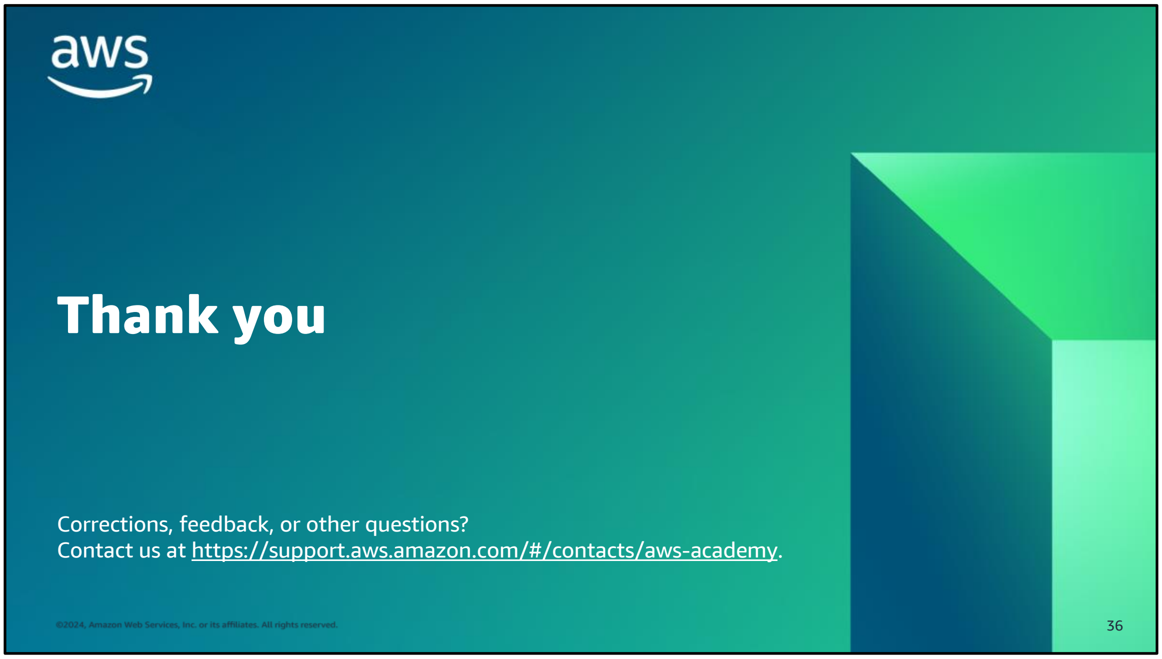
Complete the knowledge check

It is now time to complete the knowledge check for this module.



If you want to learn more about the topics that are covered in this module, you might find the following additional resources helpful:

- For Amazon SageMaker Canvas documentation, see "SageMaker Canvas" in the SageMaker Developer Guide at https://docs.aws.amazon.com/sagemaker/latest/dg/canvas.html.
- For the Amazon SageMaker Canvas product page, see "Amazon SageMaker Canvas" at https://aws.amazon.com/sagemaker/canvas/.



Thank you for completing this module.

