AWS Whitepaper

Time Series Forecasting Principles with Amazon Forecast

Time Series Forecasting Principles with Amazon Forecast: AWS Whitepaper

Table of Contents

Time Series Forecasting Principles with Amazon Forecast

Publication date: **September 1, 2021** (*Document history*)

Companies today use everything from simple spreadsheets to complex financial planning software in a bid to accurately forecast future business outcomes such as product demand, resource needs, and financial performance. This paper introduces forecasting, its terminology, challenges, and use cases. This document uses a case study to reinforce forecasting concepts, forecasting steps, and references how Amazon Forecast can help solve the many practical challenges in real-world forecasting problems.

Overview

Forecasting is the science of predicting the future. By using historical data, businesses can understand trends, make a call on what might happen and when, and in turn, build that information into their future plans for everything from product demand to inventory planning and staffing.

Given the consequences of forecasting, accuracy matters. If a forecast is too high, customers may over-invest in products and staff, which results in wasted investment. If the forecast is too low, customers may under-invest, which leads to a shortfall in raw materials and inventory, creating a poor customer experience.

Today, businesses try to use everything from simple spreadsheets to complex demand/financial planning software to generate forecasts, but high accuracy remains elusive for two reasons:

- First, traditional forecasts struggle to incorporate large volumes of historical data, missing important signals from the past that are lost in the noise.
- Second, traditional forecasts rarely incorporate related but independent data, which can offer important context (such as price, holidays/events, stock-outs, marketing promotions, and so on). Without the full history and the broader context, most forecasts fail to predict the future accurately.

Amazon Forecast is a fully managed service that overcomes these problems. Amazon Forecast provides the best algorithms for the forecasting scenario at hand. It relies on modern machine learning (ML) and deep learning when appropriate to deliver highly accurate forecasts. Amazon Forecast is easy to use and requires no machine learning experience. The service automatically provides the necessary infrastructure, processes data, and builds custom/private ML models that are hosted on AWS and ready to make predictions. In addition, as advances in machine learning techniques continue to evolve at a rapid pace, Amazon Forecast incorporates these, so that customers continue to see accuracy improvements with minimal to no additional effort on their part.

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the Framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the AWS Well-Architected Tool, available at no charge in the AWS Management Console, you can review your workloads against these best practices by answering a set of questions for each pillar.

In the Machine Learning Lens, we focus on how to design, deploy, and architect your machine learning workloads in the AWS Cloud. This lens adds to the best practices described in the Well-Architected Framework.

For more expert guidance and best practices for your cloud architecture—reference architecture deployments, diagrams, and whitepapers—refer to the AWS Architecture Center.

About forecasting

In this document, forecasting means predicting the future values of a time series: the input or output to a problem is of a time series nature.

Forecasting system

A forecasting system includes a diverse set of users:

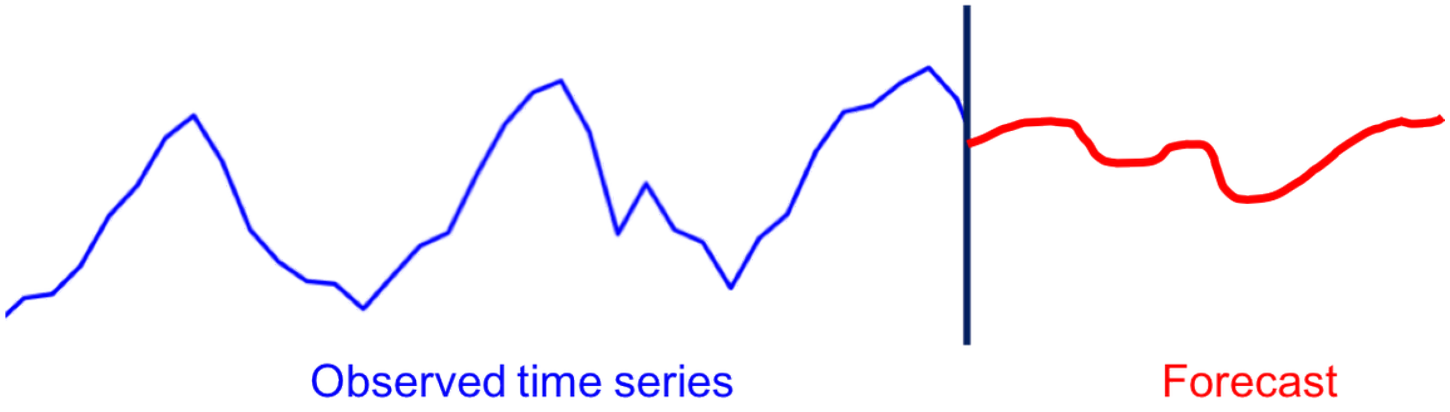
- **End users**, who query the forecast for a specific product, and decide how many units to purchase; this may be a person or an automated system.
- **Business analysts/business intelligence**, who support end users, run and organize aggregate reporting.
- **Data scientists**, who iteratively analyze the demand patterns, causal effects, and add new features to deliver incremental improvements to the model or improve the forecasting model.
- **Engineers**, who set up the infrastructure of the data collection, and ensure the availability of the input data to the system.

Amazon Forecast alleviates the work of software engineers and allows businesses with limited data science capabilities to leverage state of the art forecasting technology. For businesses with data science capabilities, a number of diagnostic functionalities are included so that forecasting problems are well addressed with Amazon Forecast.

Where do forecasting problems occur?

Forecasting problems occur in many of the areas that naturally produce time series data. These include retail sales, medical analysis, capacity planning, sensor network monitoring, financial analysis, social activity mining, and database systems. For example, forecasting plays a key role in automating and optimizing operational processes in most businesses that enable data-driven decision making. Forecasts for product supply and demand can be used for optimal inventory management, staff scheduling, and topology planning, and are more generally a crucial technology for most aspects of supply chain optimization.

The following figure contains a summary of the forecasting problem when based on an observed time series that exhibits a pattern (in this example, seasonality), and a forecast is created over a specified period. The horizontal axis represents time going from the past (left) to the future (right). The vertical axis represents measured units. Given the past (in blue) up to the vertical black line, identifying the future (in red) is the forecasting task.



Forecasting task overview

Considerations before attempting to solve a forecasting problem

The most important questions to understand before solving forecasting problems are:

- Do you need to solve a forecasting problem?
- Why are you solving the forecasting problem?

As a consequence of the ubiquity of time series data, it is easy to find forecasting problems everywhere. However, a key question is whether there truly is a need for solving a forecasting problem or whether you can circumvent it completely without sacrificing efficient decision making in the business. Posing this question is important because, scientifically speaking, forecasting is among the hardest problems in machine learning.

For example, consider product recommendations for an online retailer. This product recommendation problem can be framed as a forecasting problem where, for each customer-stock keeping unit (SKU) pair, you forecast the number of units of a specific item that this particular customer would purchase. This problem formulation has a number of benefits. One benefit is that the time component is taken explicitly into account, so you could recommend products according to the buying patterns of the customers.

However, product recommendation problems are rarely formulated as a forecasting problem, since solving such a forecasting problem is much harder (for example, the sparsity of the information at the customer-SKU level and the scale of the problem) than directly solving the recommendation problem. Therefore, as you think about a forecasting application, it is important to consider the downstream usage of the forecast, and whether it is possible to address this problem using an alternative approach.

Amazon Personalize can help in these cases. Amazon Personalize is a machine learning service that makes it easy for developers to create individualized recommendations for customers using their applications.

After determining that you need to solve a forecasting problem, the next question to ask is, why are you solving the forecasting problem? In many business settings, forecasting is usually just a means to an end. For example, for demand forecasting in a retail context, the forecast may be used to make inventory management decisions. The forecast problem is typically an input to a decision problem, which in turn may be modeled as an optimization problem.

Examples of such decision problems include the number of units to be purchased or the best approach to deal with the existing inventory. Other business forecasting problems include forecasting server capacity or forecasting demand for raw materials/parts in a manufacturing context. These forecasts may be used as inputs for other processes, either for decision problems as above, or for scenario simulations, which are then used for planning without explicit models. There are exceptions to the rule that forecasting is not an end in itself. In financial forecasting, for example, the forecast is used directly to build up financial reserves or is presented to investors.

To understand the purpose of the forecast, consider the following questions:

- How long into the future should you forecast?
- How often do you need to generate forecasts?
- Are there specific aspects of the forecasts that you should dive deep into?

Case study: Retail demand forecasting problem for an e-commerce business

To illustrate forecasting concepts in more detail, consider the case of an e-commerce business that sells products online. Optimizing decisions in the supply chain (for example, in- stock management) is critical to the core competitiveness of this business because it helps having the accurate number of products in the appropriate fulfillment locations. This essentially means having a large selection available with shorter shipping times and competitive prices, which leads to higher customer satisfaction. The key input into the supply chain software system is a prediction of demand or the forecast of potential sales of every product in the catalog. This forecast enables important downstream decisions, key among them being:

- **Macro-level planning (strategic forecasting):** For a business as a whole, what is the projected growth in terms of total sales/revenue? Where should the business be (more) active geographically? How should labor be staffed?
- **Demand (or inventory) forecasting:** How many units of each product are expected to be sold per location?
- **Promotional activity (tactical forecasting):** How should promotions be run? Should products be liquidated?

The rest of the case study focuses on the second problem, which is part of the family of *operational forecasting problems* (Januschowski & Kolassa, 2019). This document follows the main concerns: data, models (predictors), inferences (forecasts), and productionization.

For this case study, it is important to keep in mind that the forecasting problem is a means to an end. Although the forecasts are crucially important for the business, the downstream supply chain decisions are what is even more important. In our case study, these decisions are taken by automated buying systems that rely on mathematical optimization models from operations research. These systems try to minimize the *expected* cost for the business.

The key word is *expected*, meaning that the forecasts ought to cover not only one possible future but all possible futures, with the appropriate weighting according to the probability of a particular outcome. To this end, the key enabler for downstream decision making is a full distribution of the forecast values rather than just having a point forecast. The following figure shows a *probabilistic forecast* (also referred to as a *density forecast*). Note that you can derive a single point forecast (the most likely future) easily from this probabilistic forecast, but going from a point forecast to a probabilistic forecast is more difficult.

Given a probabilistic forecast, you can obtain different statistics from it and tailor the results to assist in the decision you want to take. The e-commerce business may have a number of key products for which they almost never want to be out of stock. In this case, use a high quantile (for example, the 90th percentile), which would translate to *90% of the time the products are going to be in-stock*. For other products, such as products for which replacements are more easily found (such as pencils), using a lower percentile may be more appropriate.

In Amazon Forecast, you can obtain different quantiles from the probabilistic forecast easily.

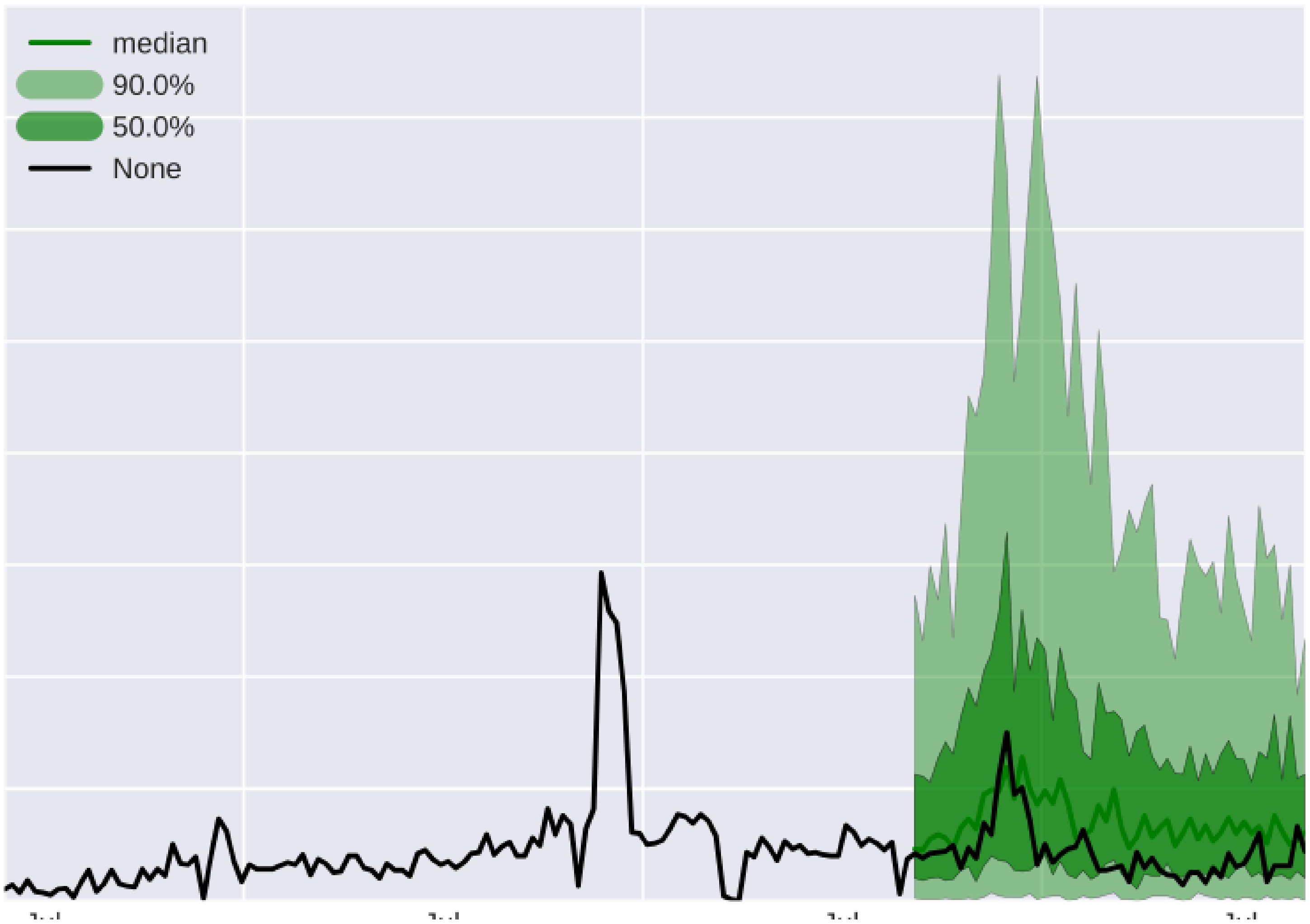


Illustration of probabilistic forecast

In the preceding figure, the black line is the actual values; the dark green line is the median of the forecast distribution; the dark green shaded area is the prediction interval into which you expect

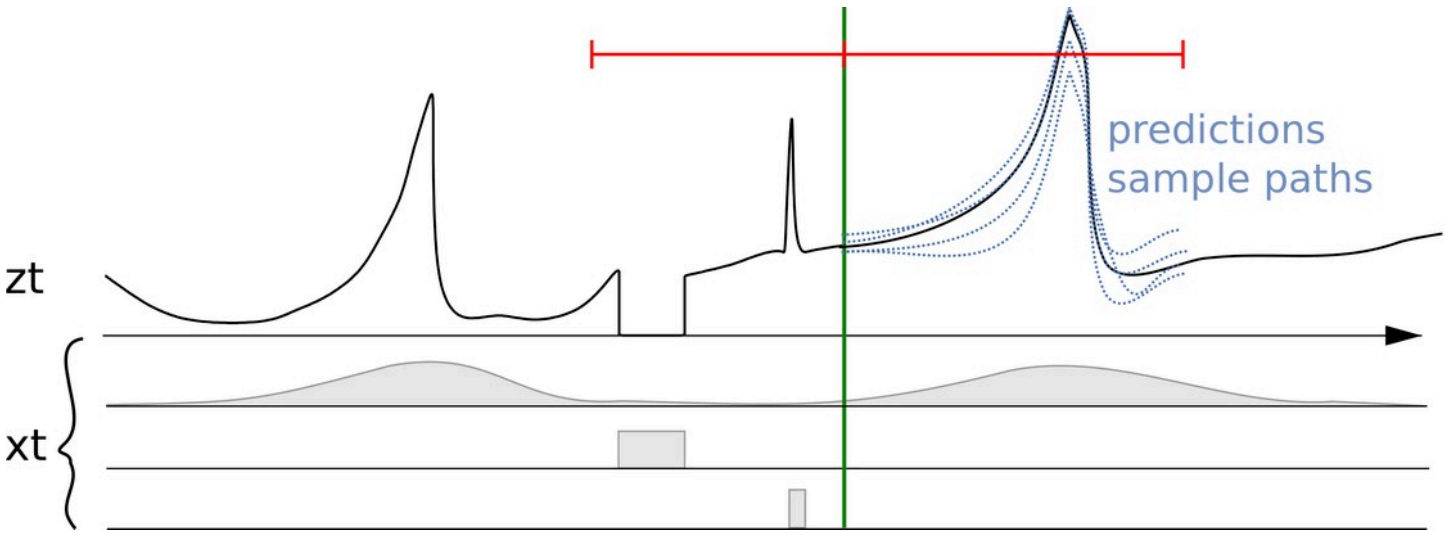
50% of the values to fall; and the light green area is the prediction interval into which you expect 90% of the actual values would fall.

The following sections cover the steps involved for solving the forecasting problem for this business, including:

- Data collection and aggregation (Step 1)
- Data preparation (Step 2)
- Creating a predictor (Step 3)
- Evaluating predictors (Step 4)
- Automating forecast generation (Step 5)

Step 1: Collect and aggregate data

The following figure shows a mental model for the forecasting problem. The goal is to forecast the time series **z_t** into the future, using as much relevant information as possible to make the forecast as accurate as possible. Therefore, the first and most important step is to collect as much as possible of the correct data.



A time series z_t together with associated features or co-variates (x_t) and multiple forecasts

In the preceding figure, multiple forecasts are shown to the right of the vertical line. These forecasts are samples of the probabilistic forecast distribution (or, conversely, can be used to represent the probabilistic forecast).

The key information for a retail business to record is:

- **The transaction sales data** — For example, the stock keeping unit (SKU), location, timestamp, and units sold.
- **SKU item details data** — The metadata of an item. Examples include color, department, size, and so on.
- **Price data** — The price time series of each item with time stamps.
- **Promotion information data** — Different types of promotions, either over a collection of items (category) or individual items with time stamps.
- **In-stock information data** —For every unit of time, the information whether a SKU was in-stock or purchasable versus the SKU was out of stock.

- **Location data** — The location of an item or sale at a given time point can be represented as a string `location_id` or `store_id` or as an actual geolocation. Geolocations can be country code plus five-digit zip code, or `latitude_longitude` coordinates. The location is considered a "dimension" of transactional sales.

In Amazon Forecast, the historical data of the quantity to be forecasted is called the Target Time Series (TTS). For the retail business, the TTS is the transactional sales data. Other historical data, that is known at exactly the same time as every sales transaction, is called the Related Time Series (RTS). For the retail business, the RTS would include price, promotion, and in-stock variables.

Note that in-stock information is important since this problem centers on forecast *demand* and not sales, but the business only records sales. When a SKU goes out of stock, the number of sales is lower than the potential demand, so it is important to know and record when such out-of-stock events occur.

Other datasets to consider include the number of webpage visits, details on search terms, social media, and weather information. It is often important to have data available for the past *and* for the future to be able to use this data in models. This is a requirement of many forecasting models and in *backtesting* (described in the Step 4: Evaluate forecasts section).

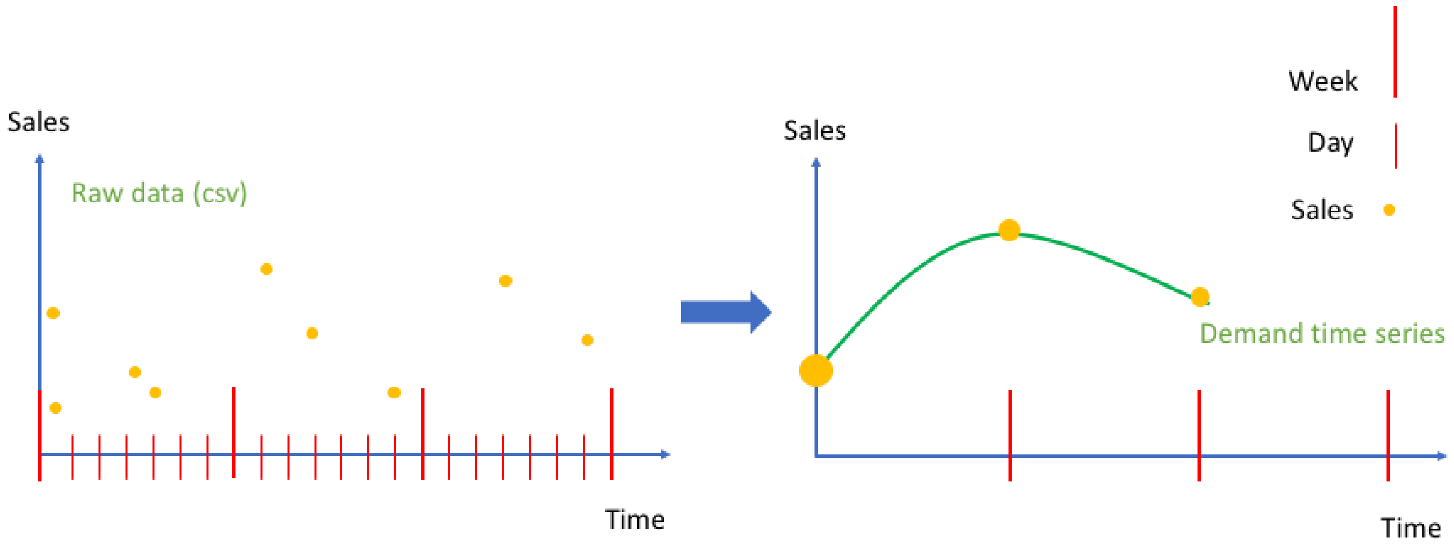
For some forecasting problems, the frequency of the raw data naturally matches that of the forecasting problem. Examples include the request of the server volume, which is sampled by minute, when you want to forecast at minute frequency.

Data is often recorded at finer frequency, or simply at arbitrary timestamps within a time range, but the forecasting problem is at a coarser granularity. This is a common occurrence in the retail case study, where the sales data is normally recorded as *transactional* data; for example, the format consists of a timestamp with a fine granularity of when sales happened. In the forecasting use case, this low granularity may not be needed, and it may be more appropriate to aggregate this data up into hourly or daily sales. Here, the level of aggregation corresponds to the downstream problem; for example, inventory management or resource planning.

Example

In the following figure, the left graph shows an example of the raw customer sales data that can be input to Amazon Forecast as a comma separated value (CSV) file. In this example, the sales data is defined on a finer daily time grid, and the problem is to forecast the weekly demand on the coarser time grid into the future. Amazon Forecast performs the aggregation of the daily values in a given week in the `create_predictor` API call.

The result transforms the raw data into a collection of well-formed time series with a fixed weekly frequency. The right graph illustrates this aggregation on the target time series using the default summation aggregation method. Other aggregation methods include averaging, maximum, minimum, or choosing a single point (for example, the first). The aggregation granularity and method must be chosen such that it best matches the business use case of the data. In this example, the aggregated value is aligned to weekly aggregation. Other aggregation methods can be set by the user using the `FeaturizationMethodParameters` key of the `FeaturizationConfig` parameter of the `create_predictor` API.



Aggregation of raw sales data as events (left), into an equally spaced time series (right)

Step 2: Prepare data

Once you have raw data available, you need to handle complications, such as missing data, and make sure that you prepare the data for the forecasting models that best capture the intended interpretation.

How to handle missing data

A common occurrence in real-world forecasting problems is the presence of missing values in the raw data. A missing value in a time series means that the true corresponding value at every time point with the specified frequency is not available for further processing. There can be multiple reasons for values being marked as missing.

Missing values can occur because of no transaction, or possible measurement errors (for example, because a service that monitored certain data was not working correctly or because the measurement could not occur correctly). The primary example for the latter in the retail case study is an out-of-stock situation in demand forecasting, which means that demand does not equal the sales on that day.

Similar effects can occur in cloud computing scenarios when a service has reached a limit (for example, Amazon EC2 instances in a certain AWS Region are all busy). Another example of missing values occurs when a product or service has not yet been launched, or goes out of production.

Missing values can also be inserted by feature processing components, to ensure equal lengths of the time series with padding. If prevalent enough, missing values can significantly impact a model's accuracy.

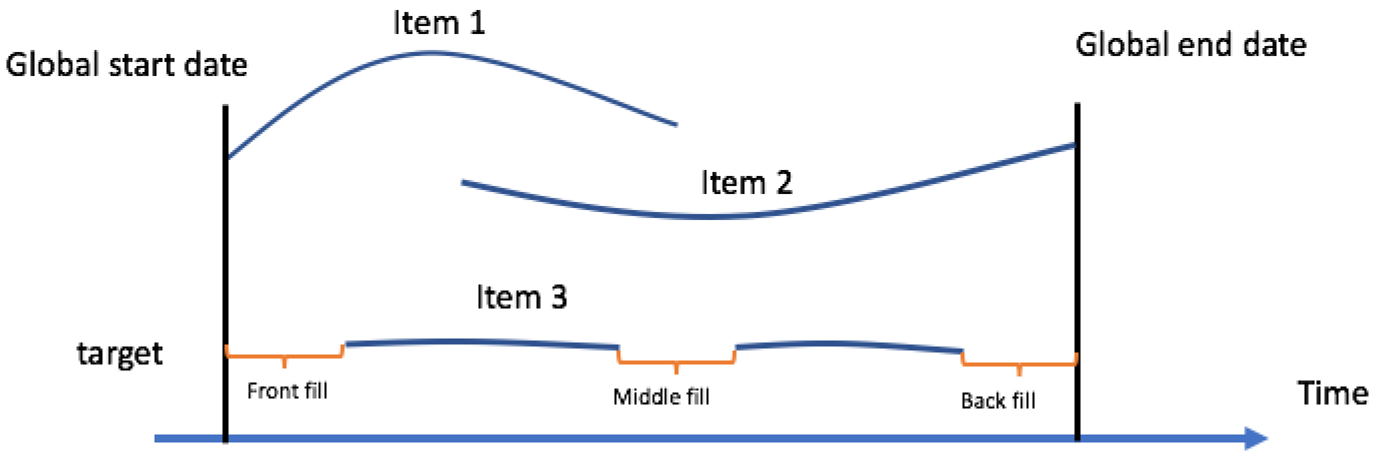
Example 1

Filling is the process of adding standardized values to missing entries in your dataset. In the following figure, the different strategies to handle missing values in Amazon Forecast—front, middle, back, and future filling—are illustrated for item 2 in a dataset of three items.

Amazon Forecast supports filling for both the target and related time series. The global start date is defined as the earliest start date over the start dates of all the items in your dataset. In the below example, the global start date occurs for item 1. Similarly, the global end date is defined as the latest end date of the time series over all items, which occurs for item 2.

Front fill fills in every value from the start of the particular time series to the global start date. At the time of publishing this document, Amazon Forecast does not turn on any front filling, and allows all the time series to start at different time points. Middle fill denotes values that have been filled in the middle of the time series (for example, between the items' start and end dates), and back fill fills from the last date of that time series to the global end date.

For the target time series, the middle and back fill methods have a default filling logic of zero. Future fill (which applies only to the related time series) fills any missing value between the items' global end date and the forecast horizon specified by the customer. Future values are required for using the related time series dataset with Prophet and DeepAR+, and optional for CNN-QR.



Missing value handling strategies in Amazon Forecast

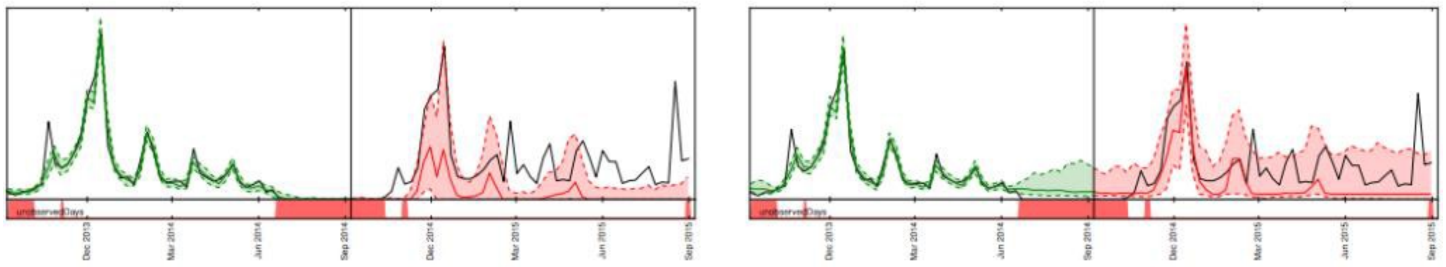
In the preceding figure, the global start date denotes the earliest start date over the start dates of all the items, and the global end date denotes the latest end date over the end dates of all items. The Forecast Horizon is the period over which Forecast provides predictions for the target value.

This is a common scenario in the retail study that represents zero sales for transactional data for available items. These values are treated as true zeros, and used in the metrics evaluation component. Amazon Forecast enables the user to identify values that are actually missing and encode them as not a number (NaN) for the algorithms to process. This paper next examines why these two cases differ and when each is useful.

In the retail case study, the information that a retailer sold zero units of an available item differs from the information that zero units of an unavailable item are sold, either in the periods outside its existence (for example, before its launch or after its deprecation), or in periods within its existence (for example, partially out of stock, or when there was no sales data recorded for this time range). The default zero filling is applicable in this former case. In the latter, even though the corresponding target value is typically zero, there is additional information conveyed in the value being marked as missing. Best practice is to preserve the information that there was missing data and not discard this information. See the following example for an illustration of why keeping the information is important.

Amazon Forecast supports additional filling logic of value, mean, median, minimum, and maximum. For the related time series (for example, price or promotion), there are no defaults specified for middle, back, or future fill methods, because the correct missing values logic varies by attribute type and use case. The filling logic supported for the related time series include zero, value, mean, median, minimum, and maximum.

To perform missing value filling, specify the types of filling to implement when you call the `CreatePredictor` operation. Filling logic is specified in `FeaturizationMethod` objects. For example, to encode a value that does not represent zero sales of an unavailable product in the target time series, mark a value as truly missing by setting the fill type equal to NaN. Unlike zero filling, the values encoded with NaN are treated as truly missing, and not used in the metrics evaluation component



The effect of 0-filling vs filling by NaN on the forecasts for the same item

In the preceding figure, in the left graph, the values to the left of the vertical black line are filled with 0s, leading to an under-biased forecast (to the right of the vertical black line). In the right graph, these values are marked as NaN, leading to appropriate forecasts.

Example 2

The preceding figure illustrates the importance of handling missing values correctly for a linear state space model, such as ARIMA or ETS. It plots the demand forecast for an item that is partially out of stock. The training region is shown in the left graph in green, the prediction range in the right panel in red, and the true target in black. The median, p10, and p90 forecasts are shown in the red line and the shaded region, respectively. The bottom shows the out-of-stock items (80% of the data) marked in red. In the left plot, the out-of-stock areas are ignored and filled by 0.

This results in the forecast models assuming that there are a lot of zeros to predict, and therefore the forecasts are too low. In the right plot, the out-of-stock areas are treated as true missing observations, and the demand becomes uncertain in the out-of-stock region. With the missing values for the out-of-stock items appropriately marked as NaN, you see no under-bias in the prediction range in this plot. Amazon Forecast fills in these gaps in data, making it easy for you to correctly handle missing data, without having to explicitly modify all of their input data.

Concepts of featurization and related time series

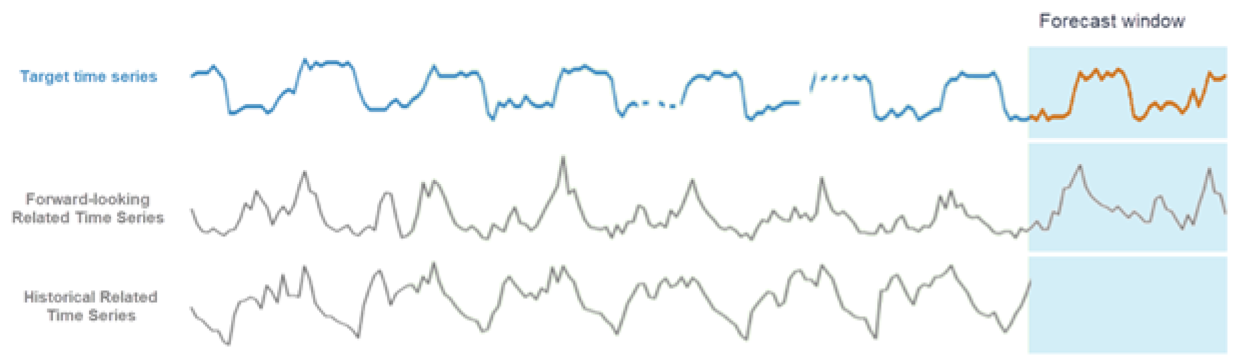
Amazon Forecast enables users to input related data to help improve the accuracy of certain supported forecasting models. This data can be of two types: *related time series* or static item metadata.

Note

Metadata and Related data are referred to as *features* in machine learning, and *covariates* in statistics.

Related time series are time series that have some correlation with the target value, and should lend some statistical strength to forecast on the target value because they provide an explanation in intuitive terms (see Amazon Forecast: predicting time-series at scale for an example). Unlike the target time series, related time series are known values in the past that may impact the target time series, and may have known values in the future.

In Amazon Forecast, you can add two types of related time series: historical time series and forward-looking time series. Historical related time series contain data points up to the forecast horizon, and do not contain any data points within the forecast horizon in the future. Forward-looking related time series contain data points up to and within the forecast horizon.



Different approaches regarding using related time series with Amazon Forecast

Example 3

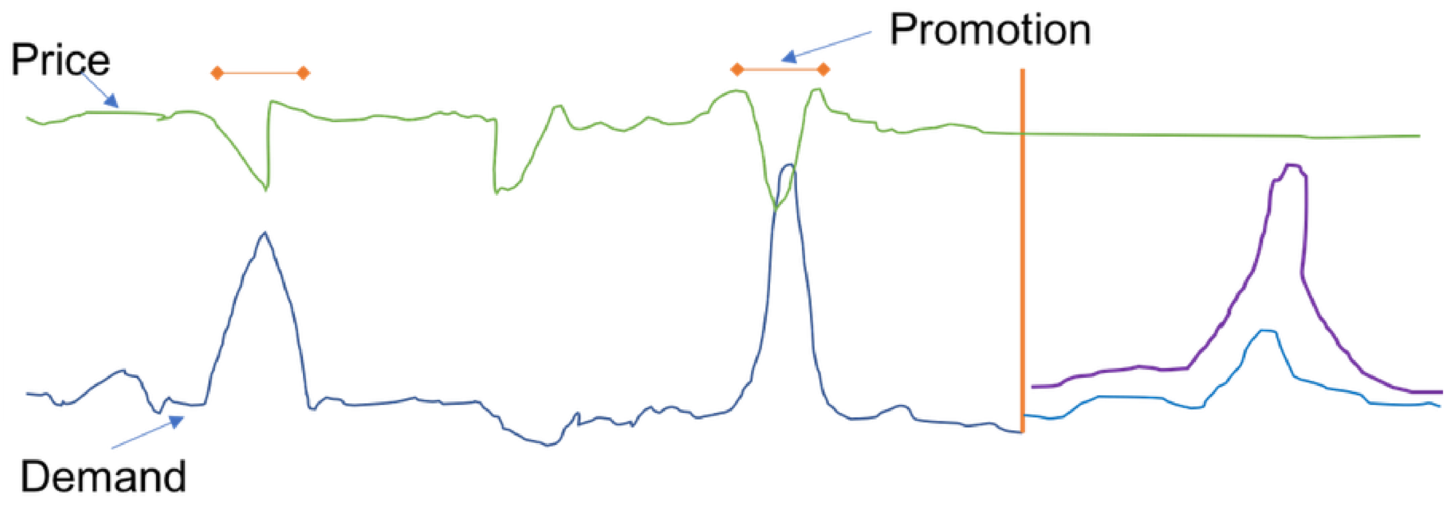
The following figure shows an example of how related time series can be used to predict the future demand of a popular book. The blue line represents the demand in the target time series. The price is shown as the green line. The vertical line represents the forecast start date, and the forecasts at the two quantiles are shown to the right of the vertical line.

This example uses a forward-looking related time series that aligns with the target time series at the forecasting granularity, and is known at all (or most) times in the future in the range of the forecast start date to the forecast start date incremented by the forecast horizon (forecast end date).

The following figure also shows that the price is a suitable feature to use, since you can see correlations between a decrease in price with an increase in sales of the product. Related time series can be supplied to Amazon Forecast through a separate CSV file, containing the item SKU, timestamp, and related time series values (in this case, price).

Amazon Forecast supports aggregation methods, such as average and summation for target time series, but not for related time series. For example, it makes little sense to sum a daily price to a weekly price, and similarly for daily promotions.

Amazon Forecast can automatically incorporate weather and holiday information into a model by including built-in featurized datasets (see `SupplementaryFeature`). Weather information and holidays can significantly affect retail demand.



The sales of a particular item (in blue, left to the vertical red line)

Item metadata, also known as categorical variables, are other helpful features that can be input to Amazon Forecast (see Amazon Forecast: predicting time-series at scale for an example). The main difference between categorical variables and related time series is that categorical variables are static — they do not change over time. Common retail examples include the colors of items, categories of books, and binary indicators of whether a TV is a smart TV or not. This information can be picked up by deep learning algorithms in learning similarities between stock-keeping units (SKUs), assuming that similar SKUs have similar sales. Since this metadata does not have a time dependency, each row in the item metadata CSV file only consists of the item SKU and the corresponding category label or description.

Step 3: Create a predictor

A predictor can be created in two ways: running AutoML or manually selecting one of six built-in Amazon Forecast algorithms. When running AutoML, at the time of writing this document, Amazon Forecast automatically tests the six built-in algorithms and chooses the one with the lowest average quantile losses over the 10th, 50th (median), and 90th quantiles.

Amazon Forecast offers four local models:

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

Local models are forecasting methods that fit a single model to each individual time series (or specific item/dimension combination), and then use those models to extrapolate the time series into the future.

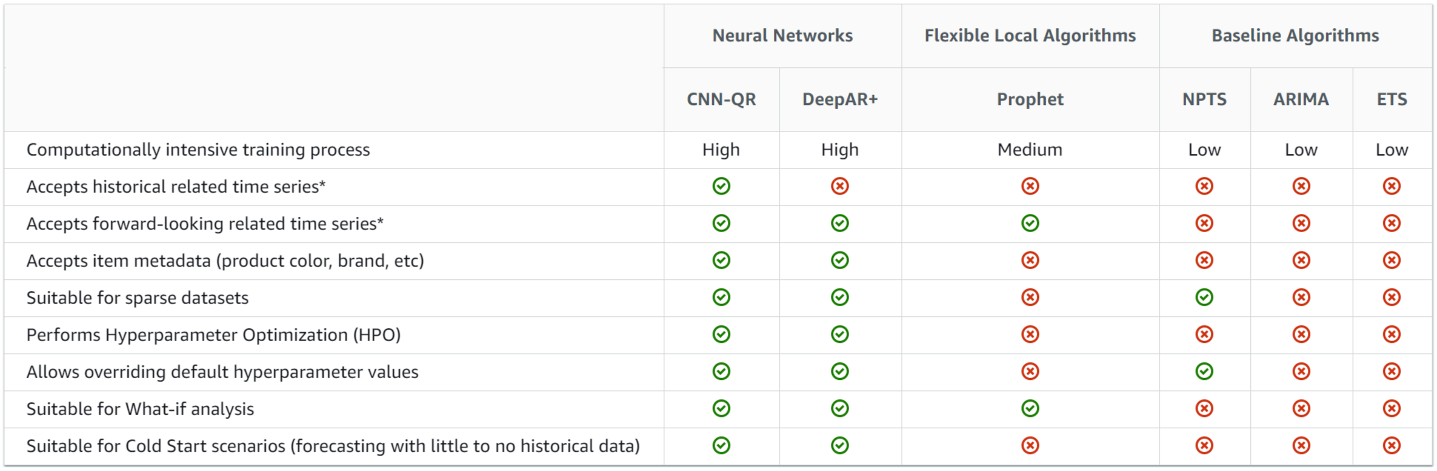
ARIMA and ETS are scalable versions of popular local models from the R forecast package. NPTS, a local method developed at Amazon, has a key difference when compared to the other local models. Unlike the simple seasonal forecasters, which provide point forecasts by repeating the last value or the value at an appropriate seasonality, NPTS produces probabilistic forecasts. NPTS uses a fixed-time index, where the previous index (T - 1) or the past season (T - tau) is the prediction for time step T. The algorithm randomly samples a time index (t) in the set {0, ..., T - 1} to generate a sample for the current time step T. NPTS is particularly effective for intermittent (sometimes also called sparse) time series with many zeros. Forecast also includes the Python implementation of Prophet, a Bayesian structural time series model.

Amazon Forecast offers two global deep learning algorithms:

- DeepAR+
- CNN-QR

Global models train a single model over the entire collection of time series in a data set. This is especially useful when there are similar time series across a set of cross-sectional units. For example, time series groupings of demand for different products, server loads, and requests for web pages.

In general, as the number of time series increases, the efficacy of CNN-QR and DeepAR+ increases. This is not always the case for the local models. The deep learning models can also be used to generate forecasts for new SKUs with little to no historical sales data. This is known as Cold Start forecasting.



	Neural Networks		Flexible Local Algorithms	Baseline Algorithms		
	CNN-QR	DeepAR+	Prophet	NPTS	ARIMA	ETS
Computationally intensive training process	High	High	Medium	Low	Low	Low
Accepts historical related time series*	✓	✗	✗	✗	✗	✗
Accepts forward-looking related time series*	✓	✓	✓	✗	✗	✗
Accepts item metadata (product color, brand, etc)	✓	✓	✗	✗	✗	✗
Suitable for sparse datasets	✓	✓	✗	✓	✗	✗
Performs Hyperparameter Optimization (HPO)	✓	✓	✗	✗	✗	✗
Allows overriding default hyperparameter values	✓	✓	✗	✓	✗	✗
Suitable for What-if analysis	✓	✓	✓	✗	✗	✗
Suitable for Cold Start scenarios (forecasting with little to no historical data)	✓	✓	✗	✗	✗	✗

Compare the algorithms available in Amazon Forecast

For more information on related time series, see Related Time Series.

Step 4: Evaluate predictors

A typical workflow in machine learning consists of training a set of models or combination of model(s) on a training set and assessing its accuracy on a holdout data set. This section discusses how to split historic data, and which metrics to use to evaluate models in time series forecasting. For forecasting, the backtesting technique is the main tool to assess forecast accuracy.

Backtesting

A proper evaluation and backtesting framework is among the most important factors in turning a machine learning application into a success. You can rely on successful backtests with your models to gain confidence over the models' future predictive power. Furthermore, you can tune models via hyper-parameter optimization (HPO), learn model combinations, and enable meta-learning and AutoML.

The time series forecasting characteristic *time* makes it different, in terms of evaluation and backtesting methodology, from other fields of applied machine learning. Usually in ML tasks, to assess the predictive error in a backtest, you split a dataset by items. For example, for cross-validation in image-related tasks, you train on some percentage of the pictures, and then use other parts for testing and validation. In forecasting, you need to split primarily by time (and to a lesser degree by items) to ensure that you do not leak information from the training set into the test or validation set, and that you simulate the production case as closely as possible.

The split by time must be done carefully because you do not want to choose a single point in time, but multiple points. Otherwise, the accuracy is too dependent on the forecast start date, as defined by the splitting point. A *rolling forecast evaluation*, where you do a series of splits over multiple time points and output the average result leads to more robust and reliable backtest results. The following figure illustrates four different backtest splits.

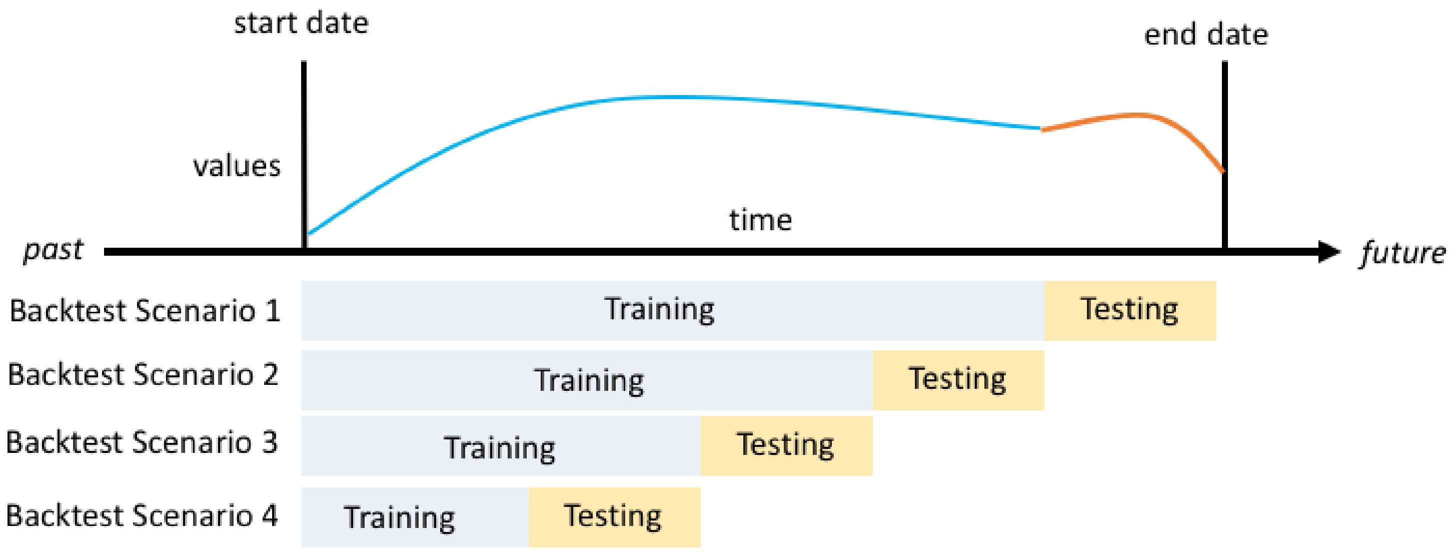


Illustration of four different backtesting scenarios with increasing training set size, but constant size of testing

In the preceding figure, all backtesting scenarios have data available during its entirety to be able to evaluate forecasted values against actuals.

The reason that multiple backtest windows are needed is that most time series in the real world are normally non-stationary. The e-commerce business in the case study is based in North America and much of its product demand is driven by the Q4 peak, with particular peaks around Thanksgiving and before Christmas. In the Q4 shopping season, the variability of the time series is higher than the rest of the year. By having multiple backtest windows, you can evaluate forecasting models in a more balanced setting.

For each backtest scenario, the following figure shows the basic elements in the terminology of Amazon Forecast. Amazon Forecast automatically splits the data into the train and test datasets. Amazon Forecast decides how to split the input data by using the `BackTestWindowOffset` parameter that is either specified as a parameter in the `create_predictor` API or uses its default value of `ForecastHorizon`.

In the following figure, you see the more general former case when the `BackTestWindowOffset` and `ForecastHorizon` parameters are not equal. The `BackTestWindowOffset` parameter defines a virtual forecast start date, shown as the dashed vertical line in the following figure. It can be used to answer the following hypothetical question: If the model is deployed on this day, what would the forecast be? The `ForecastHorizon` defines the number of time steps from the virtual forecast start date to predict.

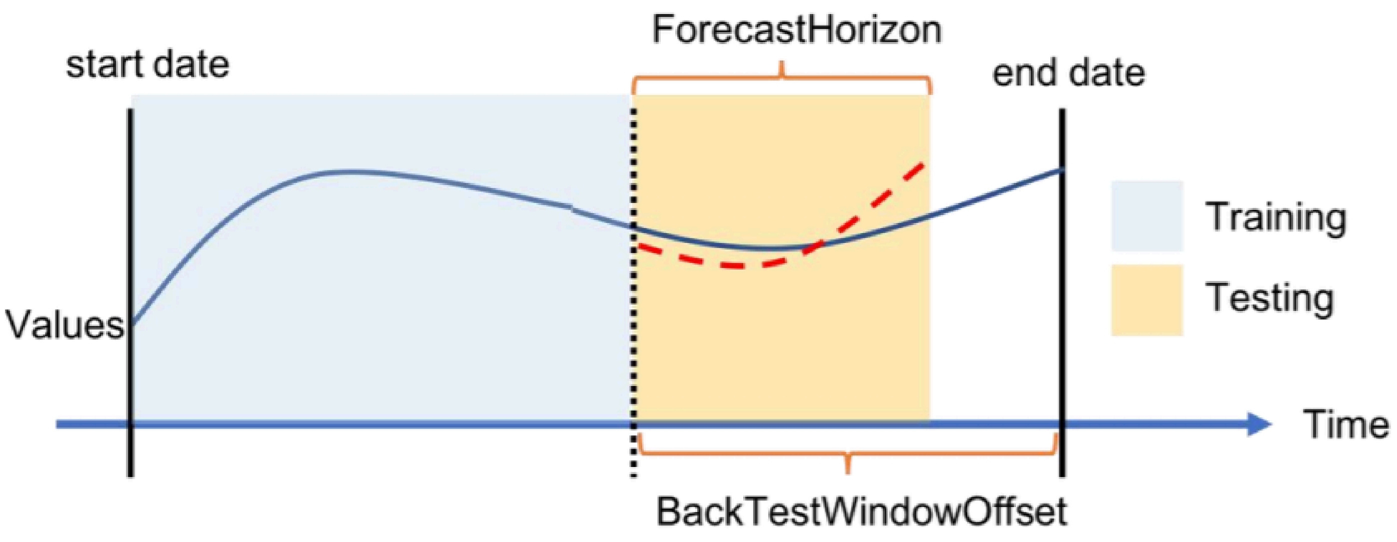


Illustration of a single backtest scenario and its configuration in Amazon Forecast

Amazon Forecast can export the forecasted values and accuracy metrics generated during backtesting. The exported data can be used to evaluate specific items at specific time points and quantiles.

Prediction quantiles and accuracy metrics

Prediction quantiles can provide an upper and lower bound for forecasts. For example, using the forecast types 0.1 (P10), 0.5 (P50), and 0.9 (P90) provides a range of values known as an 80% confidence interval around the P50 forecast. By generating predictions at P10, P50, and P90, you can expect the true value to fall between those bounds 80% of the time.

This paper further discusses quantiles in Step 5.

Amazon Forecast uses the Weighted Quantile Loss (wQL), Root Mean Square Error (RMSE), and Weighted Absolute Percentage Error (WAPE) accuracy metrics to evaluate predictors during backtesting.

Weighted Quantile Loss (wQL)

The Weighted Quantile Loss (wQL) error metric measures the accuracy of a model's forecast at a specified quantile. It is particularly useful when there are different costs for underpredicting and overpredicting. Setting the weight (τ) of the wQL function automatically incorporates differing penalties for underpredicting and overpredicting.

$$\mathrm{wQL}[\tau] = 2\frac{\sum_{i,t}[\tau \max(y_{i,t} - q_{i,t}^{(\tau)}, 0) + (1-\tau)\max(q_{i,t}^{(\tau)} - y_{i,t}, 0)]}{\sum_{i,t}|y_{i,t}|}$$

wQL function

Where:

- τ — A quantile in the set {0.01, 0.02, ..., 0.99}
- qi,t(τ) — The τ-quantile that the model predicts.
- yi,t — The observed value at point (i,t)

Weighted Absolute Percentage Error (WAPE)

The Weighted Absolute Percentage Error (WAPE) is a commonly used metric to measure model accuracy. It measures the overall deviation of forecasted values from observed values.

$$\mathrm{WAPE} = \frac{\sum_{i,t}|y_{i,t} - \hat{y}_{i,t}|}{\sum_{i,t}|y_{i,t}|}$$

WAPE

Where:

- yi,t - The observed value at point (i,t)
- ŷi,t - The predicted value at point (i,t)

Forecast uses the mean forecast as the predicted value, ŷi,t.

Root Mean Square Error (RMSE)

$$\mathrm{RMSE} = \sqrt{\frac{1}{nT}\sum_{i,t}(\hat{y}_{i,t} - y_{i,t})^2}$$

Root Mean Square Error (RMSE) is a commonly used metric to measure model accuracy. Like WAPE, it measures the overall deviation of estimates from observed values.

Where:

- yi,t - The observed value at point (i,t)
- ŷi,t - The predicted value at point (i,t)
- nT - The number of data points in a testing set

Forecast uses the mean forecast as the predicted value, ŷi,t. When calculating predictor metrics, nT is the number of data points in a backtest window.

Problems with WAPE and RMSE

In most cases, the point forecasts that can be generated internally or from other forecasting tools should match the p50 quantile or mean forecasts. For both WAPE and RMSE, Amazon Forecast uses the mean forecast to represent the predicted value (yhat).

For tau = 0.5 in the wQL[tau] equation, both weights are equal, and the wQL[0.5] reduces to the commonly used Weighted Absolute Percentage Error (WAPE) for point forecasts:

$$\mathrm{wQL}[0.5] = 2\frac{\sum_{i,t} 0.5[\max(y_{i,t} - q_{i,t}^{(0.5)}, 0) + \max(q_{i,t}^{(0.5)} - y_{i,t}), 0]}{\sum_{i,t} |y_{i,t}|} = \frac{\sum_{i,t} |y_{i,t} - q_{i,t}^{(0.5)}|}{\sum_{i,t} |y_{i,t}|}$$

where yhat = q(0.5) is the compute forecast. A scaling factor of 2 is used in the wQL formula to cancel the 0.5 factor to obtain the exact WAPE[median] expression.

Note that the above definition of WAPE differs from a common interpretation for Mean Absolute Percentage Error (MAPE). The difference is in the denominator. The way WAPE is defined above avoids the problem of division by 0, a commonly occurring problem in real-world scenarios such as the e-commerce business in the case study, which will often sell 0 units of a given SKU on a given day.

Unlike with the weighted quantile loss metric for tau not equal to 0.5, the inherent bias in each quantile cannot be captured by a calculation like the WAPE, where the weights are equal. Other disadvantages of the WAPE include that it is not symmetric, has an over inflation of percentage errors for small numbers, and is just a point-wise metric.

The RMSE is the square of the error term in the WAPE and a common error metric in other ML applications. The RMSE metric favors a model, where the individual errors are of consistent magnitude, as large variations in error will increase the RMSE over-proportionally. Due to the squared error, a few poorly predicted values in an otherwise good forecast can increase the RMSE. Also, because of the squared terms, smaller error terms have less weight in the RMSE than in the WAPE.

Accuracy metrics allow for a quantitative assessment of the forecasts. In particular for large-scale comparisons (is method A better than method B overall), these are crucial. However, complementing this with visuals for individual SKUs is often important.

Step 5: Generate and use forecasts for decision making

Once you have a model that meets the accuracy threshold required for your specific use case (as determined via backtesting), the final step involves deploying the model and generating forecasts. To deploy a model in Amazon Forecast, you must run the Create_Forecast API. This action hosts a model created by training on the entire historical dataset (unlike `Create_Predictor`, which splits the data into a train and test set). The model predictions generated over the forecast horizon can then be consumed in two ways:

- You can query the forecasts for a particular item (by specifying the item or combination of item/dimension) using the `Query_Forecast` API from the AWS CLI or directly via the AWS Management Console.
- You can generate the forecasts for all combination of items and dimensions across all quantiles using the `Create_Forecast_Export_Job` API. This API generates a CSV file that is securely stored in an Amazon Simple Storage Service (Amazon S3) location of your choice. You can then use data from the CSV file and plug them into your downstream systems used for decision making. For example, your existing supply chain systems can ingest the output from Amazon Forecast directly to help inform decision making around manufacturing of specific SKUs.

Probabilistic forecasts

Amazon Forecast can generate forecasts at different quantiles, which is particularly useful when the costs of under and over predicting are different. Similar to the predictor training stage, probabilistic forecasts can be generated for quantiles between p1 and p99.

By default, Amazon Forecast generates forecasts at the same quantiles used during predictor training. If quantiles are not specified during predictor training, forecasts will generated at p10, p50, and p90 by default.

For the p10 forecast, the true value is expected to be lower than the predicted value 10% of the time, and the wQL[0.1] metric can be used to assess its accuracy. This means the P10 forecast is an under-forecast 90% of the time, and if used to stock inventory, 90% of the time the item would be sold out. The P10 forecast could be useful when there is not a lot of storage space, or the cost of the invested capital is high.

Note

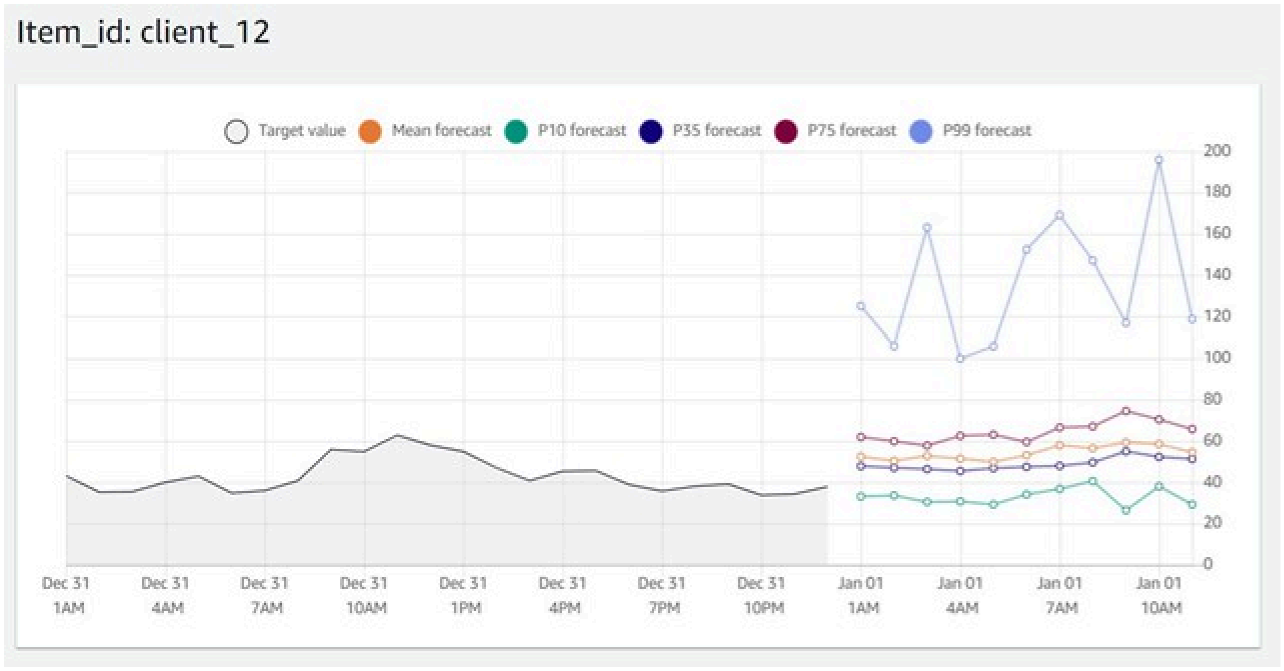
The formal definition of a quantile forecast is Pr(actual value <= forecast at quantile q) = q. Technically a quantile is a percentile/100. Statisticians tend to say ”P90 quantile-level“, since that is easier to say than "“quantile 0.9”. For example, a P90 quantile-level forecast means the actual value can be expected to be less than the forecast 90% of the time. Specifically if at time=t1 and quantile-level=0.9, the predicted value = 30, that means the actual value at time= t1, if you had 1,000 simulations, would be expected to be less than 30 for 900 simulations, and for 100 simulations, the actual value is expected to be over 30.

On the other hand, the P90 forecast is an over-forecast 90% of the time, and it is useful when the opportunity cost of not selling an item is extremely high, or the cost of invested capital is low. For a grocery store, the P90 forecast might be used for something like milk or toilet paper, where the store never wants to run out and doesn’t mind always having some remaining on the shelves.

For the p50 forecast (often also called the median forecast), the true value is expected to be lower than the predicted value 50% of the time, and the wQL[0.5] metric can be used to assess its accuracy. When being overstocked is not too concerning, and there is a moderate amount of demand for a given item, the p50 quantile forecast can be useful.

Visualization

Amazon Forecast allows for the plotting of forecasts natively in the AWS Management Console. Additionally, you can take advantage of the complete Python data science stack (see Amazon Forecast Examples). Amazon Forecast allows for the exporting of forecasts as a CSV file via the `ExportForecastJob` API, which allows users to visualize the forecast in their analytic tool of choice.



Visualization provided in the Amazon Forecast console at different quantiles

Summary of forecasting workflow and APIs

The following table matches each step of the forecasting workflow with the respective Amazon Forecast API.

Table 1: Forecasting steps and Amazon Forecast APIs

Step	API	API functions
Step 1: Collect and aggregate data Step 2: Prepare data	`Create_Dataset_Group` , `Create_Dataset` , `Create_Dataset_Import_Job`	1. Establish the high level domain (retail, metrics, and so on) for the problem. 2. Define the schema for the different datasets (target, related, item-metadata). 3. Import data from Amazon S3 into Amazon Forecast.
Step 3: Create a predictor Step 4: Evaluate predictors	`Create_Predictor`	1. Performs the ETL. 2. Splits data into train/test sets and trains the model. 3. Optionally, use `Create_predictor_backtest_Export_job` to export backtest results to CSV for calculating item-level metrics.
Step 5: Generate and use forecasts for decision making	`Create_Forecast`	1. Trains/hosts the model. 2. Generates predictions over the forecast horizon for a specific quantile of interest (for example, any integer between 1 to 99 including mean).

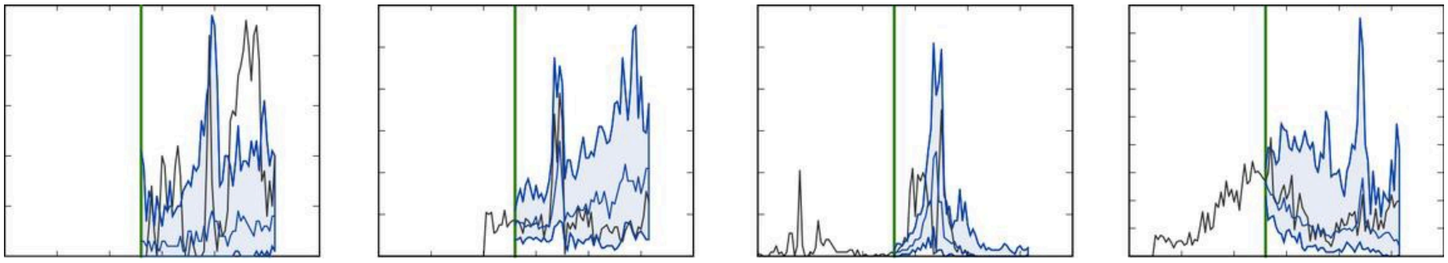
Step	API	API functions
	`Query_Forecast . Create_Forecast_Export_Job`	Enables you to consume the forecasts created by `Create_Forecast`

Using Amazon Forecast for common scenarios

You can also conduct what-if analysis by generating different forecasts based on changes in external variables (for example, prices or promotions). For example, in the e-commerce case study example, you can create different forecasts based on promotions that you may be planning. You can forecast demand for a product at 10% discount and then at 20% discount to understand the quantity of product that you would need to stock to meet demand. This can be achieved by setting up unique dataset groups and updating the related time series in each, based on the scenario of interest.

Additionally, you can also generate forecasts for items with no prior history (sometimes called the cold start problem). This approach requires creating a predictor using DeepAR+ or CNN-QR along with metadata (such as an item metadata dataset) to generate forecasts for the new item.

The following figure shows examples of four different SKUs as they appear in real-world operational forecasting problems.



Examples of four different SKUs as they appear in real-world operational forecasting problems.

In the preceding image, observed actuals are to the left off the vertical line, and the forecasts in blue are to the right to the vertical line, compared with the actuals in black. Note that each individual's SKU's history, left of the vertical line, is not indicative of the evolution of it on the right of the green line.

Implementing Forecast into production

Having completed the end-to-end Amazon Forecast workflow, it is critical to identify key differences between the `Create_Predictor` and `Create_Forecast` APIs and when each should be used.

The former is used primarily during proof of concept to evaluate model accuracy/metrics, whereas the latter is used to generate forecasts in a production environment.

Once in production, `Create_Predictor` does not need to be run every time a forecast needs to be generated but only when the model needs to be retrained due to changes in data or as part of a pre-established cadence (for example, bi-weekly or monthly). As the datasets are updated with new data, only `Create_Forecast` needs to be run to generate forecasts for the new forecast horizon.

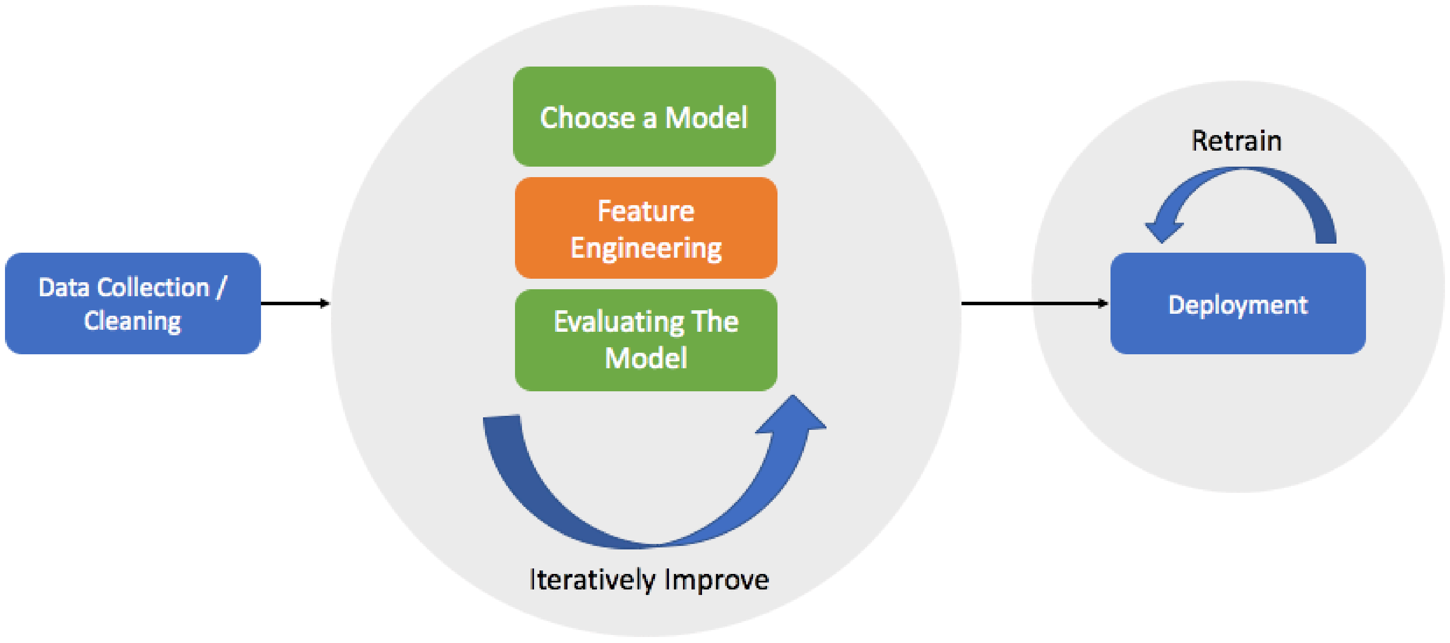
In production, you also need to automate your dataset imports and forecast operations in order to generate new predictions on a rolling basis. Today, this can be achieved by setting up cron jobs using a combination of Amazon CloudWatch Events logs, AWS Step Functions, and AWS Lambda functions. The cron job setup in turn automates Amazon Forecast API calls for import/re-train or forecast generation. Finally, it is key to manage your resources and delete them at regular intervals so that you do not exceed system limits prescribed by the service. Refer to this blog post including Amazon Redshift to learn more about setting up scheduled jobs.

Conclusion

Januschowski and Kolassa (2019) provide a classification of forecasting problems aligned with the decisions businesses need to take, including *strategic*, *tactical*, and *operational* decisions. Each decision level has corresponding forecasting tasks.

Operational and tactical forecasting problems are characterized as containing large amounts of data, and typically require a high degree of automation. Different forecasting methods cater for these problems. Local forecasting methods typically work well for strategic forecasting problems, deep learning-based methods for operational forecasting problems, and for the in-between, a bit of experimentation may be necessary. Although this paper discussed operational forecasting problems, Amazon Forecast is not opinionated about the models it offers and includes models that serve strategical as well as operational and tactical forecasting problems.

The operational forecasting problem solving process can be broken into basic steps starting from data collection and preparation to model building and deployment. In general, it is most useful to regard this as an iterative rather than linear process. For example, as models and use cases are better understood, it may make sense to go back to the data collection phase. Model development is also highly iterative in itself.



A simplified development process of bringing a forecasting model to production.

Contributors

Contributors to this document include:

- Yuyang Wang, Senior Machine Learning Scientist, AI Vertical Services
- Danielle Robinson, Applied Scientist, AI Vertical Services
- Tim Januschowski, Manager, ML Applied Science
- Namita Das, Senior Product Manager, AI Vertical Services
- Christy Bergman, Senior AI/ML Specialist Solutions Architect
- Kris Tonthat, Technical Writer, AI/ML Documentation

Further reading

For additional information on time series forecasting and deep learning methods, see:

- Amazon Forecast Documentation
- Amazon Forecast general availability blog
- Now available in Amazon SageMaker: DeepAR algorithm for more accurate
- Amazon SageMaker DeepAR now supports missing values, categorical and time series features, and generalized frequencies
- Amazon Forecast can now use Convolutional Neural Networks (CNNs) to train forecasting models up to 2X faster with up to 30% higher accuracy
- Amazon Forecast now supports accuracy measurements for individual items
- Measuring forecast model accuracy to optimize your business objectives with Amazon Forecast
- Amazon Forecast Weather Index – automatically include local weather to increase your forecasting model accuracy
- Scientific papers on time series forecasting models
- Amazon Forecast samples GitHub page
- AWS Architecture Center

Appendix A: FAQs

Q: How can I get started with Amazon Forecast?

1. First, you'll need an AWS account.
2. Next, open the Forecast service in the AWS Management Console, create a dataset group, and import a `.csv` file into the target time series dataset (required). The minimum data required to get started is historical data for the quantity you want to predict, such as electricity per timestamp per household.
3. Finally, create a model by running CreatePredictor and generating results by running CreateForecast. See the Getting Started documentation page for more details.

Also, see the GitHub Introduction and Best Practices guide.

Q: Is Amazon Forecast a good fit for me?

Not all machine learning problems are forecasting problems. The first question to ask is, "Does my business problem include time series in its statement?" For example, do you need a particular value only at a particular time and date in the future? Forecasting is not a good fit for general, static (where the particular date/time does not matter) problems, such as fraud detection or recommending movie titles to users. There are much quicker solutions to static problems.

In addition to having time series data, the data itself should be "dense," and with long histories. This is summarized in the following table:

Table 2 — Criteria and Amazon Forecast algorithm classes

Criteria	Amazon Forecast algorithm class
Large dataset with up to five million time series with similar underlying patters + seasonal effects + related data. Each time series should have a long history, ideally more than two years if trying to capture annual events, and each time series more than 300, ideally at least 1K data points.	Amazon Forecast proprietary deep learning DeepAR+, CNN-QR

Criteria	Amazon Forecast algorithm class
Small dataset with 1-100s of time series, where the majority of time series have more than 300 data points + seasonal effects + related data.	Prophet
Small dataset with 1-10s of time series, where the majority of time series have more than 300 data points + seasonal effects.	ETS, ARIMA
Intermittent (sparse containing many 0s) with 1-10s of time series, where the majority of time series have more than 300 data points.	Amazon Forecast proprietary NPTS
Small dataset (regular or sparse) with 1-10s of time series, where the majority of time series have fewer than 300 data points.	The data is too small for Amazon Forecast. Try ETS in Excel or the traditional statistical models ARIMA and Prophet instead.

The best practice is to train with AutoML mode in your Predictor, the very first time on your data. AutoML will automatically run through all the algorithms, (the DL algorithms run with HPO turned on), to learn which algorithm works best on your data.

Q: How do I think about missing data? When is it too much to generate reasonable forecasts?

It may be the case that there are problems in the recording of data or that the aggregation level of the data is too low or too high. The general rule is, Forecast length cannot be longer than 1/3 of the training data.

Besides the amount of missing data, another consideration is imputation of missing data. You can convert all 0s to nulls and let Amazon Forecast do the heavy lifting of automatically imputing missing values. Amazon Forecast will automatically detect whether the missing values occur due to new product introduction (cold-starts) or end-of-life products. You can use several missing value logics including value, median, minimum, maximum, zero, mean, and nan (target time series only. See the documentation for null-filling syntax.

- "frontfill" — (TTS only) refers to new or cold-start items and how you want to treat nulls before the item begins to have any history

- "middlefill" — Refers to nulls in the middle of time series values
- "backfill" — Refers to end-of-life items and how you want to treat nulls after an item has stopped selling
- "futurefill" — (RTS only) refers to nulls that occur after the end of training data

Q: My input historic data does not have negative values, but I see negative values in the demand forecasts? Why does this happen? What can I do to avoid this?

For all models other than NPTS (trained on the non-negative data) and DeepAR (with negative-binomial likelihood function), there is no guarantee to generate positive numbers. The solution is to change to one of the aforementioned models, or truncate forecast values to non-negative values.

Q. Why do accuracy metrics differ at quantiles? Shouldn't the error be the same since the model is the same?

See the Weighted Quantile Loss (wQL) for more explanation how weighting depends on the quantile.

Imagine you have all the forecasts at three different quantiles: p10, p50, p90. The three predictions themselves are random variables. The accuracies are calculated separately between actuals and forecasts at each quantile level. You might see a table of "wQL", weighted quantile losses, such as below. The wQL values have no deterministic relationship to each other. (Recall loss means error, so it is unordered; quantile forecasts, however, are ordered). So, there is no reason why p90 wQL must be larger than p50 wQL, for example.

Table 3 — Example forecast quantiles

	A	**B**	**C**
1	P10 wQL	P50 wQL	P90 wQL
2	0.18647	0.50879	0.30428

Q: How can I improve the forecasting accuracy?

Forecast accuracy depends on whether the right data is available in the right amount of quantity and quality. If accuracy is unsatisfactory, it may make sense to understand how predictable the problem is (or how random/noisy/stationary the data is). Other factors to consider include evaluating different models, hyperparameter settings, and incorporating additional features by using the related time series and item metadata datasets. For specific suggestions, see this Best Practices document on GitHub.

Q: I have an algorithm that works very well for my use case, and it is not offered in Amazon Forecast. What should I do?

The Amazon Forecast team will be glad to help you with this use case. Contact the service team of Amazon Forecast by emailing `<amazonforecast-poc@amazon.com>`.

Appendix B: References

Januschowski, Tim and Kolassa, Stephan. A classification of business forecasting problems. Foresight: The International Journal of Applied Forecasting. 2019

Salinas, David, Flunkert, Valentin, Gasthaus, Jan and Januschowski, Tim. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. International Journal of Forecasting. 2019

Gasthaus, Jan, Benidis, Konstantinos, Wang, Yuyang, Rangapuram, Syama Sundar, Salinas, David, Flunkert, Valentin and Januschowski, Tim. {Probabilistic Forecasting with Spline Quantile Function RNNs. The 22nd International Conference on Artificial Intelligence and Statistics. 2019

Januschowski, Tim, Gasthaus, Jan, Wang, Yuyang, Salinas, David, Flunkert, Valentin, Bohlke-Schneider, Michael and Callot, Laurent. Criteria for classifying forecasting methods. International Journal of Forecasting. 2019 (Sign in required)

Januschowski, Tim, Gasthaus, Jan, Wang, Yuyang, Rangapuram, Syama and Callot, Laurent. Deep Learning for Forecasting. Foresight: The International Journal of Applied Forecasting. 2018

Januschowski, Tim, Gasthaus, Jan, Wang, Yuyang, Rangapuram, Syama Sundar and Callot, Laurent. Deep Learning for Forecasting: Current Trends and Challenges. Foresight: The International Journal of Applied Forecasting. 2018

Bose, Joos-Hendrik, Flunkert, Valentin, Gasthaus, Jan, Januschowski, Tim, Lange, Dustin, Salinas, David, Schelter, Sebastian, Seeger, Matthias and Wang, Yuyang. Probabilistic demand forecasting at scale. Proceedings of the VLDB Endowment. 2017

Document history

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Whitepaper updated	Updates.	September 1, 2021
Initial publication	Whitepaper first published.	February 4, 2020

Note

To subscribe to RSS updates, you must have an RSS plug-in enabled for the browser that you are using.

Notices

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.