Predictive Data Science with Amazon SageMaker and a Data Lake on AWS

Quick Start Reference Deployment

August 2018

Pariveda Solutions, Inc.
AWS Quick Start Team

Contents

Quick Links 2
Overview 3
Predictive Data Science with Amazon SageMaker and a Data Lake on AWS 3
Costs and Licenses 4
Architecture 5
Prerequisites 6
Technical Requirements 6
Specialized Knowledge 7
Deployment Options 7
Deployment Steps 7
Step 1. Prepare Your AWS Account 7
Step 2. Launch the Quick Start 8
Step 3. [OPTIONAL] Test the Deployment with the Demo Scenario Provided 12
Step 4. [OPTIONAL] Train a Model on Your Own 23
Security 35

Access Control 35
Encryption 35
FAQ 36
GitHub Repository 37
Additional Resources 37
Document Revisions 37

This Quick Start was created by Pariveda Solutions, Inc. in collaboration with Amazon Web Services (AWS).

Quick Starts are automated reference deployments that use AWS CloudFormation templates to deploy key technologies on AWS, following AWS best practices.

Quick Links

The links in this section are for your convenience. Before you launch the Quick Start, please review the architecture, security, and other considerations discussed in this guide.

- If you have an AWS account, and you're already familiar with AWS services, you can launch the Quick Start to build the architecture shown in Figure 1. The deployment takes approximately 10-15 minutes. If you're new to AWS or to Predictive Data Science with Amazon SageMaker and a Data Lake on AWS, please review the implementation details and follow the step-by-step instructions provided later in this guide.

Launch

- If you want to take a look under the covers, you can view the AWS CloudFormation template that automates the deployment.

View template

> **Note** This Quick Start does not use a virtual private cloud (VPC) and has a single deployment option.

Overview

This Quick Start reference deployment guide provides step-by-step instructions for deploying Predictive Data Science with Amazon SageMaker and a Data Lake on AWS.

This Quick Start is for users who are looking to harness the power of their data to make predictive and prescriptive models that drive real business value, without needing to configure complex machine learning (ML) hardware clusters.

It uses Amazon SageMaker to train the ML models after data has been prepared, to manage the model outputs, and to deploy endpoints for making predictions (inferences) based on the data.

By launching the Quick Start, you can configure:

- The location of your Amazon Simple Storage Service (Amazon S3) data lake.
- The instance sizes and counts that Amazon SageMaker uses for training and inference.
- The storage location of the model management components.

Predictive Data Science with Amazon SageMaker and a Data Lake on AWS

Amazon SageMaker is a fully managed platform that enables developers and data scientists to quickly and easily build, train, and deploy machine learning models at any scale. Amazon SageMaker removes barriers that typically slow down developers who want to use machine learning. Examples of the barriers that Amazon SageMaker addresses include:

- Provisioning and scaling compute clusters.
- Managing parameter servers.
- Installing ML libraries.
- Capturing ML model artifacts.

Amazon SageMaker also integrates seamlessly with the rest of the AWS platform to provide a complete end-to-end solution for building predictive or prescriptive applications. No ML model can be trained without input data being available. This end-to-end solution includes the following data-pipeline stages that start with raw data:

- **Ingest.** Raw data is stored in a low cost, highly durable data lake that's built on Amazon Simple Storage Service (Amazon S3).
- **Model.** The raw data is normalized to prepare it for ML model training.

- **Enhance.** The data is enhanced to make it more easily usable, for instance by joining with master data to get more detailed context, and/or adding computed columns.
- **Transform.** The data is transformed to the proper input format for the ML algorithm, and divided into training and validation sets to test the effectiveness of the ML model.
- **Deliver.** Amazon SageMaker trains the ML model and hosts it in a predictive endpoint. An API, hosted by Amazon API Gateway, provides a simple REST interface to use the transformed data in the data lake, which simplifies delivering predictions to your applications.

This Quick Start packages all these components, and includes a demo scenario that builds and updates a predictive model for Amazon EC2 Spot pricing. The demo shows how to:

- Take raw data, and store it on a data lake in Amazon S3.
- Transform the data for consumption in Amazon SageMaker.
- Use Amazon SageMaker to build an ML model and host it in a live prediction API.

For more details, see Architecture, later in this guide.

Costs and Licenses

You are responsible for the cost of the AWS services used while running this Quick Start reference deployment. There is no additional cost for using the Quick Start.

The AWS CloudFormation template for this Quick Start includes configuration parameters that you can customize. Some of these settings, such as instance type, will affect the cost of deployment. For cost estimates, see the pricing pages for each AWS service you will be using. Prices are subject to change.

> **Tip** After you deploy the Quick Start, we recommend that you enable the AWS Cost and Usage Report to track costs associated with the Quick Start. This report delivers billing metrics to an S3 bucket in your account. It provides cost estimates based on usage throughout each month, and finalizes the data at the end of the month. For more information about the report, see the AWS documentation.

Because this Quick Start uses native AWS services, no additional licensing is required.

Architecture

Deploying this Quick Start with **default parameters** builds the following Amazon SageMaker and Data Lake environment in the AWS Cloud.

Figure 1: Quick Start architecture for Amazon SageMaker and a Data Lake on AWS

The Quick Start sets up the following:

- A structured data lake in Amazon S3 to hold the raw, modeled, enhanced, and transformed data.
- A staging bucket for the feature engineered and transformed data that will be ingested into Amazon SageMaker.
- Data transformation code hosted on AWS Lambda to prepare the raw data for consumption and ML model training, and to transform data inputs and outputs.
- Amazon SageMaker automation through Lambda functions to build, manage, and create REST endpoints for new models, based on a schedule or triggered by data changes in the data lake.
- API Gateway endpoints to host public APIs for enabling developers to get historical data or predictions for their applications.
- Amazon Kinesis Data Streams to enable real time processing of new data across the Ingest, Model, Enhance, and Transform stages.

- Amazon Kinesis Data Firehose to deliver the results of the Model and Enhance phases to Amazon S3 for durable storage.
- AWS IAM to enforce the principle of least privilege on each processing component. The IAM role and policy restrict access to only the resources that are necessary.
- A demo scenario that builds and updates a predictive model for daily Amazon Elastic Compute Cloud (Amazon EC2) Spot pricing.

The demo scenario of this Quick Start passes through the following stages to perform the tasks that deliver predictions:

- **Ingest.** Periodically reads data from the Amazon EC2 Spot price API, and places raw data into the model stream.
- **Model.** Converts raw data into a common pipeline format.
- **Enhance.** Appends current On-Demand pricing and ML category.
- **Transform.** Performs the following tasks:
  - Aggregate data into buckets of a target size and write to Amazon S3 as a time-series per Availability Zone (AZ) and instance type.
  - Run feature engineering to transform data into a format that Amazon SageMaker can consume.
  - Trigger the SageMaker training cluster to train a new model for each instance type using the Amazon SageMaker DeepAR algorithm, a supervised learning algorithm.
- **Deliver.** Delivers the following APIs:
  - A historical API using historical pricing data, for any span of time, AZ, and instance type.
  - A prediction API using a SageMaker endpoint, which is built with the model created in the Transform phase and the most recent history, pulled from the bucket that contains transformed data.

Prerequisites

Technical Requirements

You can deploy this Quick Start in a new AWS account, with no additional technical requirements.

Specialized Knowledge

Before you deploy this Quick Start, we recommend that you become familiar with the following AWS services. (If you are new to AWS, see Getting Started with AWS.)

- Amazon SageMaker
- AWS Lambda
- Amazon S3.
- Amazon Kinesis Data Streams
- Amazon Kinesis Data Firehose
- Amazon API Gateway
- AWS CloudFormation
- AWS IAM

Deployment Options

This Quick Start only has one deployment option, and does not use a VPC. By launching this Quick Start, you can configure the location of your data lake, the instance sizes and counts used within Amazon SageMaker for training and inference, and the storage location of the model management components.

Deployment Steps

Step 1. Prepare Your AWS Account

1. If you don't already have an AWS account, create one at https://aws.amazon.com by following the on-screen instructions.
2. Use the region selector in the navigation bar to choose the AWS Region where you want to deploy Predictive Data Science with Amazon SageMaker and a Data Lake on AWS. Please note that Amazon SageMaker is not supported in all regions. You can review the Region Table for updated information.
3. If necessary, request a service limit increase for Amazon SageMaker instance types you plan to use. You might need to do this if you already have an existing deployment that uses this instance type, or you think you might exceed the default limit with this deployment.

Step 2. Launch the Quick Start

> **Note** You are responsible for the cost of the AWS services used while running this Quick Start reference deployment. There is no additional cost for using this Quick Start. For full details, see the pricing pages for each AWS service you will be using in this Quick Start. Prices are subject to change.

1. Launch the AWS CloudFormation template into your AWS account.

   The deployment takes about 10-15 minutes to complete.

2. Check the region that's displayed in the upper-right corner of the navigation bar, and change it if necessary. This is where the infrastructure for Data Lake with Amazon SageMaker will be built. The template is launched in the US East (Ohio) Region by default, and can only be launched in a region where SageMaker is supported. You can review the Region Table for updated information on regional support for SageMaker.
3. On the **Select Template** page, keep the default setting for the template URL, and then choose **Next**.
4. On the **Specify Details** page, change the stack name if needed. Review the parameters for the template. Provide values for the parameters that require input. For all other parameters, review the default settings and customize them as necessary. When you finish reviewing and customizing the parameters, choose **Next**.

In the following tables, parameters are listed by category.

- **Parameters for deploying Amazon SageMaker and a Data Lake on AWS**

View template

General Settings:

Parameter label (name)	Default	Description
Resource Suffix (StackSuffix)	*Requires input*	The suffix appended to all resources in the stack. This allows multiple copies of the same stack to be created in the same account.

Amazon S3 Data and Artifact Repositories

Use a common prefix for all parameters in this section, so that you can re-use a single bucket for all four repositories.

Parameter label (name)	Default	Description
Lambda Code S3 Bucket (LambdaCodeS3Bucket)	*Requires input*	Prefix of the S3 bucket where Lambda code will be copied to (within your account). The final bucket name will be the value of this parameter concatenated with a hyphen and resource suffix value.
Data Lake S3 Bucket (DataLakeS3BucketName)	*Requires input*	Prefix of the S3 bucket that will serve as the data lake. The final bucket name will be the value of this parameter concatenated with a hyphen and resource suffix value.
SageMaker Feature Staging Bucket (SageMakerInputS3BucketName)	*Requires input*	Prefix of the S3 bucket where SageMaker input data is created by the data preparation Lambda function. The bucket name will be the value of this parameter concatenated with a hyphen and resource suffix value.
SageMaker Model Repository Bucket (SageMakerModelS3BucketName)	*Requires input*	Prefix of the S3 bucket where SageMaker models are saved after training. The bucket name will be the value of this parameter concatenated with a hyphen and resource suffix value.

Delivery APIs:

Parameter label (name)	Default	Description
SageMaker Endpoint Instance Count (InitialEndpointInstanceCount)	1	The number of instances the Auto Scaling group for the SageMaker endpoint begins with.

Parameter label (name)	Default	Description
SageMaker Endpoint Instance Type (EndpointInstanceType)	ml.t2.medium	The instance type to run the SageMaker endpoint on. Check the documentation at https://docs.aws.amazon.com/sagemaker/latest/dg/algos.html for your chosen algorithm for instance type recommendations.

Amazon SageMaker Configuration:

Parameter label (name)	Default	Description
SageMaker Algorithm (SageMakerAlgorithm)	DeepARForecasting	Name of the SageMaker ML algorithm that will be trained.
SageMaker Hyper Parameters (SageMakerHyperParameters)	{"time_freq":"H","context_length":"72","prediction_length":"24","epochs":"20"}	The hyperparameters that will be passed to the SageMaker training algorithm, in .json format. See the HyperParameters section under the algorithm you are using.
SageMaker Endpoint Name (SageMakerEndpointName)	spot-price-predictions	Name of the SageMaker endpoint that is run to generate predictions.
SageMaker Training Data Bucket (SageMakerInputDataPrefix)	features	S3 Key Prefix (within the SageMakerInputS3Bucket) where SageMaker training/testing data is created by the data preparation Lambda function. SageMaker will point to this location to find the training and test data channels.
SageMaker Model S3 Key Prefix (SageMakerOutputModelPrefix)	models	S3 Key Prefix (within the SageMakerModelS3Bucket) where the SageMaker model is saved after training.
SageMaker Jupyter Notebook Instance Type (NotebookInstanceType)	ml.t2.large	The instance type for the Exploratory Data Analysis notebook. Choose **none** to skip creating the notebook.
SageMaker Training Instance Type (TrainingInstanceType)	ml.t2.medium	The instance type used to train the SageMaker model. Check the documentation for your chosen algorithm for instance type recommendations.
SageMaker Training Instance Count (TrainingInstanceCount)	1	The number of instances used to train the SageMaker model. Ensure that the algorithm you have chosen supports parallel training before setting the instance count to a number higher than 1.
SageMaker Training Instance Volume Size (GB) (TrainingInstanceVolumeSize)	20	The EBS volume size (in GB) to attach to each instance that trains the SageMaker model.

Parameter label (name)	Default	Description
SageMaker Training Timeout (seconds) (TrainingMaxRuntimeSeconds)	3600	The maximum number of seconds to allow a SageMaker training job to run.

AWS Quick Start Configuration:

Parameter label (name)	Default	Description
Quick Start S3 Bucket Name (QSS3BucketName)	aws-quickstart	The S3 bucket you have created for your copy of Quick Start assets, if you decide to customize or extend the Quick Start for your own use. The bucket name can include numbers, lowercase letters, uppercase letters, and hyphens, but should not start or end with a hyphen.
Quick Start S3 Key Prefix (QSS3KeyPrefix)	quickstart-datalake-pariveda/	The S3 key name prefix used to simulate a folder for your copy of Quick Start assets, if you decide to customize or extend the Quick Start for your own use. This prefix can include numbers, lowercase letters, uppercase letters, hyphens (-), and forward slashes (/).

5. On the **Options** page, you can specify tags (key-value pairs) for resources in your stack and set advanced options. When you're done, choose **Next**.
6. On the **Review** page, review and confirm the template settings. Under **Capabilities**, select the check box to acknowledge that the template will create IAM resources.
7. Choose **Create** to deploy the stack.
8. Monitor the status of the stack. When the status is **CREATE_COMPLETE**, the cluster is ready.
9. Use the URLs displayed in the **Resources** tab for the stack to view the Lambda, Amazon S3, and IAM resources that were created.
10. See the URLs displayed in the **Outputs** tab for the stack, as in Figure 2, which shows:
    - The URLs to call for the API endpoints interact with the platform.
    - The S3 paths for the data lake, ML features, and SageMaker model artifacts.

Overview | Outputs | Resources | Events | Template | Parameters | Tags | Stack Policy | Change Sets | Rollback Triggers

Key	Value	Description
DataLakeBucket	imet-sagemaker-demo-data-lake-ab	The name of the Data Lake Bucket
SageMakerModelArtifactLocation	s3://imet-sagemaker-demo-ml-model-ab/models	The location in S3 where the serialized models th...
PredictEndpointUrl	https://1zymkjpiq3.execute-api.us-east-1.amazonaws.com/LATEST/predict	The URL to call for the prediction endpoint
SageMakerTrainingFeaturesLocation	s3://imet-sagemaker-demo-ml-input-ab/sagemaker-input	The location in S3 where training features for S3 t...
DeliverEndpointUrl	https://1zymkjpiq3.execute-api.us-east-1.amazonaws.com/LATEST/deliver	The URL to call for the historical data delivery en...

Figure 2: Outputs created

Step 3. [OPTIONAL] Test the Deployment with the Demo Scenario Provided

You can use the demo to test the deployment, which involves the following tasks:

1. Explore the data set
2. Train the model
3. Get predictions from the model

Explore the Data Set

1. You will already have a version of the pipeline created with your stack suffix. To see the stacks created:
   a. Open the AWS Management Console, and go to CloudFormation.
   b. Filter the list of stacks to your stack suffix (i.e. -ab).
   c. You will see four stacks created, as shown in Figure 3—one to provision the Lambda code, one for the Ingest/Model/Enhance/Transform (IMET) section of the pipeline, one for the Deliver phase, and one for the SageMaker training resources.

Figure 3: Stacks that are created

2. Additionally, historical data for EC2 Spot pricing is populated into your data lake bucket. To see this data:
   a. Open the AWS Management Console, and go to Amazon S3.
   b. Filter the bucket list to your stack suffix (i.e. -ab).
   c. Only the data-lake bucket will have any data in it at this point. Click the bucket to see the Model, Enhance, and Transform sections.
   d. Click Transform, and choose one of the Availability Zones (AZs).
   e. You will see files with the name of EC2 instance types.
   f. Click a file, download it, and open it in a text editor, as shown in Figure 4.

Figure 4: Downloading a file of a given instance type

g. You will see the Spot price history for the selected instance type and AZ, stored in the format for the DeepAR algorithm, as shown in Figure 5.

Figure 5: Viewing the Spot price history

3. Pull Spot history using the API that is created:
   a. Open the AWS Management Console, and go to API Gateway
   b. Find and select the API with your stack suffix (using CTRL+F to bring up the browser search bar).
   c. Choose the /deliver - GET endpoint, and then choose **Test**, as shown in Figure 6.

Figure 6: Getting Spot history

d. Enter the following query string: `availabilityZone=us-east-1a&instanceType=p2.16xlarge&startDate=2018-01-01T00:00:00&endDate=2018-04-01T00:00:00`

e. Choose **Test**. The Spot history by hour for the instance type you selected should be returned.

f. Change the AZ or instance type, and then choose **Test**, as shown in Figure 7. A different Spot history for the instance type you selected should be returned.

Figure 7: Testing additional Spot history

Train the Model

1. Trigger data preparation and model training:
   a. Open the AWS Management Console, and go to AWS Lambda.
   b. Search for the keywords DataPrep and your stack suffix.
   c. Click through to the function (i.e. SageMakerDataPrepLambda-ab).
   d. Choose **Select a test event**, and then choose **Configure test events**, as shown in Figure 8.

Figure 8: Triggering data preparation

   e. Search "Scheduled" and leave the default event, as shown in Figure 9.

Figure 9: Configuring a test event

f. Choose **Create**, and then choose **Test** after the popup closes.

2. Find and review the feature engineered data:

   a. Open the Amazon S3 console.

   b. Filter the bucket list to your stack suffix (i.e. -ab).

   c. Choose the bucket imet-sagemaker-demo-ml-input-<stack suffix>. You should see an input folder with the current day's date. The folder should have a test and train folder, each with a .json file, as shown in Figure 10.

Figure 10: Reviewing the feature engineered data

d. Download the test .json file, and open it in a text editor. You will see that it matches the input format specified for the DeepAR algorithm.

3. Find the model training Lambda function that was triggered after data preparation:

   a. Creating the feature engineered data triggers an S3 event called SageMaker Training Job.

   b. Open the AWS Management Console, and go to AWS Lambda.

   c. Search for the keywords "TrainingKickoff "and your stack suffix.

   d. Click through to the function (i.e. SageMakerTrainingKickoffLambda-ab.)

   e. Choose the **Monitoring** tab for getting Amazon CloudWatch metrics in a dashboard, as shown in Figure 11.

Figure 11: Getting CloudWatch metrics

f. Choose **View CloudWatch Logs**, and expand the "Received event" line (second line in Figure 12) to view the S3 event that triggered the function.

g. Expand to find the ID for the SageMaker job that was created, as shown in Figure 12.

Figure 12: Viewing the ID for the SageMaker job that was created

4. Find and review the SageMaker training job:

   a. In a new tab, open the AWS Management Console, and go to Amazon SageMaker.

   b. Choose **Training Jobs**.

   c. Choose the job "training-<datetime>" that matches the ID from the logs.

d. Review the job settings, as shown in Figure 13, and look for the code on the Lambda function that created the settings.

Figure 13: Reviewing the job settings

e. Scroll to the bottom of the Job screen, and then choose **View logs** to view the logs from the SageMaker training cluster. If the job is still running, let it finish.

f. When the job finishes, it will output model-scoring metrics into the CloudWatch logs (RMSE, Quantile Loss, etc.), as shown in Figure 14.

```
[07/26/2018 08:02:34 INFO 140374342735680] #test_score (algo-1, RMSE): 0.054256177305
[07/26/2018 08:02:34 INFO 140374342735680] #test_score (algo-1, mean_wQuantileLoss): 0.0161211
[07/26/2018 08:02:34 INFO 140374342735680] #test_score (algo-1, wQuantileLoss[0.1]): 0.00981048
[07/26/2018 08:02:34 INFO 140374342735680] #test_score (algo-1, wQuantileLoss[0.2]): 0.0151153
[07/26/2018 08:02:34 INFO 140374342735680] #test_score (algo-1, wQuantileLoss[0.3]): 0.0184072
[07/26/2018 08:02:34 INFO 140374342735680] #test_score (algo-1, wQuantileLoss[0.4]): 0.0201189
[07/26/2018 08:02:34 INFO 140374342735680] #test_score (algo-1, wQuantileLoss[0.5]): 0.0206258
[07/26/2018 08:02:34 INFO 140374342735680] #test_score (algo-1, wQuantileLoss[0.6]): 0.0198028
[07/26/2018 08:02:34 INFO 140374342735680] #test_score (algo-1, wQuantileLoss[0.7]): 0.0178069
[07/26/2018 08:02:34 INFO 140374342735680] #test_score (algo-1, wQuantileLoss[0.8]): 0.0145113
[07/26/2018 08:02:34 INFO 140374342735680] #test_score (algo-1, wQuantileLoss[0.9]): 0.00889108
[07/26/2018 08:02:34 INFO 140374342735680] #quality_metric: host=algo-1, test RMSE <loss>=0.054256177305
[07/26/2018 08:02:34 INFO 140374342735680] #quality_metric: host=algo-1, test mean_wQuantileLoss <loss>=0.0161210857332
```

Figure 14: Viewing model-scoring metrics

g. When the job finishes, refresh the SageMaker job screen. The output should be populated with an S3 link to the model that was built, as shown in Figure 15.

Figure 15: Viewing the link to the S3 model artifact

h. You can opt to download the .tar.gz file, and expand it to view the binary and .json files that make up the model artifact.

Get Predictions from the Model

1. Find the SageMaker endpoint for the new model:

   a. Note that creating the model artifact triggers another Lambda function to update a SageMaker endpoint with the latest model.

   b. On the AWS Lambda console, go back to all functions, and search EndpointUpdate and your stack suffix.

   c. Click through to the function, observe that it has been run, and choose **Jump to logs** to view the logs.

   d. Within the logs, expand to see the S3 event that triggered the function, the creation of a SageMaker model record, the creation of a SageMaker endpoint configuration and either creation or update of the SageMaker endpoint, as shown in Figure 16.

```
2018-07-26T08:15:29.995Z 0df90c59-90ac-11e8-9b47-530d8209253f Successfully created model:
{
    "ModelArn": "arn:aws:sagemaker:us-east-1:613652601873:model/ec2-spot-data-model-2018-07-26t07-57-05"
}

03:15:30    2018-07-26T08:15:30.324Z 0df90c59-90ac-11e8-9b47-530d8209253f Successfully created endpoint configuration: {"Endpoi

2018-07-26T08:15:30.324Z 0df90c59-90ac-11e8-9b47-530d8209253f Successfully created endpoint configuration:
{
    "EndpointConfigArn": "arn:aws:sagemaker:us-east-1:613652601873:endpoint-config/ec2-spot-data-endpoint-config-2018-07-26t07-57-05"
}

03:15:30    2018-07-26T08:15:30.520Z 0df90c59-90ac-11e8-9b47-530d8209253f No endpoint found. Will create new endpoint.
03:15:30    2018-07-26T08:15:30.520Z 0df90c59-90ac-11e8-9b47-530d8209253f Creating endpoint with params: {"EndpointConfigNam
03:15:30    2018-07-26T08:15:30.813Z 0df90c59-90ac-11e8-9b47-530d8209253f Successfully created endpoint: {"EndpointArn":"arn:a

2018-07-26T08:15:30.813Z 0df90c59-90ac-11e8-9b47-530d8209253f Successfully created endpoint:
{
    "EndpointArn": "arn:aws:sagemaker:us-east-1:613652601873:endpoint/ec2-spot-data-endpoint-ab"
}
```

Figure 16: Viewing the S3 event that triggered actions

e. In the Amazon SageMaker console, inspect the created model, endpoint configuration, and endpoint, as shown in Figure 17.

Figure 17: Inspecting the model, endpoint configuration, and endpoint

2. Call the predictions API to get Spot price predictions:
   a. Open the AWS Management Console, and go to API Gateway.
   b. Find and choose the API with your stack suffix (using CTRL+F to bring up the browser search bar).
   c. Select the Predict - GET endpoint, and then choose **Test**.
   d. Enter an Availability Zone, instance type, and start date and end date that are within the next 24 hours (i.e. `availabilityZone=us-east-1a&instanceType=p2.16xlarge&startTime=<startTime>&endTime=<endTime>`).

e. Choose **Test**. The Spot predictions for the instance type that you selected should be returned, as shown in Figure 18.

Figure 18: Viewing the Spot predictions for the specified instance type

Step 4. [OPTIONAL] Train a Model on Your Own

At a high level, training a model involves the following tasks:

- Upload raw data to the data lake bucket
- Open an existing notebook instance

  OR
- Create a notebook instance
- Explore the data in your notebook
- Prepare the dataset for training
- Create a SageMaker training job
- Create a SageMaker hyperparameter optimization job
- Move the SageMaker trained model to production

After you complete these tasks, you are ready to use the /predict endpoint on your API to get live predictions from your new model.

Upload raw data

1. Create a new space for your data under the model folder in your data lake bucket.

a. Open the S3 console to the DataLakeS3Bucket, and you will see three folders (model, enhance, and transform).

b. Click into the model folder, and then click **Create folder**, as shown in Figure 19.

Figure 19: Creating a folder

c. Enter a folder name that describes your new model.

2. Upload raw data into the model folder in the data lake bucket, as shown in Figure 20.

   a. Open the new folder, and then choose **Upload**.

   b. Select the files you are interested in building your predictive model from.

   c. Leave the default permissions, and then choose **Next**.

   d. Select the appropriate encryption option (at the least, the Amazon S3 master-key is recommended), and then choose **Next**.

Figure 20: Uploading raw data into the model folder

d. To add your source files, choose **Upload**.

Open an existing notebook instance

If you selected an instance type for the NotebookInstanceType parameter when launching the Quick Start, Amazon SageMaker creates a notebook instance for a Jupyter notebook. To open the Jupyter notebook for this instance, choose **Open**, as shown in Figure 21.

Figure 21: Finding the notebook instance created by the Quick Start

Create a notebook instance

1. If you selected **none** for the NotebookInstanceType parameter when launching the Quick Start, choose **Create a notebook instance**, as shown in Figure 22, to perform exploratory analysis on your data.

Figure 12: Create SageMaker notebook instance

2. Provide a name for your notebook instance, as shown in Figure 23.
3. Select the notebook instance type. The instance size must be large enough to hold the data in memory.
4. Choose **Create a new role** for notebook instances to call other services such as Amazon SageMaker and Amazon S3, since the roles created for the Quick Start stack provide restricted access.

Figure 23: Choosing the instance type and creating a new role

5. Enter the name of the data lake bucket—in this case, imet-sagemaker-demo-data-lake-ab, as shown in Figure 24. You can find the name in the **Outputs** tab for the stack, as shown earlier in Figure 2.

Figure 24: Specifying the data lake bucket for the IAM role

6. Click **Create role** to have SageMaker create the IAM Role and Policy on your behalf.
7. Back on the Create Notebook Instance page, leave the other options as the default and choose **Create Notebook instance**. You will see that your instance is in a Pending state as SageMaker prepares it, as shown in Figure 25.

Figure 25: Viewing the details of your notebook instance

8. Once your instance has launched, choose **Open** to securely launch the Jupyter Notebook UI.

Explore the data in your notebook

> **Note** It is beyond the scope of this guide to provide details on data exploration using Jupyter. For more information, see:
>
> - Python Notebook from Martin Seeler
> - Advanced Jupyter Notebook Tricks from Domino Data Lab
> - Jupyter Notebooks Demonstration from the Coursera Course by Arizona State: Introduction to Data Exploration and Visualization course

1. Set up the notebook to add imports for enabling the rest of the notebook, as shown in Figure 26. Also, prepare it to read from Amazon S3, and to read one of the files, as shown in Figure 27.

```
In [10]: bucket = 'imet-sagemaker-demo-data-lake-ab'
         prefix = 'transform'

         # Define IAM role
         import boto3
         import re
         from sagemaker import get_execution_role

         role = get_execution_role()

WARNING:sagemaker:Couldn't call 'get_role' to get Role ARN from role name datalake-sagemaker-demo-a-SageMakerNotebook
Executi-IGW4KP3LVD05 to get Role path.
```

Now we'll import the Python libraries we'll need.

```
In [2]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import io
        import os
        import time
        import json
        import sagemaker.amazon.common as smac
        import sagemaker
        from sagemaker.predictor import csv_serializer, json_deserializer
        import s3fs
```

Figure 26: Basic setup (imports) in Jupyter

Let's setup the notebook to read from S3

```
In [3]: fs = s3fs.S3FileSystem()
```

And load one of our files.

```
In [14]: files = fs.ls(f"{bucket}/{prefix}/us-east-1a/")
         with fs.open(files[0], 'rb') as file:
             content = str(file.read())
             print(content)
```

```
ull,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,nul
l,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,
null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,nu
ll,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null
,null,null,null,null,null,null,null,null,null,null,null,null,"0.109200",null,null,null,null,null,null,null,null,null,null,null,
null,null,null,null,null,null,null,null,null,null,null,null,null,null,"0.109200",null,null,null,null,null,null,null,null,null,n
ull,null,null,null,null,null,null,null,null,null,null,null,null,null,null,"0.109200",null,null,null,null,null,null,null,nu
ll,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,"0.109200",null,null,null,null,null,null,null
l,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null
,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,"0.109200",null,
null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,"
0.109200",null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,n
ull,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,nu
ll,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,nul
l,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null
,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,
null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,n
ull,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null
,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,
null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null
,"0.109200"],"cat":3}'
```

Figure 27: Viewing data in Jupyter

2. After initial setup, load your data from Amazon S3 into the notebook, as shown in Figure 28. Once data is loaded, you can join, clean, visualize, and analyze the data as needed using the standard Python data tools (NumPy, pandas, matplotlib, etc.).

```
In [38]: instance_history = json.loads(content)
         file = files[0].replace(bucket, '')
         labels = pd.date_range(start=instance_history['start'], periods=len(instance_history['target']), freq='1H')
         values = pd.Series(instance_history['target'], dtype=float, index=labels).fillna(method='ffill')
         values.plot(title=f'spot price for {file} (category {instance_history["cat"]})')

Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x7f28d85d5b70>
```

Figure 28: Visualizing data in Jupyter

Prepare the dataset for training

1. After you've found the right tabular input dataset, prepare it for training by using Amazon SageMaker:

   a. Perform one-hot or label encoding on categorical variables.

b. Convert data to Protobuf-RecordIO format to improve performance.

c. Split the dataset under train and test prefixes in Amazon S3.

d. Use S3FS to save the results to the sandbox folder in your data lake bucket, as shown in Figure 29.

Out[54]:

	0	date_id_DAYOFWEEK	date_id_MONTH	date_id_WEEK	date_id_DAY
2017-12-02 20:00:00+00:00	0.1221	5	12	48	2
2017-12-02 21:00:00+00:00	0.1221	5	12	48	2
2017-12-02 22:00:00+00:00	0.1221	5	12	48	2
2017-12-02 23:00:00+00:00	0.1221	5	12	48	2
2017-12-03 00:00:00+00:00	0.1232	6	12	48	3

In [59]:
```
with fs.open(f"{bucket}/sandbox/train/c3.2xlarge.csv", 'wb') as obj:
    bytes_to_write = data.to_csv(None, index=False, header=False).encode()
    obj.write(bytes_to_write)
```

Figure 29: Saving data to Amazon S3 from Jupyter

e. View the file in the Amazon S3 console, as shown in Figure 30.

Figure 30: Viewing the data file from Jupyter in Amazon S3

Create a SageMaker training job

You can now create a SageMaker training job to process your data. The inputs for the training job will vary based on the algorithm you train.

Within the SageMaker notebook, you can find examples for each algorithm under the **SageMaker Examples** tab, as shown in Figure 31.

Figure 31: SageMaker example notebooks

As you train and validate your model, use the notebook to visualize the resulting predictions. This is an iterative process of improving your model and viewing the results.

Create an hyperparameter optimization job

Once you have results based on your manual feature engineering, you can also create a SageMaker Hyperparameter Optimization Job to find the optimal set of hyperparameters for your algorithm.

Move the model to production

Once you have a SageMaker trained model from your experimentation, move it to production:

1. Update the transform Lambda function to generate the tabular input dataset from your notebook (as discussed earlier in Explore the data in your notebook).
   a. This generates the transformed files for historical lookups and for ML inputs.
   b. If you need a historical data API, you can also update the deliver Lambda function to read and return data from these files.
2. Update the data prep Lambda function to run the same data preparation for the algorithm you will be training. (You can use the code from your notebook as discussed earlier in Prepare the dataset for training.)
   – This function reads the transform data and writes the SageMaker input into the SageMakerInputS3Bucket.
   – The results being written trigger SageMakerTrainingKickoffLambda.
3. Update the Quick Start master stack by creating a Change Set that selects the algorithm you will be training, and specify the correct hyperparameters (from the Hyperparameter Optimization Job discussed earlier). Figures 32-36 show the required tasks involved.

Figure 32: Creating a change set for the stack

   a. Choose **Use current template**, as shown in Figure 33.

Figure 33: Using the current template

b. Choose the algorithm and hyperparameters, as shown in Figure 34.

Figure 34: Updating with algorithm and hyperparameters

c. Accept the defaults, and choose **Create change set**, as shown in Figure 35.

Figure 35: Using the defaults and clicking through to Create change set

d. Choose **Execute**, as shown in Figure 36. This runs the change set to update the stack.

Figure 36: Running the change set to update the stack

- This updates SageMakerTrainingKickoffLambda and SageMakerEndpointUpdateLambda with the proper image for the new algorithm.
- SageMakerTrainingKickoffLambda will create a SageMaker training cluster to train the selected algorithm over the data in the SageMakerInputS3Bucket using the hyperparameters specified.
- The resulting model is stored in the SageMakerModelS3Bucket, and an endpoint is automatically generated for the model by SageMakerEndpointUpdateLambda.

e. Update the /predict GET endpoint in API Gateway to take in the proper inputs for your new algorithm by updating the query string parameters and updating the mapping (as shown in Figures 37 and 38, respectively).

Figure 37: Updating the query string parameters

Figure 38: Updating the API mapping

f. Update the PredictApiLambda to generate the proper input format for your predictive endpoint.

   – You can find the proper format for your algorithm using the Inference Formats section of the Algorithm Description from the SageMaker documentation (You will have to click into the specific algorithm you are using to find these details).

   – You can load historical data from the /transform section of the Data Lake to look up the proper values for features that were not specified by the user

At this point, you should be able to use the /predict endpoint on your API to get live predictions from your new model.

Security

Access Control

AWS IAM roles are used to ensure that the compute resources (Lambda functions, SageMaker training clusters, and inference endpoints) can only access the necessary S3 data and AWS resources.

Encryption

All objects (data, models, and Lambda code) stored in Amazon S3 in this Quick Start are AES256 encrypted at rest using S3 server-side encryption.

FAQ

Q. I encountered a CREATE_FAILED error when I launched the Quick Start.

A. If AWS CloudFormation fails to create the stack, we recommend that you relaunch the template with **Rollback on failure** set to **No**. (This setting is under **Advanced** in the AWS CloudFormation console, **Options** page.) With this setting, the stack's state will be retained and the instance will be left running, so you can troubleshoot the issue. (Look at the log files in `%ProgramFiles%\Amazon\EC2ConfigService` and `C:\cfn\log`.)

> **Important** When you set **Rollback on failure** to **No**, you will continue to incur AWS charges for this stack. Please make sure to delete the stack when you finish troubleshooting.

For additional information, see Troubleshooting AWS CloudFormation on the AWS website.

Q. I encountered a size limitation error when I deployed the AWS CloudFormation templates.

A. We recommend that you launch the Quick Start templates from the links in this guide or from another S3 bucket. If you deploy the templates from a local copy on your computer or from a non-S3 location, you might encounter template size limitations when you create the stack. For more information about AWS CloudFormation limits, see the AWS documentation.

Q. What is the real-world hypothesis for the predictive application demo?

A. As an AWS DevOps engineer, I want to save my company money using Spot pricing. I need to be confident, however, that my Spot instances won't be terminated if I'm outbid.

Q. Who will use the predictions from the ML model, and how?

A. The IT department will use these predictions to choose instance type, to bid on Spot instances, and to decide whether to use Spot or On-Demand pricing.

Q. How can a company benefit from the predictive application demo?

A. Benefits include lower costs and less infrastructure management.

Q. What is the data-science hypothesis?

A. Using historic and recent pricing data, we can predict the upcoming Spot prices for a given instance type in a specific AZ.

Q. What specific values are predicted?

A. Predictions include the mean price and the 90th and 10th percentile price of an instance in a particular AZ for a given time span.

Q. What data sources are used to make the predictions?

A. Data sources include the Spot price API and the AWS price ledger.

GitHub Repository

You can visit our GitHub repository to download the templates and scripts for this Quick Start, to post your comments, and to share your customizations with others.

Additional Resources

AWS services

- Amazon SageMaker
- AWS Lambda
- Amazon Kinesis Data Streams
- Amazon Kinesis Data Firehose
- Amazon S3
- Amazon API Gateway
- AWS CloudFormation
- AWS IAM for roles and policies

Amazon SageMaker

- Amazon SageMaker documentation
  https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html

Quick Start reference deployments

- AWS Quick Start home page
  https://aws.amazon.com/quickstart/

Document Revisions

Date	Change	In sections
August 2018	Initial publication	—

© 2018, Amazon Web Services, Inc. or its affiliates, and Pariveda Solutions, Inc. All rights reserved.

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS 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.

The software included with this paper is licensed under the Apache License, Version 2.0 (the "License"). You may not use this file except in compliance with the License. A copy of the License is located at http://aws.amazon.com/apache2.0/ or in the "license" file accompanying this file. This code is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.