Improving Forecast Accuracy with Machine Learning

Implementation Guide

Improving Forecast Accuracy with Machine Learning: Implementation Guide

Table of Contents

Deploy a solution designed to help organizations generate accurate forecasts from diverse datasets

July 2020 (last update (p. 43): August 2021)

The Improving Forecast Accuracy with Machine Learning solution helps organizations generate accurate forecasts from diverse datasets. You can use this solution to configure and upload your datasets, generate forecasts, and visualize your results. This solution reduces overhead costs for organizations developing new forecasts and optimizes their existing forecasting processes by standardizing tasks and saving developer time.

This implementation guide provides procedures for setting up visualizations using either an Amazon QuickSight analysis or a Jupyter Notebook Instance, provides infrastructure and configuration information for planning and deploying the solution in the AWS Cloud, and describes architectural considerations. It also includes links to two AWS CloudFormation templates that launch and configure the AWS services required to deploy this solution using AWS best practices for security and availability.

Overview

Forecasting helps organizations develop informed business strategies based on past data points. Forecasting software helps organizations identify patterns and provide estimates that can optimize business operations. For example, in retail forecasting, forecasts are used to estimate future sales, predict when and how many units must be reordered, and reduce inventory holding costs. Key metrics such as revenue, sales, and cash flow are also predicted.

Generating and leveraging accurate forecasts can be time-consuming and resource intensive for data science and development teams. Data scientists and developers use an array of tools, ranging from manually filled spreadsheets to complex forecasting software, in an attempt to perform time-series-demand forecasting. The difference between simply generating a good forecast and using the most optimal forecast configuration can represent millions of dollars of loss in certain scenarios. For example, a business might want to minimize the cost of overstocking due to excessive carrying costs, a high cost of capital, and/or perishable goods. Another business might want to minimize understocking to ensure customer demand is met, or to avoid lost revenue due to lost sales.

To customize forecasts to suit your requirements, Amazon Forecast outputs probabilistic predictions at three default quantiles (shown in Figure 1) to address the sensitivities of each business to over and understocking:

- Businesses avoiding overstocking can use the p10 forecast, where the true future demand value is expected to be lower than the predicted value only 10% of the time.
- Businesses more sensitive to missing customer demand can use the p90 forecast, where the true value is expected to be lower than the predicted value 90% of the time.
- Businesses aiming to retain compatibility with their legacy tools, or with equal sensitivity to over and understocking, can use a point forecast (p50).

While the three existing quantiles supported by Amazon Forecast are useful, they can also be limiting. The fixed quantiles may not always meet specific use case requirements. For example, if meeting customer demand is imperative at all costs, a p99 forecast may be more useful than p90. Fortunately, Amazon Forecast quantiles can be customized to meet any customer requirement.

Amazon Forecast also allows you to evaluate predictor accuracy on a per-item basis to better understand the forecasting model's performance for the items that most impact your business.

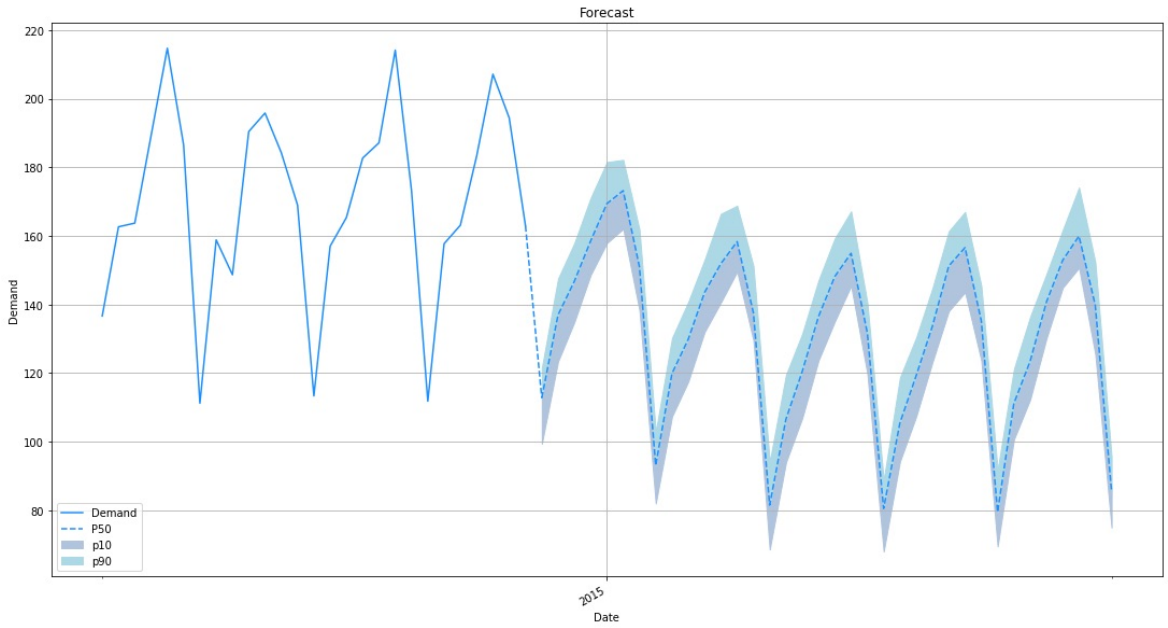

Figure 1: Sample forecast output

This solution supports multiple forecasts and per-forecast parameter configuration in order to reduce the repetitive task of generating multiple forecasts. For more information on benchmarking multiple Forecasts, refer to *Forecast experimentation* (p. 32). This solution automates all aspects of the configuration of Amazon Forecast—allowing developers and data scientists to focus on the accuracy of their forecasts.

To better capture and alert users of data quality issues, this solution supports a configurable alert function which is deployed through Amazon Simple Notification Service (Amazon SNS). This notifies the user on success and failure of the automated forecasting job, reducing the need for users to monitor their forecast workflow.

This solution allows you to provide related time-series data that may improve your accuracy. The sample in this solution uses item price as an additional input data set. For more details, refer to *Overriding forecast defaults* (p. 30).

This solution also allows you to visualize your historical demand and forecast output alongside item level accuracy metrics in Amazon QuickSight or a Jupyter Notebook.

Cost

You are responsible for the cost of the AWS services used while running this solution. At the date of publication, the cost for running this solution with the default settings in the US East (N. Virginia) Region is approximately $14.50 per 100 forecasts generated for one item, or approximately **$76.62 per month, when deployed with a ml.t2.medium Amazon SageMaker notebook instance**. Prices are subject to change. For full details, see the pricing webpage for each AWS serviceused in this solution.

Cost per 100 forecasts for one item

The solution uses the following resources that are billed per use.

AWS Service	Quantity	Total Cost
AWS Lambda	*125 billed seconds of AWS Lambda* *128 MB per Lambda function*	$0.05
AWS Step Functions	*Approximately 6,000 state transitions*	$0.15
Amazon Forecast	*100 forecasts* *4 Predictor creations (2 hours each)*	$9.60
Amazon Athena	*500 GB of data scanned*	$2.50
AWS Glue	*5 DPU-hours*	$2.20
TOTAL		$14.50

Cost per month

This solution uses the following resources that are billed on a monthly basis. Note that the Amazon SageMaker notebook instance and Amazon QuickSight dashboard are optional.

AWS Service	Quantity	Total Cost
Amazon SageMaker	*1 x ml.t2.medium notebook instance*	$33.46
Amazon Forecast	*1.5 GB Data storage*	$0.13
Amazon S3	*1.5 GB Data storage*	$0.03
Amazon QuickSight	1x Author 5 x Readers ($0.30/session or $5.00/month max)	$18.00 $25.00 (max)
TOTAL		$76.62

Architecture overview

Deploying this solution **with the default parameters** builds the following environment in the AWS Cloud.

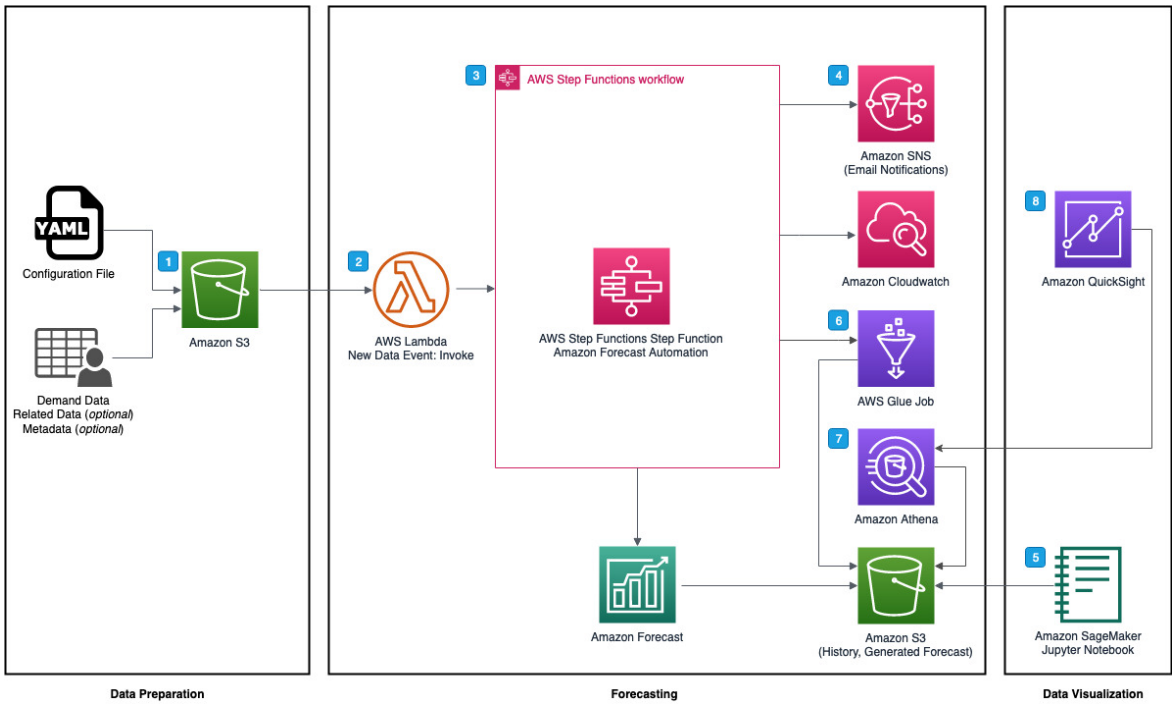

Figure 2: Improving Forecast Accuracy with Machine Learning solution architecture

The AWS CloudFormation templates deploy the resources required to automate your Amazon Forecast usage and deployments. Based on the capabilities of the solution, the architecture is divided into three parts: Data Preparation, Forecasting, and Data Visualization. The template includes the following components:

1. An Amazon Simple Storage Service (Amazon S3) bucket for Amazon Forecast configuration where you specify configuration settings for your dataset groups, datasets predictors, and forecasts, as well as the datasets themselves.
2. An Amazon S3 event notification that triggers when new datasets are uploaded to the related Amazon S3 bucket.
3. An Improving Forecast Accuracy with Machine Learning AWS Step Functions state machine. This combines a series of AWS Lambda functions that build, train, and deploy your Machine Learning (ML) models in Amazon Forecast. All AWS Step Functions log to Amazon CloudWatch.
4. An Amazon Simple Notification Service (Amazon SNS) topic and email subscription that notify administrative users with the results of the AWS Step Functions.
5. An Amazon SageMaker Notebook Instance that data scientists and developers can use to prepare and process data, and evaluate Forecast output.
6. An AWS Glue Job combines raw forecast input data, metadata, predictor backtest exports and forecast exports into an aggregated view of your forecasts.
7. Amazon Athena can be used to query your forecast output using standard SQL queries.
8. Amazon QuickSight analyses can be created on a per-forecast basis to provide users with forecast output visualization across hierarchies and categories of forecasted items, as well as item level accuracy metrics. Dashboards can be created from these analyses and shared within your organization.

> **Note**
> AWS CloudFormation resources are created from AWS Cloud Development Kit (AWS CDK) components.

Solution components

AWS Step Functions state machine

The AWS Step Functions state machine controls the architecture automating Amazon Forecast.

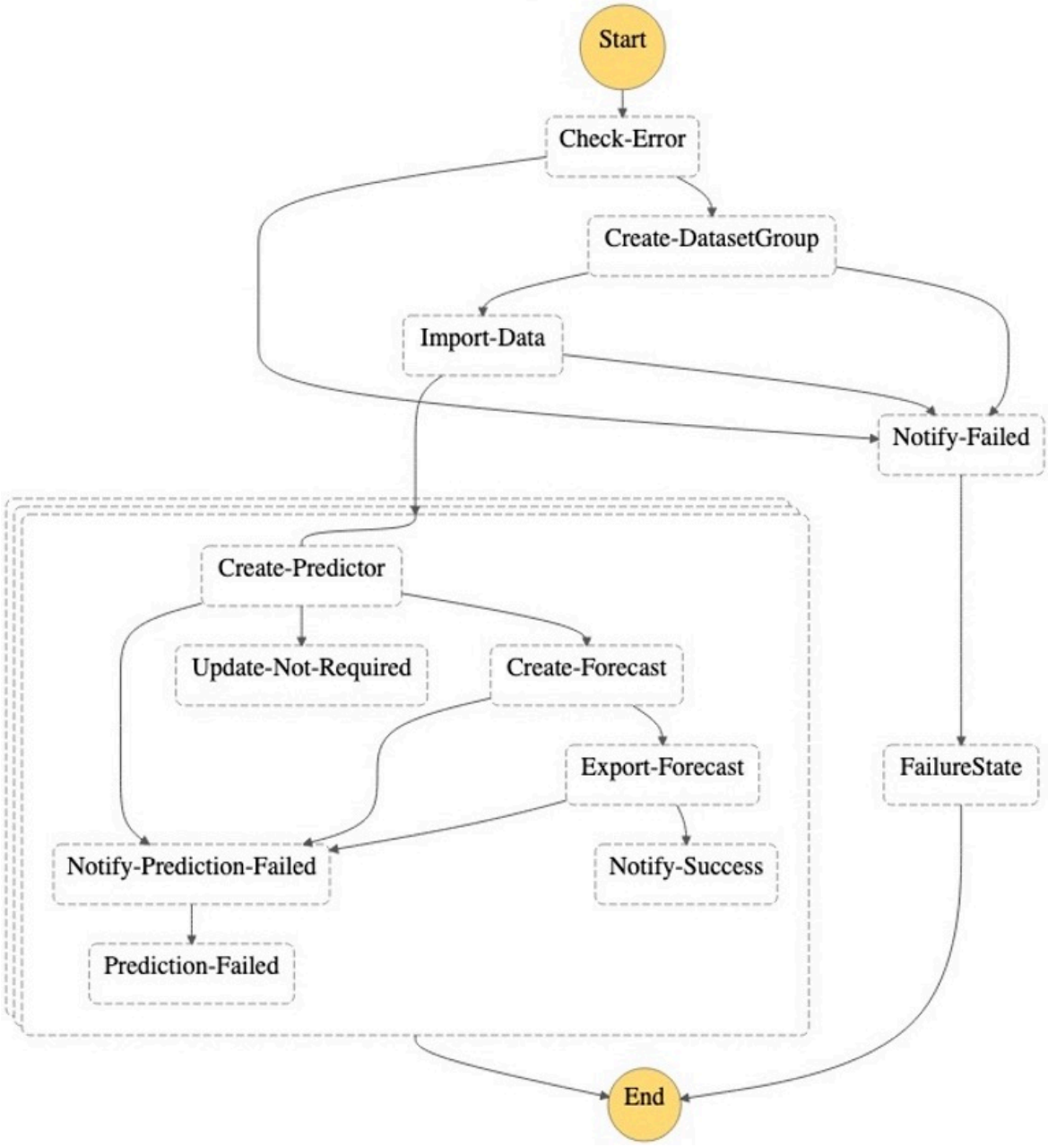

Figure 3: Improving Forecast Accuracy with Machine Learning state machine workflow

The Step Functions workflow contains the following Lambda functions:

- **Check-Error** – Validates whether the configuration file `forecast-defaults.yaml` is present and well formatted. This step abandons the pipeline process if error(s) are found, otherwise, it continues the automation.
- **Create-DatasetGroup** – Creates the Amazon Forecast dataset group and datasets based on the configuration in `forecast-defaults.yaml`. For example, if the Amazon S3 event that triggered the pipeline was the result of an upload of a .csv file named `s3://<data_bucket_name>/train/retail_sales.csv`, a dataset group named "retail_sales" is created.
- **Import-Data** – Imports the data defined by the specification used in **Create-Dataset**. To determine if your data must be imported, the solution checks whether data is present, and if its specification has changed, or if the .csv being imported has a different number of rows.
- **Create-Predictor** – Creates a predictor based on the specification in the `forecast-defaults.yaml` configuration file. This operation might take some time since the forecasting ML models are trained using your datasets. You can specify a `MaxAge` parameter with the number of seconds allowed to pass between dataset import jobs when the previously generated predictor is used. This is useful if your data changes frequently and forecasts must be generated at a different interval than your predictor.
- **Update-Not-Required** – If the create predictor step detects that an update to the predictor and forecast is not required (for example, if the data has not changed, or if the predictor is not using the most recent dataset), the task is terminated with a successful result by this state.
- **Create-Forecast** – Creates and exports a forecast based on the specification in the `forecast-defaults.yaml` configuration file.
- **Create-Forecast-Export / Create-Predictor-Backtest-Export** – Exports a forecast and its predictor backtest data to S3. The forecast is exported to `s3://`*`<data_bucket_name>`*`/exports/export_`*`<dataset_group_name>_<YYYY_mm_dd_HH_MM_SS>`* where the timestamp refers to the most recent modification time of the datasets the forecast was generated with, and the predictor backtest is exported to `s3://<data_bucket_name>/exports/export_<dataset_group_name>_<YYYY_mm_dd_HH_MM_SS>` where the timestamp refers to the most recent predictor creation time.
- **Create-Glue-Table-Name** – Creates a name for the table in glue to store the aggregated forecast input and export data. These are formatted as *`<dataset_group_name>_<YYYY_mm_dd_HH_MM_SS>`* where the timestamp was the time this step is invoked.
- **Forecast-ETL** – Invokes the AWS Glue Job to aggregate forecast input and export data. The table created will appear in the AWS Glue data catalog of the deployed solution, and the aggregated data will be available in glueparquet format under *`s3://<data_bucket_name>/<output>/<glue_table_name>`*.
- **Create-QuickSight-Analysis** – Creates a QuickSight analysis pointing to the data aggregated and exported by the preceding step.
- **Notify-Success** – Sends a success notification to the user (via their Amazon SNS subscription).
- **Notify-Failed / Notify-Prediction-Failed** – Sends a failure notification to the user (via their Amazon SNS subscription). Users can correct their datasets and retry the pipeline.
- **SuccessState / FailureState / Prediction-Failed** – Represents success and failure end states.

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared model reduces your operational burden because AWS operates, manages, and controls the components including the host operating system, the virtualization layer, and the physical security of the facilities in which the services operate. For more information about AWS security, visit the AWS Cloud Security.

IAM Roles

AWS Identity and Access Management (IAM) roles enable customers to assign granular access policies and permissions to services and users on the AWS Cloud. This solution creates IAM roles that grant the solution's AWS Lambda functions access to access and create Regional resources.

Design considerations

Regional deployments

This solution uses Amazon Forecast, which is only available in supported AWS Regions. For the most current availability by Region, refer to the AWS Regional Services List.

Amazon QuickSight

This solution optionally uses an Amazon SageMaker Notebook Instance, where supported instance types depend on AWS Region and Availability Zone. Refer to the Supported Instance Types and Availability Zones topic in the *Amazon SageMaker Developer guide* for more information.

This solution will automatically create Amazon QuickSight data sources, data sets and analyses for each triggered forecast if Amazon QuickSight Enterprise Edition is enabled and the solution is deployed referencing a Amazon QuickSight analysis owner ARN.

AWS CloudFormation templates

This solution uses AWS CloudFormation to automate the deployment of the Improving Forecast Accuracy with Machine Learning solution in the AWS Cloud. It includes the following CloudFormation template, which you can download before deployment:

View Template

Improving-forecast-accuracy-with-machine-learning-demo.template: This template includes a demo stack along with the main stack listed below. The demo stack automates the process of generating a forecast for New York City Taxi rides per location per hour for the next 7 days. The demo shows how you can automatically include local weather and holiday information to generate an accurate forecast following the process outlined in the Amazon Forecast Weather Index – automatically include local weather to increase your forecasting model accuracy blog post.

View Template

Improving-forecast-accuracy-with-machine-learning.template: This template provides the main stack to launch the solution and all associated components. The default configuration deploys Amazon S3 buckets, AWS Lambda functions, an AWS Step Functions workflow, an Amazon SNS topic, and optionally an Amazon SageMaker Notebook Instance. You can customize the template to meet your specific needs.

> **Note**
> AWS CloudFormation resources are created from AWS Cloud Development Kit (AWS CDK) constructs.

Automated deployment

Before you launch the solution, review the architecture and configuration in this guide. Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Time to deploy: Approximately five minutes

Prerequisites

Update the stack

If you have previously deployed the solution, use this procedure to update the Improving Forecast Accuracy with Machine Learning CloudFormation stack to get the latest version of the solution.

1. Sign in to the AWS CloudFormation Console, select the existing Improving Forecast Accuracy with Machine Learning CloudFormation stack, and select **Update**.
2. Select **Replace current template**.
3. Under **Specify template**:
    a. Select **Amazon S3 URL**.
    b. Copy the link of the latest template for the stack.
    c. Verify that the correct template URL shows in the **Amazon S3 URL** text box, and choose **Next**. Choose **Next** again.
4. Under **Parameters**, review the parameters for the template and modify them as necessary. Refer to **Step 2. Launch the Stack** for details about the parameters.
5. Choose **Next**.
6. On the **Configure stack options** page, choose **Next**.
7. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template might create AWS Identity and Access Management (IAM) resources.
8. Choose **View change set** and verify the changes.
9. Choose **Update stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should see a status of **UPDATE_COMPLETE** in approximately 10 minutes depending on the options chosen.

Deployment overview

Use the following steps to deploy this solution on AWS. For detailed instructions, follow the links for each step.

Step 1. Launch the demo stack (optional) (p. 12)

- Launch the demo AWS CloudFormation template into your AWS account to learn how to get started with the solution.

  This step is optional. Start with Step 2 to generate a forecast using your own data.

Step 2. Launch the main stack (p. 14)

- Launch the main AWS CloudFormation template into your AWS account. Optionally, you can also continue to use the demo template if you launched it in Step 1.
- Review the template parameters, and adjust if necessary.

Step 3. Create and upload your Forecast configuration file (p. 17)

- Review the Forecast configuration file format.
- Modify the configuration file's default parameters for dataset group, datasets, predictor, and forecast.

Step 4. Experiment with Amazon Forecast (p. 19)

- Generate and evaluate multiple predictors using different configurations.

Step 5. Visualize forecast output (p. 19)

- Set up visualizations automatically using an Amazon QuickSight dashboard or a Jupyter Notebook instance.

Step 1. Launch the demo stack

This automated AWS CloudFormation template deploys the Improving Forecast Accuracy with Machine Learning solution in the AWS Cloud.

Note
You are responsible for the cost of the AWS services used while running this solution. For more details, view the Cost (p. 3) section in this guide, and refer to the pricing webpage for each AWS service used in this solution.

1. Sign in to the AWS Management Console and select the button below to launch the `improving-forecast-accuracy-with-machine-learning-demo.template` AWS CloudFormation template.

   Alternatively, you can download the template as a starting point for your own implementation.
2. The template launches in the US East (N. Virginia) Region by default. To launch the solution in a different AWS Region, use the Region selector in the console navigation bar.

   Note
   This solution uses the Amazon Forecast service, and Amazon SageMaker, which are not currently available in all AWS Regions. You must launch this solution in an AWS Region where Amazon Forecast is available. For the most current availability by Region, refer to the AWS Service Region Table.
3. On the **Create stack** page, verify that the correct template URL is in the **Amazon S3 URL** text box and choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack.
5. Under **Parameters**, review the parameters for this solution template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
Email	*<Optional input>*	The email that receives status notifications from the AWS Step Functions state machine. If this parameter is blank, you are not notified with your forecast results. You must accept the SNS subscription (through the email link that is sent after stack deployment) to activate notifications.
CloudWatch Log Level	WARNING	The Step Functions logs messages at this log level by default. Adjust for troubleshooting, or if logs are too verbose.
Target Time Series URL	NYC taxi data URL	The URL for the target time series to use.
Related Time Series URL	*<Optional input>*	The URL for the related time series to use.
Metadata URL	NYC taxi metadata URL	The URL for the item metadata dataset to use.
Forecast Defaults URL	NYC taxi forecast defaults URL	The URL to the `forecast-defaults.yaml` file used to configure all forecasts generated by the stack.
Forecast Bucket	*<Optional input>*	Leave this parameter blank to have the demo stack deploy the `improving-forecast-accuracy-with-machine-learning` stack for forecasting and use the forecast bucket from that stack. If you have already deployed the `improving-forecast-accuracy-with-machine-learning` stack, set this to the stack's forecast bucket name.

6. Choose **Next.**
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template might create AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation console in the **Stack info** column. You should receive a CREATE_COMPLETE status in approximately five minutes.

Note
In addition to the primary AWS Lambda functions used by the AWS Step Functions workflow deployed by this solution, this solution includes the `SolutionMetrics`, `BucketNameFunction`, `UrlDownloaderFunction`, `UrlInfoFunction`, and `UniqueNameFuction` Lambda functions, which run only during initial configuration or when resources are updated or deleted.
When you run this solution, you will notice multiple Lambda functions in the AWS Management Console. Only the AWS Lambda functions used by the AWS Step Functions workflow are regularly active. These functions are necessary to manage associated resources. Do not delete them.

Step 2. Launch the main stack

This automated AWS CloudFormation template deploys the Improving Forecast Accuracy with Machine Learning solution in the AWS Cloud.

Note
You are responsible for the cost of the AWS services used while running this solution. For more details, view the Cost (p. 3) section in this guide, and refer to the pricing webpage for each AWS service used in this solution.

1. Sign in to the AWS Management Console and select the button below to launch the `improving-forecast-accuracy-with-machine-learning.template` AWS CloudFormation template.

   Launch Solution

   Alternatively, you can download the template as a starting point for your own implementation.
2. The template launches in the US East (N. Virginia) Region by default. To launch the solution in a different AWS Region, use the Region selector in the console navigation bar.

   Note
   This solution uses the Amazon Forecast service, and when specified, Amazon SageMaker, which are not currently available in all AWS Regions. You must launch this solution in an AWS Region where Amazon Forecast is available. For the most current availability by Region, refer to the AWS Service Region Table.

3. On the **Create stack** page, verify that the correct template URL is in the **Amazon S3 URL** text box and choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack. The stack name must be 20 characters or less – for example, “forecast-stack”.
5. Under **Parameters**, review the parameters for this solution template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
Email	*<Optional input>*	The email that receives status notifications from the AWS Step Functions state machine. If this parameter is blank, you are not notified with your forecast

Parameter	Default	Description
		results. You must accept the SNS subscription (through the email link that is sent after stack deployment) to activate notifications.
Deploy QuickSight Dashboards	*<Optional input>*	To deploy Amazon QuickSight analyses automatically, enable Amazon QuickSight Enterprise Edition in your account and region, then set this parameter to the ARN of the desired default owner of the Amazon QuickSight analyses that will be created. For example, `arn:aws:quicksight:<region>:<account default/Admin/<username>`. To find the ARN, use the following AWS CLI command: `aws quicksight list-users --region us-east-1 --aws-account-id <your_account_id> --namespace default`
Deploy Jupyter Notebook	`No`	The Amazon SageMaker Jupyter Notebook Instance is not deployed by default.
Jupyter Notebook instance type	`ml.t2.medium`	The Amazon SageMaker Jupyter Notebook Instance type. The default configuration might not be supported in all Regions.
Jupyter Notebook volume size	*10*	The Amazon SageMaker Notebook Instance volume size. This must be an integer between 5 GB and 16384 GB (16 TB). Note that a value must be provided, even if the notebook is not deployed.

Parameter	Default	Description
KMS Key ARN used to encrypt Datasets and Predictors managed by Amazon Forecast	*<Optional input>*	While Amazon Forecast will encrypt your data by default, you can monitor and restrict access to your data by specifying a managed KMS key in your account that can be used by the Amazon Forecast service. Specifying an AWS KMS key ARN in this parameter allows Amazon Forecast to use that key to protect your data. Revoking, disabling, or modifying key policy associated with this key may render your data unusable by the Amazon Forecast service. To use your own key, specify the full key ARN, for example, `arn:aws:kms:<region>:<account_id>:ke` `f8fed2cd-14ab-4ac4-a8a3-57975cbff81b`. Leave this parameter blank to have Amazon Forecast use its default encryption configuration for your data.
CloudWatch Log Level	WARNING	The Step Functions logs messages at this log level by default. Adjust for troubleshooting, or if logs are too verbose.

Note
The Amazon SageMaker Jupyter Notebook Instances deployed by this solution allow direct internet access and are internet-enabled. Though not recommended for production use, these notebooks are useful for development testing and collaboration.
For a more secure production configuration, it is recommended that Amazon SageMaker Jupyter Notebook Instances are deployed using the security best practices outlined in the Connect a Notebook Instance to Resources in a VPC topic in the *Amazon SageMaker Developer Guide*.

6. Choose **Next.**
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation console in the **Stack info** column. You should receive a CREATE_COMPLETE status in approximately five minutes.

Note
In addition to the primary AWS Lambda functions used by the AWS Step Functions workflow deployed by this solution, this solution includes the `SolutionMetrics`, `RedeployLamdasFunction`, `BucketNameFunction`, and `UniqueNameFuction` Lambda functions, which run only during initial configuration or when resources are updated or deleted. When you run this solution, you will notice multiple Lambda functions in the AWS Management Console. Only the AWS Lambda functions used by the AWS Step Functions workflow are regularly active. However, you must not delete these, because it is necessary to manage associated resources.

Step 3. Create and upload your forecast configuration file

The forecast configuration file (`forecast-defaults.yaml`) must be present at the root of the S3 forecast bucket prior to triggering the AWS Step Functions state machine by uploading your data. This is a manual task that must be performed prior to generating forecasts. Follow the procedure below to generate the forecast configuration file.

In the output of your deployed CloudFormation stack from Step 1, take note of the `ForecastBucketName` value under the **Outputs** tab. This is the bucket where the configuration file must be uploaded. You can use the AWS Management Console to copy your forecast configuration file to the root of your `ForecastBucket` bucket.

The following is a minimal configuration file that you can use to configure the AWS Step Functions state machine:

```
Default:
  DatasetGroup:
    Domain: RETAIL

  Datasets:
    - Domain: RETAIL
      DatasetType: TARGET_TIME_SERIES
      DataFrequency: D
      TimestampFormat: yyyy-MM-dd
      Schema:
        Attributes:
          - AttributeName: item_id
            AttributeType: string
          - AttributeName: timestamp
            AttributeType: timestamp
          - AttributeName: demand
            AttributeType: float

  Predictor:
    MaxAge: 604800
    PerformAutoML: True
    ForecastHorizon: 72
    FeaturizationConfig:
      ForecastFrequency: D

  Forecast:
    ForecastTypes:
      - "0.10"
      - "0.50"
      - "0.90"
```

The following sections must be present in your configuration file:

- **Default** – All files uploaded use the forecast defaults unless an override is provided. For more information about overriding defaults, refer to Overriding forecast defaults (p. 30).
  - **Default**.**DatasetGroup** – Under the default key, you must specify your dataset group configuration. The only supported configuration key under **DatasetGroup** is `Domain`, which represents the supported dataset domain. This can be RETAIL, CUSTOM, INVENTORY_PLANNING, EC2_CAPACITY, WORK_FORCE, WEB_TRAFFIC, or METRICS.

- **Default**.**Datasets** – Under the default key, you must either specify your dataset configurations, or reference datasets from another top-level configuration item. In order to specify your dataset configurations, this must be configured as an array, and each array element must contain the following:
  - **Domain** (matching the domain of the **Default.DatasetGroup**).
  - **DatasetType** (TARGET_TIME_SERIES, RELATED_TIME_SERIES or _ITEM_METADATA).
  - **DataFrequency** (Y, M, W, D, 30min, 15min or 1min — only required for target time series and related time series dataset types).
  - **TimestampFormat** (this must be yyyy-MM-dd if the DataFrequency is Y, M, W or D, and must be yyyy-MM-dd HH:mm:ss if your DataFrequency is 30min, 15min or 1min — only required for target time series and related time series dataset types).
  - **Schema** (a schema attribute is required for every field in a dataset. The schema object contains an attributes object, which describes an array of SchemaAttribute objects (containing AttributeName and AttributeValue).

    To reference datasets from another top-level configuration item, that item must be configured with datasets (as above, containing Domain, DatasetType, DataFrequency and TimestampFormat), then this item contains the following key:
  - **From** (other dataset group name, for example, Default).

    If you want to use AWS managed weather data ingestion to improve your forecast accuracy you can also specify GeoLocationFormat (as LAT_LONG) and TimeZone for your TARGET_TIME_SERIES dataset, while ensuring that the schema has a geolocation dimension (with AttributeType geolocation) and values in LAT_LONG format (such as 45.4236_75.7009) for each item. If this is enabled, you must also ensure that your predictor configuration specifies the geolocation attribute as a forecast dimension. If you have a related time series dataset, it must also include the geolocation dimension as well. For more information on this particular configuration, refer to *Demo stack configuration (p. 28)*.

- **Default**.**Predictor** – Under the default key, you must specify your predictor configuration. This can include all supported properties of the Amazon Forecast service for the CreatePredictor API call, and can also contain:
  - **MaxAge** (number of seconds between consecutive dataset import jobs where a new file upload should not generate a new predictor). This defaults to 604800 seconds (one week), if not otherwise specified.

- **Default**.**Forecast** – Under the forecast key, you must specify your forecast configuration. This can include all supported properties of the Amazon Forecast service for the CreateForecast API call, commonly "ForecastTypes".

Step 4. Experiment with Amazon Forecast

The Improving Forecast Accuracy with Machine Learning solution makes it straightforward to generate and evaluate multiple predictors using different configurations by tuning a forecast configuration file and uploading the same dataset under different names. For example, during testing, you can evaluate the following different scenarios:

- **Scenario A:** Try AutoML to automatically determine which algorithm generates the most accurate forecast for your time-series data.
- **Scenario B:** Based on the AutoML results, select the winning algorithm and tune its hyperparameters based on your domain specific knowledge, or using automated hyperparameter Optimization (HPO).
- **Scenario C:** Based on the AutoML results, select the top-performing model and experiment with different predictor featurization configurations concurrently.

Refer to *Forecast experimentation* (p. 32) for a detailed walkthrough of automating Amazon Forecast experiments with the Improving Forecast Accuracy with Machine Learning solution. To improve forecast accuracy, ensure that you reference and follow the best practices listed in the Time Series Forecasting Principles with Amazon Forecast AWS technical guide.

Step 5. Visualize forecast output

After this solution successfully deploys, you can configure your data sets and visualizations using any visualization service. This implementation guide provides procedures for setting up visualizations using either an Amazon Quicksight analysis or a Jupyter Notebook instance.

Visualization with Amazon QuickSight

An Amazon QuickSight analysis can be created on a per-forecast basis to assist users with forecast output visualization across hierarchies and categories of forecasted items.

Consider the following scenarios:

- You want to quickly visualize any probabilistic forecast for any item or aggregate set of items.
- You want to look up and aggregate forecasted items across any of your forecast dimensions. You want to view forecasts for all your sales on a per-store, per-region, or per-geography basis.
- You want insight into the performance of certain types of items (for example, brands), or to compare the sales of two different brands, or to identify high and low performing items.

Amazon QuickSight analyses can combine forecast input and output visualizations that present data to address these scenarios, as well as other useful analyses. You can create each of the above visualizations in Amazon QuickSight by dragging and dropping data into the solution's bundled QuickSight analysis.

As forecasts are generated, the raw dataset files, predictor backtest data, and forecast outputs are combined to create a combined forecast output file containing both the forecast input and forecast horizon. These are output to Amazon S3 in the forecast data bucket in a format that can be ingested by Amazon QuickSight via Amazon Athena. In order to offer generic dashboarding capability, column names are mapped to generic names that can be dragged to Amazon QuickSight charts. This Amazon Athena table can also be consumed by third-party visualization tools.

This solution provides the following mappings:

Dataset Domain	Fields	Mapping
RETAIL	item_id (string) timestamp (date) demand (float or integer)	Identifier (string) timestamp (date) metric (float)
CUSTOM	item_id (string) timestamp (date) target_value (float or integer)	identifier (string) timestamp (date) metric (float)
INVENTORY_PLANNING	item_id (string) timestamp (date) demand (integer)	identifier (string) timestamp (date) metric (float)
EC2_CAPACITY	instance_type (string) timestamp (date) number_of_instances (integer)	identifier (string) timestamp (date) metric (float)
WORK_FORCE	workforce_type (string) timestamp (date) workforce_demand (float)	identifier (string) timestamp (date) metric (float)
WEB_TRAFFIC	item_id (string) timestamp (string) value (float)	identifier (string) timestamp (date) metric (float)
METRICS	metric_name (string) timestamp (date) metric_value (float)	identifier (string) timestamp (date) metric (float)

In addition to these mappings, arbitrary forecast dimensions and metadata can be added to augment the analysis:

- Up to 10 forecast dimensions (mapped as forecast_dimension_1 … forecast_dimension_10)
- Up to 10 forecast metadata attributes (mapped as forecast_metadata_1 … forecast_metadata_10)

Predictor backtest export data is also added to the analysis, including:

- Predictor backtest quantiles (backtest_p1 … backtest_p99), however only the calculated quantiles will have values, the rest will be null
- Predictor backtest accuracy metrics values (wQL, RMSE and WAPE) are not currently added to the export data

Forecast output through the forecast horizon is also added to the analysis, including:

- Forecast quantiles (p1 … p99). However, only the calculated quantiles will have values, the rest will be null.

> **Note**
> The Amazon Athena table representing the combined forecast output is limited in size to the maximum cumulative size of all files in your Amazon S3 bucket (currently 30GB for Amazon Forecast datasets). The solution automatically partitions data on the month_starting string value.
> This partitioning strategy effectively limits the amount of data per partition to between approximately 720 records and 43,200 records, which reduces the amount of data that needs to be queried when using direct queries.
> To use the partition, filter your queries using a WHERE clause on month_starting.
> Further, we format the table as Parquet to minimize the cost of querying the data when using direct queries from Athena. Performance can also be improved by importing the data directly into SPICE, which avoids performing direct queries.

After the Amazon Athena table representing the combined forecast output file has been created, a QuickSight analysis is cloned into your account matching the name of the forecast export. This analysis can be customized using drag-and-drop controls to suit your needs, then published and shared as a dashboard within your organization.

Use the following procedure to customize your visualization.

Example 1	Instructions
I want to be able to quickly visualize any probabilistic forecast for any identifier or set of items as well as some historical demand.	• Drag the desired probabilistic forecasts to the Forecast Horizon line chart. • Items can be filtered out based on the Amazon QuickSight controls for dimensions and metadata – for instance, restricting to only items sold matching a specific dimension (e.g. location) . • In the diagram below, note the field wells for the Forecast Horizon line chart contain the three quantiles p10, p50 and p90. Select the output quantiles of your forecast to visualize in this graph. • To restrict the date range, adjust the **Date Filter** control.

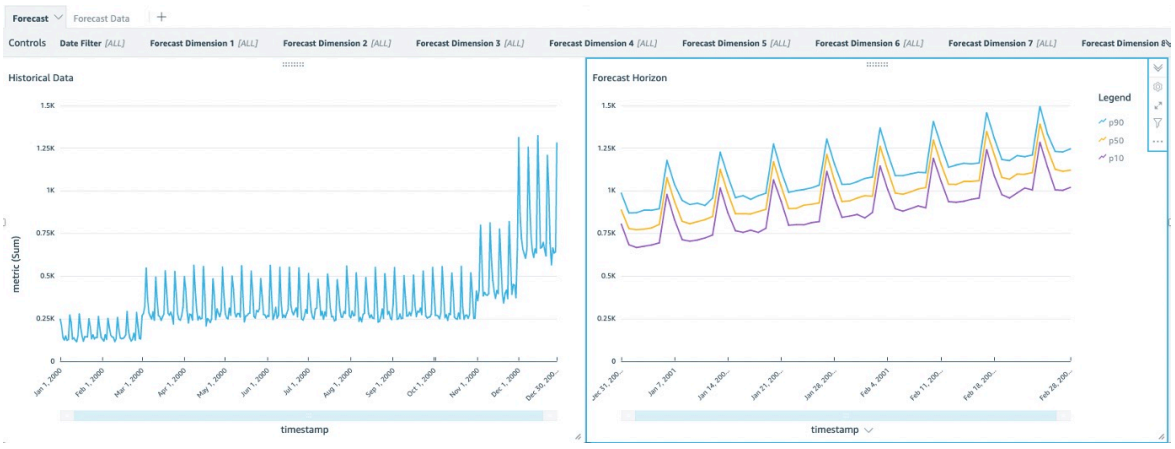

Figure 4: Example forecast to quickly visualize any probabilistic forecast for any identifier or set of items as well as some historical demand.

Example 2	Instructions
I want to be able to look up and aggregate items across my forecast dimensions. Specifically, I want to view forecasts for all items on a per-dimension basis.	• To filter the data, use the multiselect dropdown menus, in this case, we are filtering based on Forecast Dimension 1, which corresponds in this forecast to store location. We have selected the **Kanata** location to display only data for that location.

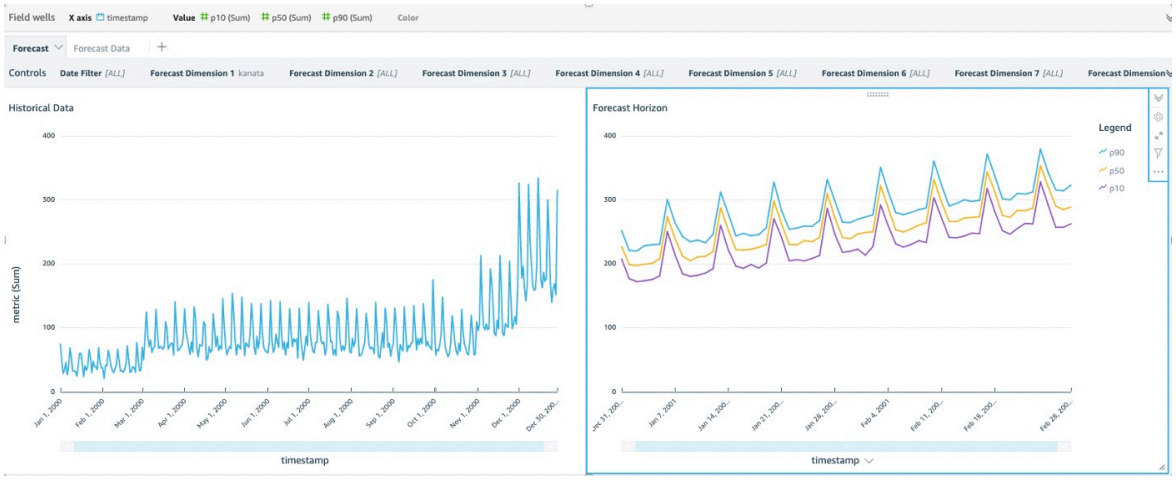

Figure 5: Example forecast to look up and aggregate items across forecast dimensions.

Example 3	Instructions
I want better insight into the performance of certain items matching different metadata attributes (for example, different brands).	• Consuming Amazon Forecast fields in QuickSight can allow us to further filter based on metadata attributes. Some commonly used metadata might be to filter based on category, brand, or genre. • To filter the displayed data, use the controls to filter based on the metadata fields provided to the visualization. In the following diagram, data is further filtered to only include sales of items matching of 'brand x'

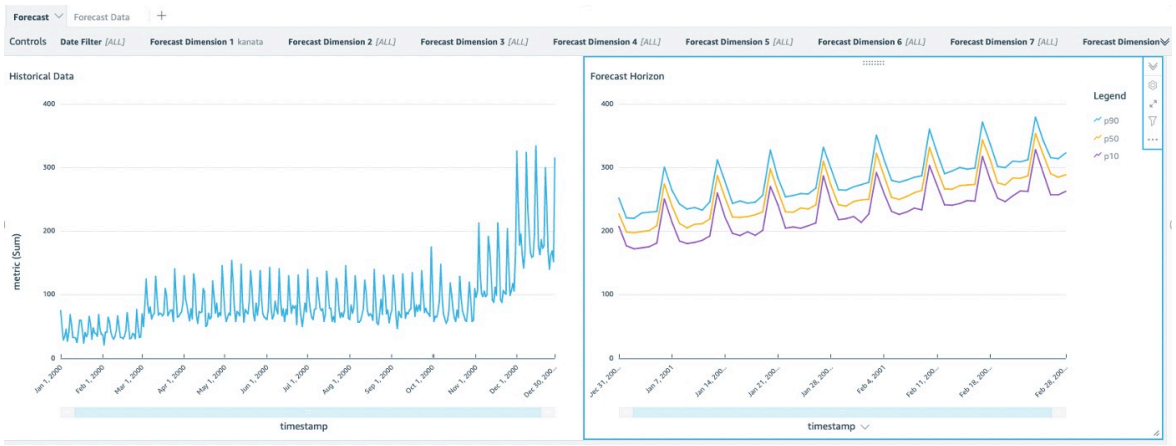

Figure 6: Example forecast for better insight into the performance of certain items matching different metadata attributes.

Example 4	Instructions
I want to be able to visualize predictor accuracy on the training data and combine it with the forecast data to assess if my forecast is likely to be accurate.	• Drag the desired backtest output to the Historical Data line chart. • Items can be filtered out based on the Amazon QuickSight controls for dimensions and metadata – for instance, restricting to only items sold matching a specific dimension (e.g. location). • In the diagram below, note the field wells for the Historical Data line chart contain the input data and forecast backtest exports for three quantiles -p10, p50 and p90. Select the output quantiles of your forecast to visualize in this graph. To restrict the date range, adjust the **Date Filter** control.

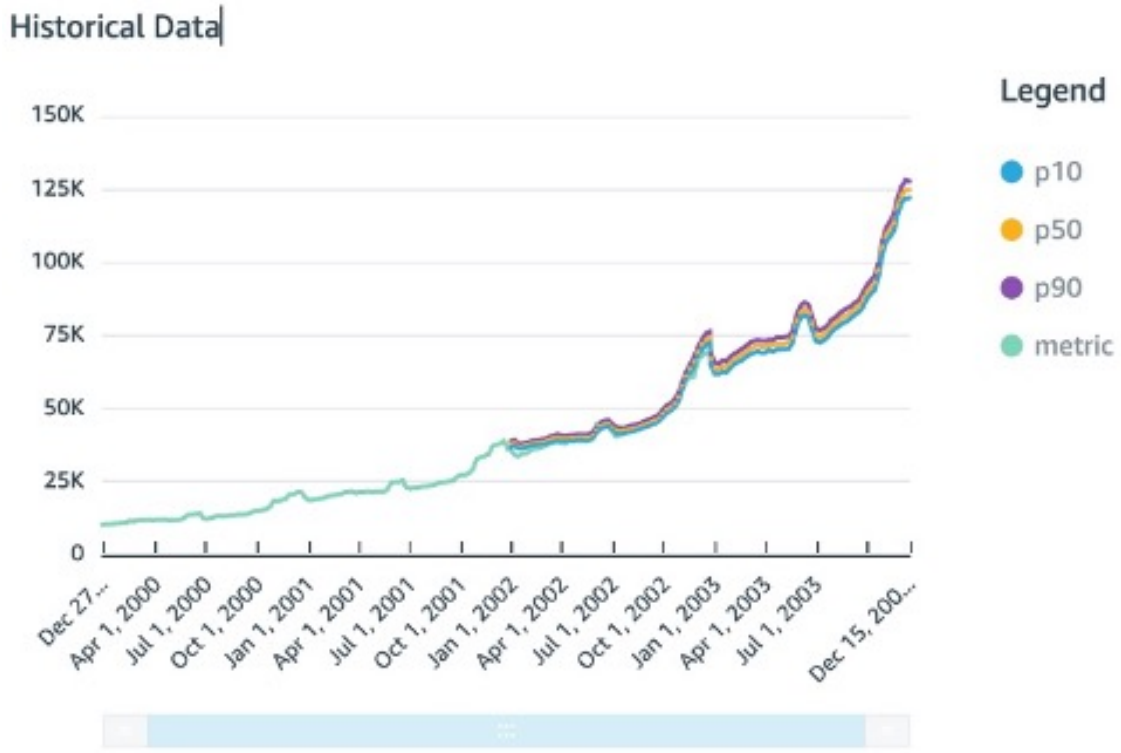

Figure 7: Example forecast to visualize predictor accuracy on the training data and combine it with the forecast data.

Visualization with a Jupyter Notebook

The optional Jupyter Notebook included with the solution can be used to visualize forecast output and related time series to help you select features for your related timeseries datasets.

The graph in Figure 8 displays item demand for a synthetic dataset alongside the source code for this solution. Two predictors have been generated, one using Prophet and the other with DeepAR+. Under the demand and forecast, a related time series (item price) is displayed.

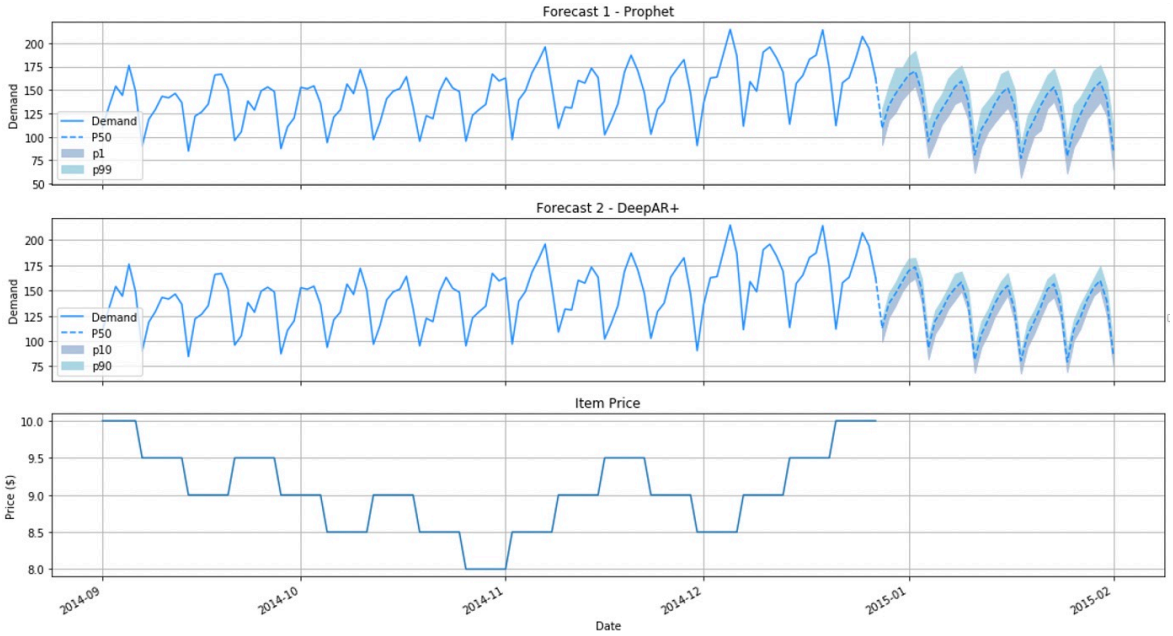

Figure 8: Item demand vs price

Comparing the results show similarities between the forecast generated using the Prophet algorithm and the DeepAR+ algorithm. These results show that p1, p50, and p99 forecasts were generated with Prophet, and p10, p50, and p90 forecasts were generated with DeepAR+. The similarities in output (both across models and across probabilistic forecasts) suggest that using the p50 forecast for this dataset is sufficient if the business impact of over- or under-forecasting is negligible.

You can use the following experiments to identify features that improve forecast accuracy.

Upload several copies of your dataset(s) to the solution-managed S3 bucket, with different related timeseries datasets (for example, millimeters of rain that fell during the time interval, or number of ongoing sporting events that might have influenced demand during that time interval). The solution generates predictors based on your configuration. Observe the predictor accuracy in the console and graph the results as shown in Figure 8.

Use the following procedure to open the Jupyter notebook.

1. Sign in to the AWS Console.
2. Navigate to the Amazon SageMaker service.
3. Find your notebook instance and select **Open JupyterLab**.
4. A new browser tab opens. Navigate to the **improving-forecast-accuracy-with-machine-learning/v1.1.0/notebooks** folder, and select `SampleVisualization.ipynb`.
5. In the first cell of the notebook, configure your start date, end date, and paths to your time series data, related timeseries data, and forecast exports.

```
# Set the start and end dates of the forecast
start_date = '2014-09-01' # YYYY-MM-DD
end_date = '2015-02-01'   # YYYY-MM-DD

# provide the full CSV name uploaded to the /train folder in S3
demand_dataset_name = 'RetailDemandTRMProphet.csv'
related_dataset_name = 'RetailDemandTRMProphet.related.csv'
```

```
# provide the exports to show from the export/ folder in S3 (these are created by Amazon
 Forecast)
forecast_exports = [
    {
        'path': 'export_2020_07_04_17_30_13/
export_2020_07_04_17_30_13_2020-07-06T19-22-29Z_part0.csv',
        'name': 'Forecast 1 - Prophet'
    },
    {
        'path': 'export_2020_07_02_15_19_45/
export_2020_07_02_15_19_45_2020-07-02T17-12-57Z_part0.csv',
        'name': 'Forecast 2 - DeepAR+'
    }

]
```

6. In the Run menu, select **Run All Cells**. If your data is not aggregated by date, modify the date formatter for the graph axis.

Experiment and develop a forecast model for your dataset. When new demand data is available, follow the procedure in *Forecast experimentation* (p. 32) to update your datasets and generate a new predictor, then compare predictor accuracy through similar visualizations.

Additional resources

AWS services

- Amazon Athena
- AWS CloudFormation
- Amazon CloudWatch
- Amazon Forecast
- AWS Lambda
- Amazon Simple Storage Service
- Amazon Simple Notification Service
- AWS Step Functions
- Amazon QuickSight
- AWS Glue

Related AWS Resources

- The Amazon Forecast Visualization Automation blog discusses a pipeline similar to the Improving Forecast Accuracy with Machine Learning solution, but its default implementation is designed for cases where related time-series data and item metadata are not required.

Demo stack configuration

The configuration for the demo stack is shown below. This example configuration includes:

- A target time series compatible with the Amazon Forecast Weather Index
- An example of how to use the Amazon Forecast Weather Index (through predictor InputDataConfig)
- An example of how to customize hyperparameters for model training (through TrainingParameters

```
NYC_TAXI_DEMO:
  DatasetGroup:
    Domain: CUSTOM

  Datasets:
    - Domain: CUSTOM
      DatasetType: TARGET_TIME_SERIES
      DataFrequency: H
      TimestampFormat: yyyy-MM-dd HH:mm:ss
      GeolocationFormat: LAT_LONG
      TimeZone: America/New_York
      Schema:
        Attributes:
          - AttributeName: timestamp
            AttributeType: timestamp
          - AttributeName: item_id
            AttributeType: string
          - AttributeName: target_value
            AttributeType: float
          - AttributeName: geolocation
            AttributeType: geolocation
    - Domain: CUSTOM
      DatasetType: ITEM_METADATA
      Schema:
        Attributes:
          - AttributeName: item_id
            AttributeType: string
          - AttributeName: borough
            AttributeType: string
          - AttributeName: taxi_zone
            AttributeType: string
          - AttributeName: taxi_service_zone
            AttributeType: string

  Predictor:
    MaxAge: 604800  # one week
    AlgorithmArn: arn:aws:forecast:::algorithm/Deep_AR_Plus
    ForecastHorizon: 168
    FeaturizationConfig:
      ForecastFrequency: H
      ForecastDimensions: ["geolocation"]
      Featurizations:
        - AttributeName: target_value
          FeaturizationPipeline:
            - FeaturizationMethodName: filling
              FeaturizationMethodParameters:
                aggregation: sum
                backfill: zero
                frontfill: none
                middlefill: zero
```

```
    InputDataConfig:
      SupplementaryFeatures:
        - Name: holiday
          Value: US
        - Name: weather
          Value: "true"
    EvaluationParameters:
      NumberOfBacktestWindows: 3
      BackTestWindowOffset: 168
    ForecastTypes:
      - "0.50"
      - "0.60"
      - "0.70"
    TrainingParameters:
      context_length: "63"
      epochs: "250"
      learning_rate: "0.014138165570842774"
      learning_rate_decay: "0.5"
      likelihood: negative-binomial
      max_learning_rate_decays: "0"
      num_averaged_models: "1"
      num_cells: "40"
      num_layers: "2"

  Forecast:
    ForecastTypes:
      - "0.50"
      - "0.60"
      - "0.70"

Default:  # a default is required
  DatasetGroup:
    Domain: RETAIL

  Datasets:
    - Domain: RETAIL
      DatasetType: TARGET_TIME_SERIES
      DataFrequency: D
      TimestampFormat: yyyy-MM-dd
      Schema:
        Attributes:
          - AttributeName: item_id
            AttributeType: string
          - AttributeName: timestamp
            AttributeType: timestamp
          - AttributeName: demand
            AttributeType: float

  Predictor:
    MaxAge: 604800  # one week
    AlgorithmArn: arn:aws:forecast:::algorithm/ARIMA
    ForecastHorizon: 30
    FeaturizationConfig:
      ForecastFrequency: D

  Forecast:
    ForecastTypes:
      - "0.10"
      - "0.50"
      - "0.90"
```

Overriding forecast defaults

High forecast accuracy might require more than time-series data. We recommend that you create override configurations in your forecast configuration file to represent these jobs. The following example shows an override forecast configuration that matches the following dataset files being uploaded:

- `RetailDemandTRMProphet.csv` – demand data
- `RetailDemandTRMProphet.related.csv` – related demand data
- `RetailDemandTRMProphet.metadata.csv` – item metadata

`RetailDemandTRMProphet` is a top-level key in the forecast configuration file (alongside the required `Default` key).

```
RetailDemandTRMProphet:
  DatasetGroup:
    Domain: RETAIL

  Datasets:
    - Domain: RETAIL
      DatasetType: TARGET_TIME_SERIES
      DataFrequency: D
      TimestampFormat: yyyy-MM-dd
      Schema:
        Attributes:
          - AttributeName: item_id
            AttributeType: string
          - AttributeName: timestamp
            AttributeType: timestamp
          - AttributeName: demand
            AttributeType: float

    - Domain: RETAIL
      DatasetType: RELATED_TIME_SERIES
      DataFrequency: D
      TimestampFormat: yyyy-MM-dd
      Schema:
        Attributes:
        - AttributeName: item_id
          AttributeType: string
        - AttributeName: timestamp
          AttributeType: timestamp
        - AttributeName: price
          AttributeType: float

    - Domain: RETAIL
      Type: ITEM_METADATA
      Schema:
        Attributes:
          - AttributeName: item_id
            AttributeType: string
          - AttributeName: category
            AttributeType: string
          - AttributeName: brand
            AttributeType: string

  Predictor:
    AlgorithmArn: arn:aws:forecast:::algorithm/Prophet
```

```
  ForecastHorizon: 72
  FeaturizationConfig:
    ForecastFrequency: D
    Featurizations:
      - AttributeName: price
        FeaturizationPipeline:
          - FeaturizationMethodName: filling
            FeaturizationMethodParameters:
              futurefill: max
              middlefill: median
              backfill: median

Forecast:
  ForecastTypes:
    - "0.01"
    - "0.50"
    - "0.99"
```

Forecast experimentation

This section provides an example scenario of retail forecasting.

A shop owner would like to predict future demand for a specific SKU. To easily evaluate several predictor options, they can use the AutoML feature of Amazon Forecast. The example uses the following steps:

1. Set up defaults for their dataset and perform a forecast using AutoML.
2. Override their defaults to add information about related time-series data (for example, retail price), and item metadata, and then perform automated hyperparameter optimization.

   Note
   Amazon Forecast predictor algorithms for DeepAR+ and Prophet can benefit from related time-series data. NPTS, ARIMA and ETS do not take related time-series data into consideration. Item metadata can only be used by DeepAR+.
3. Select the top-performing model and experiment with different predictor featurization configurations concurrently.

Set up defaults and forecast using AutoML

This example uses synthetic retail demand data for one SKU. The dataset comprises daily sales data ("Item Demand") for this SKU (item_000), starting at date 2000-01-01 and ending at date 2014-12-27.

The graph in Figure 5 displays item demand and probabilistic forecasts for a synthetic dataset as processed by this solution. Two predictors have been generated, one using Prophet and the other with DeepAR+. Under the demand and forecast, a related time series (item price) is displayed.

Note
The synthetic dataset in Figure 9 is included in the source code for this solution.

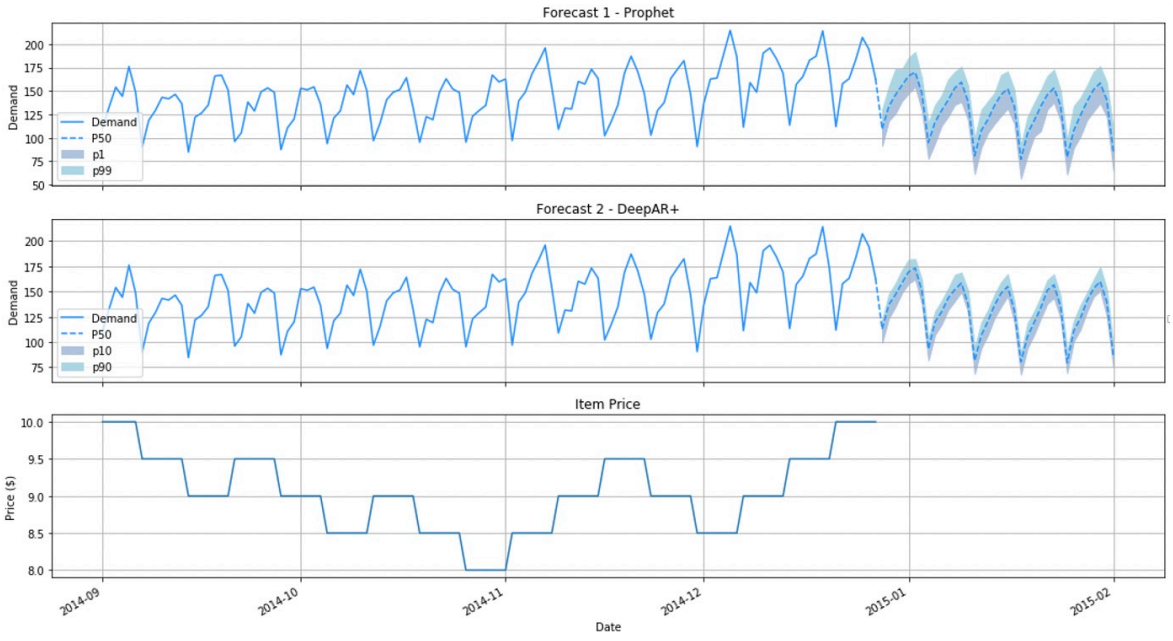

Figure 9: Retail demand data and related data

Comparing the results show similarities between the forecast generated using the Prophet algorithm and the DeepAR+ algorithm. These results show that p1, p50, and p99 forecasts were generated with Prophet, and p10, p50, and p90 forecasts were generated with DeepAR+. The similarities in output (both across models and across probabilistic forecasts) suggest that using the p50 forecast for this dataset is sufficient if the business impact of over or under-forecasting is negligible.

This example uses the following forecast configuration file `forecast-defaults.yaml`:

```
Default:
  DatasetGroup:
    Domain: RETAIL

  Datasets:
    - Domain: RETAIL
      DatasetType: TARGET_TIME_SERIES
      DataFrequency: D
      TimestampFormat: yyyy-MM-dd
      Schema:
        Attributes:
          - AttributeName: item_id
            AttributeType: string
          - AttributeName: timestamp
            AttributeType: timestamp
          - AttributeName: demand
            AttributeType: float

  Predictor:
    MaxAge: 604800  # one week
    PerformAutoML: True
    ForecastHorizon: 30
    FeaturizationConfig:
      ForecastFrequency: D

  Forecast:
    ForecastTypes:
      - "0.01"
      - "0.50"
      - "0.99"
```

This default configuration tells the Improving Forecast Accuracy with Machine Learning solution that all datasets uploaded represent retail data of the format specified under **datasets**. Predictors are retrained if new data is uploaded after one week (before this, forecasts will simply be run against the most recent data).

We will now begin importing data and generating forecasts using the solution. First, use the AWS Management Console to upload the default configuration file to the root of the `Forecast` data bucket. The configuration file must be named `forecast-defaults.yaml`.

Second, upload our dataset to the same Amazon S3 bucket as the configuration file, in the **train** folder. Upload the file as `retail_experiment_1.csv`.

> **Note**
> Your filenames must not start with a number, and must only contain upper and lowercase letters (a-z, A-Z) and the underscore ("_") character.

Confirm that the Improving Forecast Accuracy with Machine Learning solution is in progress by navigating to the AWS Step Functions console. If you see an deployment in progress "retail_experiment_1_target_time_series_<id>," that reflects the name of the dataset uploaded, its dataset type (target time series), and a unique ID. For an example of the AWS Step Functions state machine in process, refer to Figure 6.

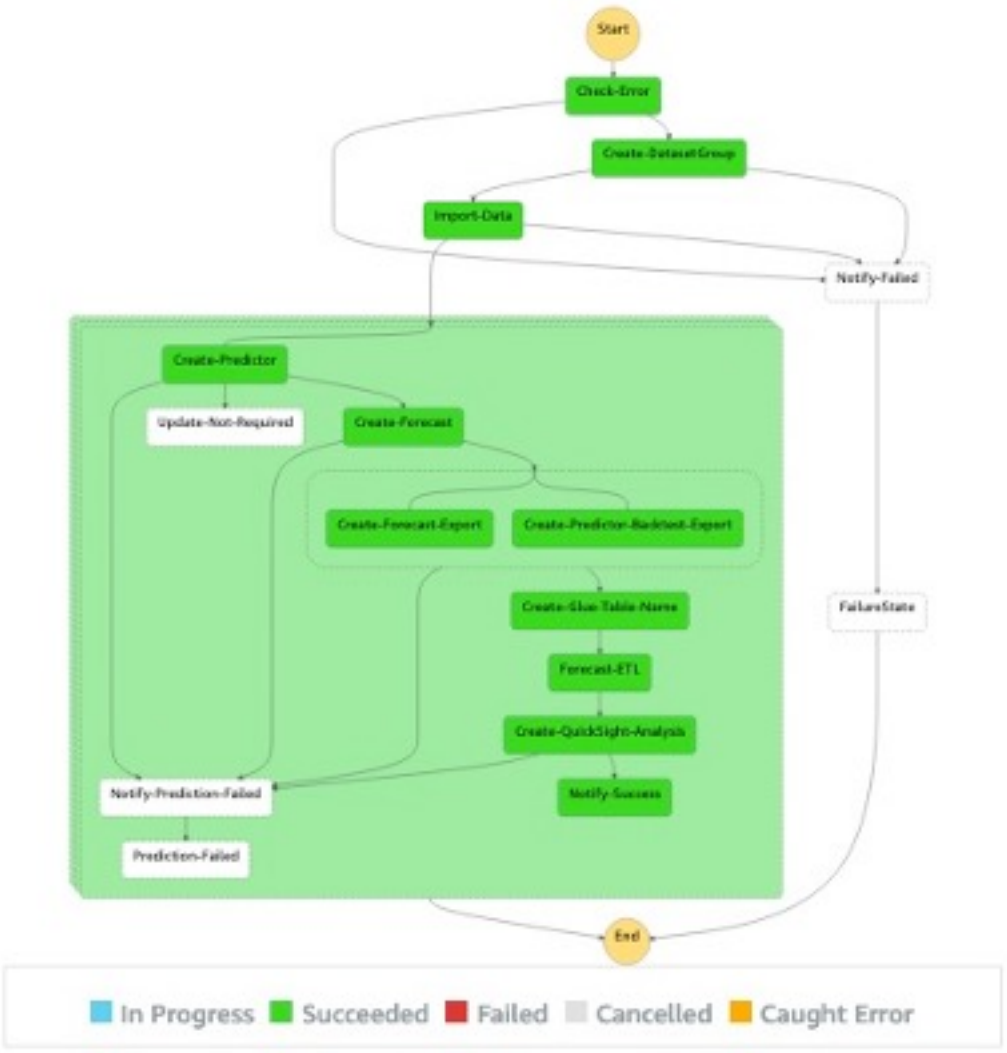

Figure 6: Improving Forecast Accuracy with Machine Learning state machine

As the state machine progresses, individual nodes change color from white (not executed) to blue (in progress), to green (succeeded), orange (caught error), or red (failed). Certain steps will take more time (specifically, Import-Data, Create-Predictor, and Create-Forecast). You do not have to monitor the execution of the state machine – an email is sent to the subscriber to the Amazon SNS topic.

After the AWS Step Functions state machine completes, that the winning AutoML algorithm is DeepAR +, with a wQL[0.5]/MAPE of 0.0614. You can graph the output of your forecast in the Amazon Forecast console under **Forecast lookup**, through SQL quieries in Amazon Athena or by using the included QuickSight visualization.

Override your defaults to improve your forecast

Based on the results of the AutoML predictor training, you should test whether or not adding related metadata and item metadata might improve our forecast accuracy.

For the Improving Forecast Accuracy with Machine Learning solution to consume related time-series data and item metadata, you must provide it with a new configuration. Replace the forecast

configuration file, `forecast-defaults.yaml`. The new configuration file states that for datasets named `retail_experiment_2`, you expect time-series data, related time-series data, and item metadata to be uploaded prior to generating a predictor and forecast. Specify that the related time-series data will contain a price, and the item metadata will contain an item category and brand.

Because the related time-series data does not contain future pricing information, you must inform the Amazon Forecast service to perform future filling for missing price values using the median.

The following example is the new defaults and override forecast configuration file:

```
Default:
  DatasetGroup:
    Domain: RETAIL

  Datasets:
    - Domain: RETAIL
      DatasetType: TARGET_TIME_SERIES
      DataFrequency: D
      TimestampFormat: yyyy-MM-dd
      Schema:
        Attributes:
          - AttributeName: item_id
            AttributeType: string
          - AttributeName: timestamp
            AttributeType: timestamp
          - AttributeName: demand
            AttributeType: float

  Predictor:
    MaxAge: 604800  # one week
    PerformAutoML: True
    ForecastHorizon: 30
    FeaturizationConfig:
      ForecastFrequency: D

  Forecast:
    ForecastTypes:
      - "0.10"
      - "0.50"
      - "0.90"


retail_experiment_2:
  DatasetGroup:
    Domain: RETAIL

  Datasets:
    - Domain: RETAIL
      DatasetType: TARGET_TIME_SERIES
      DataFrequency: D
      TimestampFormat: yyyy-MM-dd
      Schema:
        Attributes:
          - AttributeName: item_id
            AttributeType: string
          - AttributeName: timestamp
            AttributeType: timestamp
          - AttributeName: demand
            AttributeType: float
    - Domain: RETAIL
      DatasetType: RELATED_TIME_SERIES
      DataFrequency: D
      TimestampFormat: yyyy-MM-dd
      Schema:
```

```
      Attributes:
        - AttributeName: item_id
          AttributeType: string
        - AttributeName: timestamp
          AttributeType: timestamp
        - AttributeName: price
          AttributeType: float
  - Domain: RETAIL
    DatasetType: ITEM_METADATA
    Schema:
      Attributes:
        - AttributeName: item_id
          AttributeType: string
        - AttributeName: category
          AttributeType: string
        - AttributeName: brand
          AttributeType: string

Predictor:
  AlgorithmArn: arn:aws:forecast:::algorithm/Deep_AR_Plus
  MaxAge: 604800  # one week
  PerformHPO: True
  ForecastHorizon: 30
  FeaturizationConfig:
    ForecastFrequency: D
    Featurizations:
      - AttributeName: price
        FeaturizationPipeline:
          - FeaturizationMethodName: filling
            FeaturizationMethodParameters:
              futurefill: median
              middlefill: median
              backfill: median

Forecast:
  ForecastTypes:
    - "0.01"
    - "0.50"
    - "0.99"
```

After our configuration is set, we upload the time-series data (`retail_experiment_2.csv`), related time-series data (`retail_experiment_2.related.csv`) and item metadata (`retail_experiment_2.metadata.csv`).

Note
You can add related time-series data and item metadata to improve your predictions, predictors, and forecasts. We recommend that you specify an override configuration for any dataset group requiring related time-series data and metadata. This solution recognizes related time-series data and metadata by filenames uploaded with names matching `<dataset_group_name>.<related|metadata>.csv`.
In this example, we used item price as a related time series to improve forecast accuracy.

When you upload these new datasets, you will receive two emails notifying you that datasets are missing – this is because the predictor can only be generated after all three datasets are present. If you navigate to the Amazon Forecast console after uploading all three files, you will see the dataset import jobs in progress for all three files. After the last file uploads and is ACTIVE, you can observe that a predictor is generated for your data, and a forecast will be generated.

We now have two copies of the same data uploaded to separate dataset groups, which helps you track your experiments independently. For example, you might want to change the related time-series dataset and re-run the forecast to do a what-if analysis on different future pricing strategies.

Warning
If you attempt to override the configuration for the existing experiment (retail_experiment_1) you will observe an error in predictor generation because that experiment was configured with only one dataset in its dataset group at the time of predictor generation.

Use the same dataset in multiple predictor configurations for what-if testing

To perform what-if testing, you can leverage the same dataset across several predictor configurations. For example, the following forecast configuration file, `forecast-defaults.yaml`, allows the same dataset to be used across multiple predictor configurations – the first simulating setting the future price of an item to $20, and the second simulating setting the future price of an item to $15. The dataset configuration is specified only for the demand_15 dataset group, and demand_20 uses those datasets.

As demand_15.csv, demand_15.related.csv, and demand_15.metadata.csv are uploaded, the datasets are imported once and referenced by both the demand_15 and demand_20 dataset groups. Predictors are trained under each dataset group to track these experiments independently.

```
Default:
  <your default configuration goes here>

demand_20:
  DatasetGroup:
    Domain: RETAIL

  Datasets:
    From: demand_15

  Predictor:
    AlgorithmArn: arn:aws:forecast:::algorithm/Prophet
    MaxAge: 604800  # one week
    ForecastHorizon: 30
    FeaturizationConfig:
      ForecastFrequency: D
      Featurizations:
        - AttributeName: price
          FeaturizationPipeline:
            - FeaturizationMethodName: filling
              FeaturizationMethodParameters:
                futurefill: value
                futurefill_value: "20"
                middlefill: median
                backfill: median

  Forecast:
    ForecastTypes:
      - "0.10"
      - "0.50"
      - "0.90"

demand_15:
  DatasetGroup:
    Domain: RETAIL

  Datasets:
    - Domain: RETAIL
      DatasetType: TARGET_TIME_SERIES
      DataFrequency: D
```

```
    TimestampFormat: yyyy-MM-dd
    Schema:
      Attributes:
        - AttributeName: item_id
          AttributeType: string
        - AttributeName: timestamp
          AttributeType: timestamp
        - AttributeName: demand
          AttributeType: float
  - Domain: RETAIL
    DatasetType: RELATED_TIME_SERIES
    DataFrequency: D
    TimestampFormat: yyyy-MM-dd
    Schema:
      Attributes:
        - AttributeName: item_id
          AttributeType: string
        - AttributeName: timestamp
          AttributeType: timestamp
        - AttributeName: price
          AttributeType: float
  - Domain: RETAIL
    DatasetType: ITEM_METADATA
    Schema:
      Attributes:
        - AttributeName: item_id
          AttributeType: string
        - AttributeName: category
          AttributeType: string
        - AttributeName: brand
          AttributeType: string

Predictor:
  AlgorithmArn: arn:aws:forecast:::algorithm/Prophet
  MaxAge: 604800  # one week
  ForecastHorizon: 30
  FeaturizationConfig:
    ForecastFrequency: D
    Featurizations:
      - AttributeName: price
        FeaturizationPipeline:
          - FeaturizationMethodName: filling
            FeaturizationMethodParameters:
              futurefill: value
              futurefill_value: "15"
              middlefill: median
              backfill: median

Forecast:
  ForecastTypes:
    - "0.10"
    - "0.50"
    - "0.90"
```

Uninstall the solution

Use one of the following procedures to uninstall the solution.

Using the AWS Management Console

1. Sign in to the AWS CloudFormation console.
2. Select this solution's installation stack.
3. Choose **Delete**.

Using AWS Command Line Interface

Determine whether the AWS Command Line Interface (AWS CLI) is available in your environment. For installation instructions, see What Is the AWS Command Line Interface in the *AWS CLI User Guide*. After confirming that the AWS CLI is available, run the following command.

```
$ aws cloudformation delete-stack --stack-name <your-stack-name>
```

Uninstalling manually

Certain resources will remain after the deletion of the CloudFormation stack. To completely remove the solution, these must be manually removed from the AWS Management Console:

- Any generated Dataset Groups, Datasets, Dataset Import Jobs, Predictors, Forecasts and Forecast Export Jobs in Amazon Forecast.
- The forecast data and Amazon S3 buckets.
- The CloudWatch log groups for all functions created by this solution.

Collection of operational metrics

This solution includes an option to send anonymous operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When enabled, the following information is collected and sent to AWS:

- **Solution ID:** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier for each Improving Forecast Accuracy with Machine Learning deployment
- **Timestamp:** Data-collection timestamp
- **Instance Data:** Information about this stack deployment
  - **Region:** The AWS Region you deployed this stack in
  - **NotebookType:** The Notebook instance Type (for example, `ml.t2.medium`)
  - **NotebookDeployed:** Whether or not the Jupyter Notebook instance was deployed
  - **QuickSightDeployed:** Whether or not automated Amazon QuickSight analysis creation was enabled

AWS owns the data gathered though this survey. Data collection is subject to the AWS Privacy Policy.

To opt out of this feature, modify the AWS CloudFormation template mapping section as follows:

```
Mappings:
  Solution:
    Data:
      SendAnonymousUsageData: 'Yes'
```

to

```
Mappings:
  Solution:
    Data:
      SendAnonymousUsageData: 'No'
```

Source code

Visit the GitHub repository to download the the source files for this solution and to share your customizations with others. The Improving Forecast Accuracy with Machine Learning template is generated using the AWS Cloud Development Kit (AWS CDK). Refer to the README.md file for additional information.

Contributors

The following individuals contributed to this document:

- Paul Shuparski-Miller

Revisions

Date	Change
July 2020	Initial release
October 2020	Release version 1.1.0: Added documentation for setting up visualizations using an Amazon Quicksight analysis. For more information about changes for v1.1.0, refer to the CHANGELOG.md file in the Github repository.
November 2020	Release version 1.2.0: Updated the AWS CloudFormation templates; for more information, refer to the CHANGELOG.md file in the GitHub repository.
March 2021	Release version 1.3.0: Updated the AWS CloudFormation templates and added a demo stack. For more information, refer to the CHANGELOG.md file in the GitHub repository.
April 2021	Release version 1.3.1: Updated Python package versions and removed unused packages. Minor documentation bug fixes. For more information, refer to the CHANGELOG.md file in the GitHub repository.
June 2021	Release version 1.3.2: Upgraded Amazon Athena engine version. For more information, refer to the CHANGELOG.md file in the GitHub repository.
June 2021	Release version 1.3.3: Updated the default notebook instance type to `ml.t3.medium`, and fixed an issue that might cause stack deployments to fail. For more information, refer to the CHANGELOG.md file in the GitHub repository.
August 2021	Document enhancements to better describe architecture flow.

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents AWS current product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. AWS responsibilities and liabilities to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.