



Build, Train, and Deploy a Machine Learning Model

with Amazon SageMaker

In this tutorial, you will learn how to use [Amazon SageMaker](#) to build, train, and deploy a machine learning (ML) model. We will use the popular XGBoost ML algorithm for this exercise. Amazon SageMaker is a modular, fully managed machine learning service that enables developers and data scientists to build, train, and deploy ML models at scale.

Taking ML models from conceptualization to production is typically complex and time-consuming. You have to manage large amounts of data to train the model, choose the best algorithm for training it, manage the compute capacity while training it, and then deploy the model into a production environment. Amazon SageMaker reduces this complexity by making it much easier to build and deploy ML models. After you choose the right algorithms and frameworks from the wide range of choices available, it manages all of the underlying infrastructure to train your model at petabyte scale, and deploy it to production.

In this tutorial, you will assume the role of a machine learning developer working at a bank. You have been asked to develop a machine learning model to predict whether a customer will enroll for a certificate of deposit (CD). The model will be trained on the marketing dataset that contains information on customer demographics, responses to marketing events, and external factors.

The data has been labeled for your convenience and a column in the dataset identifies whether the customer is enrolled for a product offered by the bank. A version of this dataset is [publicly available](#) from the ML repository curated by the University of California, Irvine. This tutorial implements a supervised machine learning model since the data is labeled. (Unsupervised learning occurs when the datasets are not labeled.)

In this tutorial, you will:

1. Create a notebook instance





3. Train the model to learn from the data

4. Deploy the model

5. Evaluate your ML model's performance

The resources created and used in this tutorial are AWS free tier eligible. Remember to complete Step 7 and terminate your resources. If your account has been active with these resources for longer than two months, your account will be charged less than \$0.50.

This tutorial requires an AWS account

Create a Free Account

The resources you create in this tutorial are Free Tier eligible.

[More about the Free Tier >>](#)

Step 1. Enter the Amazon SageMaker console

Navigate to the Amazon SageMaker console.

When you [click here](#), the AWS Management Console will open in a new window, so that you can keep this step-by-step guide open. Begin typing **SageMaker** in the search bar and select **Amazon SageMaker** to open the service console.





AWS Management Console

AWS services

Find Services

You can enter names, keywords or acronyms.



Amazon **SageMaker**

Build, Train, and Deploy Machine Learning Models

Recently visited services



Amazon SageMaker



Rekognition



Billing



EC2



S3

► All services

Access resources on the go



Access the Management Console using the AWS Console Mobile App. [Learn more](#)

Explore AWS

Visit AWS around the world at a Summit

AWS Global Summits bring the cloud computing community together to connect, collaborate, and learn about AWS. [Learn more](#)

Data Lake Storage

Build your data lake on the most secure, durable, and scalable storage. [Learn more](#)

Amazon SageMaker

Machine learning for every developer and data scientist. [Learn more](#)

Run Serverless Containers with AWS Fargate

AWS Fargate runs and scales your containers without having to manage servers or clusters. [Learn more](#)

Build a solution

Get started with simple wizards and automated workflows.

Launch a virtual machine

With EC2

2-3 minutes



Build a web app

With Elastic Beanstalk

6 minutes



Build using virtual servers

With Lightsail

1-2 minutes

Connect an IoT device

With AWS IoT

5 minutes

Have feedback?

(click to enlarge)

Step 2. Create an Amazon SageMaker notebook instance

In this step, you will create an Amazon SageMaker notebook instance.





(click to enlarge)

2b. On the **Create notebook instance** page, enter a name in the **Notebook instance name** field. This tutorial uses *MySageMakerInstance* as the instance name, but you can choose a different name, if desired.

For this tutorial, you can keep the default **Notebook instance type** of *ml.t2.medium*.





create a role with the required permissions and assign it to your instance. Alternately, you can choose an existing IAM role in your account for this purpose.

Amazon SageMaker > Notebook instances > Create notebook instance

Create notebook instance

Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. [Learn more](#)

Notebook instance settings

Notebook instance name

MySageMakerInstance

Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Notebook instance type

ml.t2.medium

Elastic Inference [Learn more](#)

none

► Additional configuration

Permissions and encryption

IAM role

Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the [AmazonSageMakerFullAccess](#) IAM policy attached.

Choose an IAM role

Create a new role

Enter a custom IAM role ARN

Use existing role

Encryption key - optional

Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.

No Custom Encryption

(click to enlarge)





2c. In the **Create an IAM role** box, select **Any S3 bucket**. This allows your Amazon SageMaker instance to access all S3 buckets in your account. Later in this tutorial, you'll be creating a new S3 bucket. However, if you have a bucket you'd want to use instead, select **Specific S3 buckets** and specify the name of the bucket.

Choose **Create role**.

Create an IAM role

Passing an IAM role gives Amazon SageMaker permission to perform actions in other AWS services on your behalf. Creating a role here will grant permissions described by the [AmazonSageMakerFullAccess](#) IAM policy to the role you create.

The IAM role you create will provide access to:

- ☒ **S3 buckets you specify - optional**
 - ☐ Specific S3 buckets

Example: bucket-name-1, bucket-name-2, bucket-name-3

Comma delimited. ARNs, "*" and "/" are not supported.
 - ☒ **Any S3 bucket**
Allow users that have access to your notebook instance access to any bucket and its contents in your account.
 - ☐ None
- ☒ Any S3 bucket with "sagemaker" in the name
- ☒ Any S3 object with "sagemaker" in the name
- ☒ Any S3 object with the tag "sagemaker" and value "true" [See Object tagging](#)
- ☒ S3 bucket with a Bucket Policy allowing access to SageMaker [See S3 bucket policies](#)

[Cancel](#) [Create role](#)

(click to enlarge)

2d. Notice that Amazon SageMaker created a role called *AmazonSageMaker-ExecutionRole-**** for you.





Permissions and encryption

IAM role
Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the [AmazonSageMakerFullAccess](#) IAM policy attached.

AmazonSageMaker-ExecutionRole-20190816T163711 ▼

✓ Success! You created an IAM role.
[AmazonSageMaker-ExecutionRole-20190816T163711](#) ↗

Root access - optional

☒ Enable - Give users root access to the notebook

☐ Disable - Don't give users root access to the notebook
Lifecycle configurations always have root access

Encryption key - optional
Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.

No Custom Encryption ▼

► **Network - optional**

► **Git repositories - optional**

► **Tags - optional**

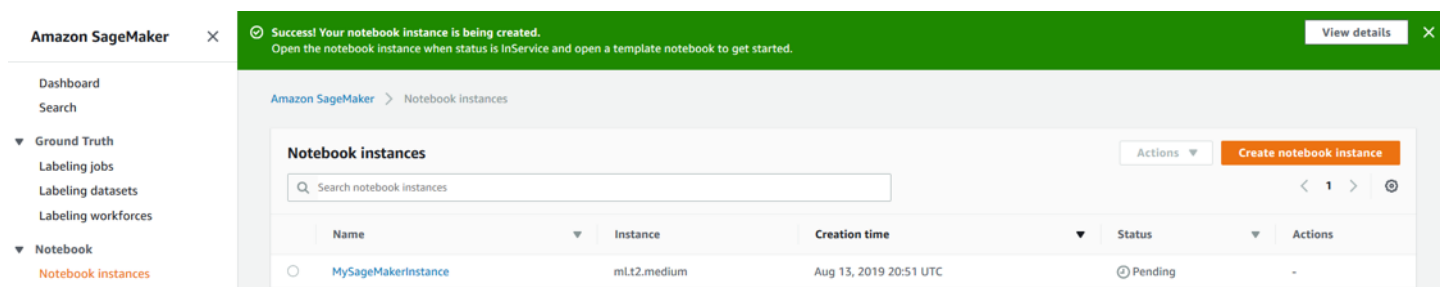
Cancel

Create notebook Instance

(click to enlarge)

2e. On the **Notebook instances** page, you should see your new *MySageMakerInstance* notebook instance in **Pending** status.





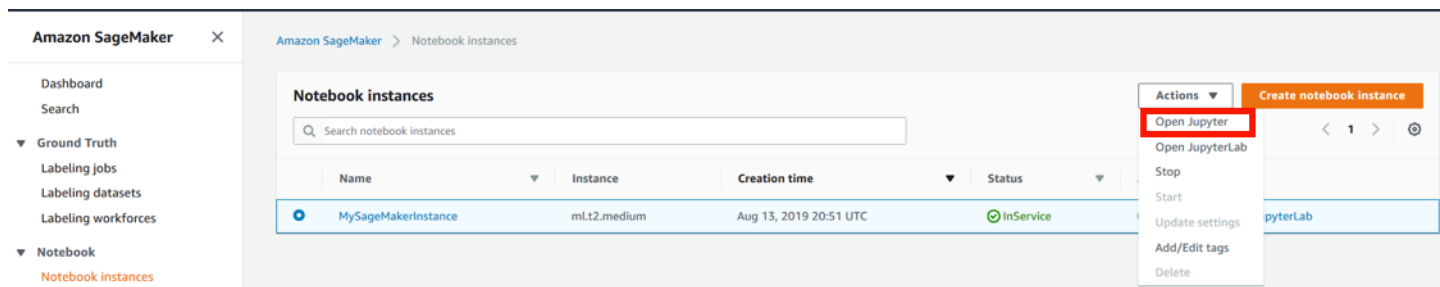
(click to enlarge)

Step 3. Prepare the data

In this step you will use your Amazon SageMaker notebook to preprocess the data that you need to train your machine learning model.

3a. On the **Notebook instances** page, wait until *MySageMakerInstance* has transitioned from **Pending** to **InService** status.

After the status is **InService**, select *MySageMakerInstance* and open it using the **Actions** drop down menu, or by choosing **Open Jupyter** next to the **InService** status.



(click to enlarge)





Search notebook instances					
	Name	Instance	Creation time	Status	Actions
▼ Ground Truth Labeling jobs Labeling datasets Labeling workforces	MySageMakerInstance	ml.t2.medium	Aug 13, 2019 20:51 UTC	InService	Open Jupyter Open JupyterLab

(click to enlarge)

3b. After Jupyter opens, from the **Files** tab, choose **New** and then choose **conda_python3**.

The screenshot shows the JupyterLab interface. At the top, there's a search bar and a menu icon. Below that, a table lists notebook instances. The 'Files' tab is selected. The 'New' button is highlighted, and a dropdown menu is open showing various notebook templates. 'conda_python3' is highlighted in the dropdown menu.

Name	Instance	Creation time	Status	Actions
MySageMakerInstance	ml.t2.medium	Aug 13, 2019 20:51 UTC	InService	Open Jupyter Open JupyterLab

Files Running Clusters SageMaker Examples Conda

Select items to perform actions on them.

Upload New

Notebook:

- R (Beta)
- Sparkmagic (PySpark)
- Sparkmagic (Spark)
- Sparkmagic (SparkR)
- conda_amazonei_mxnet_p27
- conda_amazonei_mxnet_p36
- conda_amazonei_tensorflow_p27
- conda_amazonei_tensorflow_p36
- conda_chainer_p27
- conda_chainer_p36
- conda_mxnet_p27
- conda_mxnet_p36
- conda_python2
- conda_python3**
- conda_pytorch_p27
- conda_pytorch_p36
- conda_tensorflow_p27
- conda_tensorflow_p36

Other:

- Text File
- Folder
- Terminal

(click to enlarge)



following code into the code cell in your instance and select **Run**.

While the code runs, an ***** appears between the square brackets as pictured in the first screenshot to the right. After a few seconds, the code execution will complete, the ***** will be replaced with the number **1**, and you will see a success message as pictured in the second screenshot to the right.

Python

```
1  # import libraries
2  import boto3, re, sys, math, json, os, sagemaker, urllib.request
3  from sagemaker import get_execution_role
4  import numpy as np
5  import pandas as pd
6  import matplotlib.pyplot as plt
7  from IPython.display import Image
8  from IPython.display import display
9  from time import gmtime, strftime
10 from sagemaker.predictor import csv_serializer
11
12 # Define IAM role
13 role = get_execution_role()
14 prefix = 'sagemaker/DEMO-xgboost-dm'
15 containers = {'us-west-2': '433757028032.dkr.ecr.us-west-2.amazonaws.com/xgboost:lat
16               'us-east-1': '811284229777.dkr.ecr.us-east-1.amazonaws.com/xgboost:lat
17               'us-east-2': '825641698319.dkr.ecr.us-east-2.amazonaws.com/xgboost:lat
18               'eu-west-1': '685385470294.dkr.ecr.eu-west-1.amazonaws.com/xgboost:lat
19 my_region = boto3.session.Session().region_name # set the region of the instance
20 print("Success - the MySageMakerInstance is in the " + my_region + " region. You wil
```





```
In [*]: import boto3, re, sys, math, json, os, sagemaker
from sagemaker import get_execution_role
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import Image
from IPython.display import display
from time import gmtime, strftime
from sagemaker.predictor import csv_serializer

# Define IAM role
role = get_execution_role()
prefix = 'sagemaker/DEMO-xgboost-dm'
containers = {'us-west-2': '433757028032.dkr.ecr.us-west-2.amazonaws.com/xgboost:latest',
              'us-east-1': '811284229777.dkr.ecr.us-east-1.amazonaws.com/xgboost:latest',
              'us-east-2': '825641698319.dkr.ecr.us-east-2.amazonaws.com/xgboost:latest',
              'eu-west-1': '685385470294.dkr.ecr.eu-west-1.amazonaws.com/xgboost:latest'} # each region has its XGBoost

my_region = boto3.session.Session().region_name # set the region of the instance
print("Success - the MySageMakerInstance is in the " + my_region + " region. You will use the " + containers[my_region])
```

```
In [ ]:
```

(click to enlarge)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

+ % % Run Code

```
In [1]: import boto3, re, sys, math, json, os, sagemaker
from sagemaker import get_execution_role
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import Image
from IPython.display import display
from time import gmtime, strftime
from sagemaker.predictor import csv_serializer

# Define IAM role
role = get_execution_role()
prefix = 'sagemaker/DEMO-xgboost-dm'
containers = {'us-west-2': '433757028032.dkr.ecr.us-west-2.amazonaws.com/xgboost:latest',
              'us-east-1': '811284229777.dkr.ecr.us-east-1.amazonaws.com/xgboost:latest',
              'us-east-2': '825641698319.dkr.ecr.us-east-2.amazonaws.com/xgboost:latest',
              'eu-west-1': '685385470294.dkr.ecr.eu-west-1.amazonaws.com/xgboost:latest'} # each region has its XGBoost

my_region = boto3.session.Session().region_name # set the region of the instance
print("Success - the MySageMakerInstance is in the " + my_region + " region. You will use the " + containers[my_region])
```

```
Success - the MySageMakerInstance is in the us-east-1 region. You will use the dkr.ecr.us-east-1.amazonaws.com/xgboost:latest container for your SageMaker endpoint.
```

```
In [ ]:
```

(click to enlarge)

3d. In this step, you create an S3 bucket that will store your data for this tutorial.





restrictions and limitations.

Select **Run**. If you don't receive a success message, change the bucket name and try again.

Python

```
1 bucket_name = 'your-s3-bucket-name' # <--- CHANGE THIS VARIABLE TO A UNIQUE NAME FOR
2 s3 = boto3.resource('s3')
3 try:
4     if my_region == 'us-east-1':
5         s3.create_bucket(Bucket=bucket_name)
6     else:
7         s3.create_bucket(Bucket=bucket_name, CreateBucketConfiguration={ 'LocationCons
8         print('S3 bucket created successfully')
9 except Exception as e:
10    print('S3 error: ',e)
```

```
In [2]: bucket_name = 'testyourname' # <--- change this variable to a unique name for your bucket
s3 = boto3.resource('s3')
try:
    if my_region == 'us-east-1':
        s3.create_bucket(Bucket=bucket_name)
    else:
        s3.create_bucket(Bucket=bucket_name, CreateBucketConfiguration={ 'LocationConstraint': my_region })
    print('S3 bucket created successfully')
except Exception as e:
    print('S3 error: ',e)
```

S3 bucket created successfully

(click to enlarge)

3e. Next, you need to download the data to your Amazon SageMaker instance and load it into a dataframe. Copy and **Run** the following code:





```
1 try:
2     urllib.request.urlretrieve ("https://d1.awsstatic.com/tmt/build-train-deploy-machi
3     print('Success: downloaded bank_clean.csv.')
4 except Exception as e:
5     print('Data load error: ',e)
6
7 try:
8     model_data = pd.read_csv('./bank_clean.csv',index_col=0)
9     print('Success: Data loaded into dataframe.')
10 except Exception as e:
11     print('Data load error: ',e)
```

```
In [3]: try:
        urllib.request.urlretrieve ("https://d1.awsstatic.com/tmt/build-train-deploy-machine-learning-model-sagemaker/bank_cl
        print('Success: downloaded bank_clean.csv.')
    except Exception as e:
        print('Data load error: ',e)

    try:
        model_data = pd.read_csv('./bank_clean.csv',index_col=0)
        print('Success: Data loaded into dataframe.')
    except Exception as e:
        print('Data load error: ',e)
```

```
Success: downloaded bank_clean.csv.
Success: Data loaded into dataframe.
```

(click to enlarge)

3f. Now, we will shuffle the data and split it into training data and test data.

The **training data** (70% of customers) will be used during the model training loop. We will use gradient-based optimization to iteratively refine the model parameters. Gradient-based optimization is a way to find model parameter values that minimize the model error, using the gradient of the model loss function.

The **test data** (remaining 30% of customers) will be used to evaluate the performance of the model, and measure how well the trained model generalizes to unseen data.

Copy the following code into a new code cell and select **Run** to shuffle and split the data:



```
1 train_data, test_data = np.split(model_data.sample(frac=1, random_state=1729), [int(
2 print(train_data.shape, test_data.shape)
```

```
In [4]: train_data, test_data = np.split(model_data.sample(frac=1, random_state=1729), [int(0.7 * len(model_data))])
print(train_data.shape, test_data.shape)
(28831, 61) (12357, 61)
```

(click to enlarge)

Step 4. Train the model from the data

In this step, you will train your machine learning model with the training dataset.

4a. To use an Amazon SageMaker pre-built XGBoost model, you will need to reformat the header and first column of the training data and load the data from the S3 bucket.

Copy the following code into a new code cell and select **Run** to reformat and load the data:

Python

```
1 pd.concat([train_data['y_yes'], train_data.drop(['y_no', 'y_yes'], axis=1)], axis=1)
2 boto3.Session().resource('s3').Bucket(bucket_name).Object(os.path.join(prefix, 'train
3 s3_input_train = sagemaker.s3_input(s3_data='s3://{}/{}/train'.format(bucket_name, p
```





new code cell and select **Run**:

Python

```
1 sess = sagemaker.Session()
2 xgb = sagemaker.estimator.Estimator(containers[my_region],role, train_instance_count
3 xgb.set_hyperparameters(max_depth=5,eta=0.2,gamma=4,min_child_weight=6,subsample=0.8
```

4c. With the data loaded and the XGBoost estimator set up, train the model using gradient optimization on a *ml.m4.xlarge* instance by copying the following code into the next code cell and selecting **Run**.

After a few minutes, you should start to see the training logs being generated.

Python

```
1 xgb.fit({'train': s3_input_train})
```

In [7]: `xgb.fit({'train': s3_input_train})`

```
[92]#011train-error:0.095314
[17:36:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10 extra nodes, 14 pruned nodes, max_depth=5
[93]#011train-error:0.095314
[17:36:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 24 extra nodes, 30 pruned nodes, max_depth=5
[94]#011train-error:0.095314
[17:36:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 24 pruned nodes, max_depth=3
[95]#011train-error:0.095314
[17:36:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 12 extra nodes, 30 pruned nodes, max_depth=5
[96]#011train-error:0.095279
[17:36:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 18 extra nodes, 12 pruned nodes, max_depth=5
[97]#011train-error:0.094828
[17:36:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 4 extra nodes, 22 pruned nodes, max_depth=2
[98]#011train-error:0.094863
[17:36:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 30 extra nodes, 12 pruned nodes, max_depth=5
[99]#011train-error:0.094759

2019-08-15 17:36:34 Uploading - Uploading generated training model
2019-08-15 17:36:34 Completed - Training job completed
Billable seconds: 56
```

In []:

(click to enlarge)



Step 5. Deploy the model

In this step, you will deploy the trained model to an endpoint, reformat then load the CSV data, then run the model to create predictions.

5a. To deploy the model on a server and create an endpoint that you can access, copy the following code into the next code cell and select **Run**:

Python

```
1 xgb_predictor = xgb.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge')
```

```
In [9]: xgb_predictor = xgb.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge')
INFO:sagemaker:Creating model with name: xgboost-2018-07-13-14-29-39-425
INFO:sagemaker:Creating endpoint with name xgboost-2018-07-13-14-25-03-272
-----!
```

(click to enlarge)

5b. To predict whether customers in the test data enrolled for the bank product or not, copy the following code into the next code cell and select **Run**:

Python

```
1 test_data_array = test_data.drop(['y_no', 'y_yes'], axis=1).values #load the data in
2 xgb_predictor.content_type = 'text/csv' # set the data type for an inference
3 xgb_predictor.serializer = csv_serializer # set the serializer type
4 predictions = xgb_predictor.predict(test_data_array).decode('utf-8') # predict!
5
```




```
In [20]: test_data_array = test_data.drop(['y_no', 'y_yes'], axis=1).values #load the data into an array
xgb_predictor.content_type = 'text/csv' # set the data type for an inference
xgb_predictor.serializer = csv_serializer # set the serializer type
predictions = xgb_predictor.predict(test_data_array).decode('utf-8') # predict!
predictions_array = np.fromstring(predictions[1:], sep=',') # and turn the prediction into an array
print(predictions_array.shape)
```

(12357,)

(click to enlarge)

Step 6. Evaluate model performance

In this step, you will evaluate the performance and accuracy of the machine learning model.

6a. Copy and paste the code below and select **Run** to compare actual vs. predicted values in a table called a *confusion matrix*.

Based on the prediction, we can conclude that you predicted a customer will enroll for a certificate of deposit accurately for 90% of customers in the test data, with a precision of 65% (278/429) for enrolled and 90% (10,785/11,928) for didn't enroll.

Python

```
1 cm = pd.crosstab(index=test_data['y_yes'], columns=np.round(predictions_array), rown
2 tn = cm.iloc[0,0]; fn = cm.iloc[1,0]; tp = cm.iloc[1,1]; fp = cm.iloc[0,1]; p = (tp+
3 print("\n{0:<20}{1:<4.1f}%\n".format("Overall Classification Rate: ", p))
4 print("{0:<15}{1:<15}{2:>8}".format("Predicted", "No Purchase", "Purchase"))
5 print("Observed")
6 print("{0:<15}{1:<2.0f}% ({2:<}){3:>6.0f}% ({4:<})".format("No Purchase", tn/(tn+fn)
7 print("{0:<16}{1:<1.0f}% ({2:<}){3:>7.0f}% ({4:<}) \n".format("Purchase", fn/(tn+fn)
```



```
In [12]: cm = pd.crosstab(index=test_data['y_yes'], columns=np.round(predictions_array), rownames=['Observed'], colnames=['Predicted'])
tn = cm.iloc[0,0]; fn = cm.iloc[1,0]; tp = cm.iloc[1,1]; fp = cm.iloc[0,1]; p = (tp+tn)/(tp+tn+fp+fn)*100
print("\n{0:<20}{1:<4.1f}%\n".format("Overall Classification Rate: ", p))
print("{0:<15}{1:<15}{2:>8}".format("Predicted", "No Purchase", "Purchase"))
print("Observed")
print("{0:<15}{1:<2.0f}% ({2:<}){3:>6.0f}% ({4:<})".format("No Purchase", tn/(tn+fn)*100,tn, fp/(tp+fp)*100, fp))
print("{0:<16}{1:<1.0f}% ({2:<}){3:>7.0f}% ({4:<}) \n".format("Purchase", fn/(tn+fn)*100,fn, tp/(tp+fp)*100, tp))
```

Overall Classification Rate: 89.5%

Predicted	No Purchase	Purchase
Observed		
No Purchase	90% (10785)	35% (151)
Purchase	10% (1143)	65% (278)

(click to enlarge)

Step 7. Terminate your resources

In this step, you will terminate your Amazon SageMaker-related resources.

Important: Terminating resources that are not actively being used reduces costs and is a best practice. Not terminating your resources will result in a charge.

7a. To delete the Amazon SageMaker endpoint and the objects in your S3 bucket, copy, paste and **Run** the following code:

Python

```
1 sagemaker.Session().delete_endpoint(xgb_predictor.endpoint)
2 bucket_to_delete = boto3.resource('s3').Bucket(bucket_name)
3 bucket_to_delete.objects.all().delete()
```





```
Out[22]: [{'ResponseMetadata': {'RequestId': '28E42783694CB6C0',  
    'HostId': 'j7ajjzyseZHs5ruIrbmaKuXeJh/I7EjYGs9VADB9wY5HTbuG24AFetOjta603cWo39XPXBkf7XA=',  
    'HTTPStatusCode': 200,  
    'HTTPHeaders': {'x-amz-id-2': 'j7ajjzyseZHs5ruIrbmaKuXeJh/I7EjYGs9VADB9wY5HTbuG24AFetOjta603cWo39XPXBkf7XA=',  
    'x-amz-request-id': '28E42783694CB6C0',  
    'date': 'Thu, 15 Aug 2019 21:21:43 GMT',  
    'connection': 'close',  
    'content-type': 'application/xml',  
    'transfer-encoding': 'chunked',  
    'server': 'AmazonS3'},  
    'RetryAttempts': 0},  
    'Deleted': [{'Key': 'sagemaker/DEMO-xgboost-dm/output/xgboost-2019-08-15-18-39-29-757/output/model.tar.gz'},  
    {'Key': 'sagemaker/DEMO-xgboost-dm/train/train.csv'}]]]
```

(click to enlarge)

You have learned how to use Amazon SageMaker to prepare, train, deploy and evaluate a machine learning model. Amazon SageMaker makes it easy to build ML models by providing everything you need to quickly connect to your training data and select the best algorithm and framework for your application, while managing all of the underlying infrastructure, so you can train models at petabyte scale.





Was this tutorial helpful?

Yes

No

[Have more feedback? Tell us here](#)

[Sign In to the Console](#)

Learn About AWS

What Is AWS?

Resources for AWS

Getting Started

Training and Certification

AWS Solutions Portfolio



Developers on AWS

Developer Center

SDKs & Tools

.NET on AWS

[What Is a Container?](#)[Analyst Reports](#)[PHP on AWS](#)[What Is a Data Lake?](#)[AWS Partner Network](#)[Javascript on AWS](#)[AWS Cloud Security](#)[What's New](#)[Blogs](#)[Press Releases](#)

Help

[Contact Us](#)[AWS Careers](#)[File a Support Ticket](#)[Knowledge Center](#)[AWS Support Overview](#)[Legal](#)[Sign In to the Console](#)

Amazon is an Equal Opportunity Employer: *Minority / Women / Disability / Veteran / Gender Identity / Sexual Orientation / Age.*

Language

[عربي |](#)[Bahasa Indonesia |](#)[Deutsch |](#)[English |](#)[Español |](#)[Français |](#)[Italiano |](#)[Português |](#)[Tiếng Việt |](#)[Türkçe |](#)[Русский |](#)[ไทย |](#)[日本語 |](#)



[中文 \(繁體\)](#)

[Privacy](#)



[Site Terms](#)



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

