



Project Report

Fall – 2020

LAB: MACHINE LEARNING ON AWS

Aarya Shekhaar Tilekar

U36766567

Guided by Dr Phil Ventura, Professor, University of South Florida

Lab: Machine Learning on AWS

INTRODUCTION:

With the shift towards Artificial Intelligence, AWS along with the broadest and deepest set of machine learning and AI services, has put machine learning in the hands of every developer. The revolution of AI in the last few decades has taken a different approach. It was originally called computational intelligence (neural nets, fuzzy sets, evolutionary computing) and now is typically called machine learning. Here, one takes big data sets that embody successful (human) activities and then trains general purposes algorithms (usually neural nets) on these data sets so that they can also be successful on new data [1].

Machine learning is always taking a snapshot of something and then "learning" the features in the snapshot. As the snapshot changes, humans annotate and re-annotate, train (or more likely refine) a model, and release it to production again. Thus, any unexpected contextual issues that arise and were not seen in that snapshot will pose new challenges - in the form of false negative or positives [1].

Amazon SageMaker is an Amazon Machine Learning service that enables developers and data scientists to quickly and easily build, train, and deploy machine learning models at any scale. It removes the complexity that gets in the way of successfully implementing machine learning across use cases and industries - from running models for real-time fraud detection, to virtually analyzing biological impacts of potential drugs, to predicting stolen-base success in baseball [2].

LAB OBJECTIVE:

Build, Train and Deploy a Machine Learning Model with Amazon SageMaker

In this lab, 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.

Note: This tutorial requires an AWS account (the AWS Educate account does not give all the required permissions for the training of the Machine Learning Model). The resources you create in this tutorial are Free Tier eligible [3].

In this tutorial, you will perform following tasks:

1. Create a notebook instance
2. Prepare the data
3. Train the model to learn from the data
4. Deploy the model
5. Evaluate your ML model's performance

MOTIVATION:

It is typically complex and time-consuming to take ML models from conceptualization to production; there is a need to manage large amounts of data for training the model, choosing best algorithm for training it and managing the compute capacity while training it and finally deploying the model into a production environment [3].

Amazon SageMaker reduces this complexity by making it much easier to build and deploy ML models, once the right algorithms and frameworks are chosen, it manages all of the underlying infrastructure to train the machine learning model at petabyte scale, and deploy it to production [3].

BACKGROUND:

In this lab, you assume that the machine learning developer is an employee working at a bank. Your task is to develop a machine learning model to predict whether a customer will enrol 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 [3].

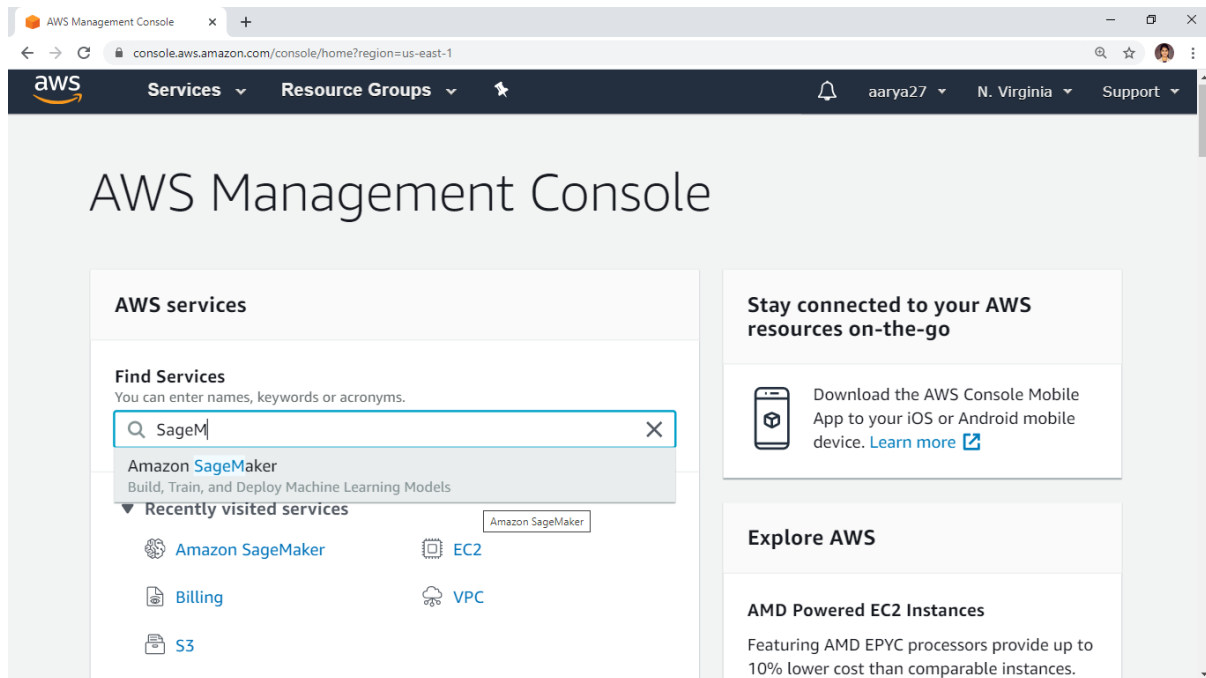
DATASET:

The data has been labelled 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 labelled. (Unsupervised learning occurs when the datasets are not labelled.)[3]

STEPS:

I. ENTER THE AMAZON SAGEMAKER CONSOLE

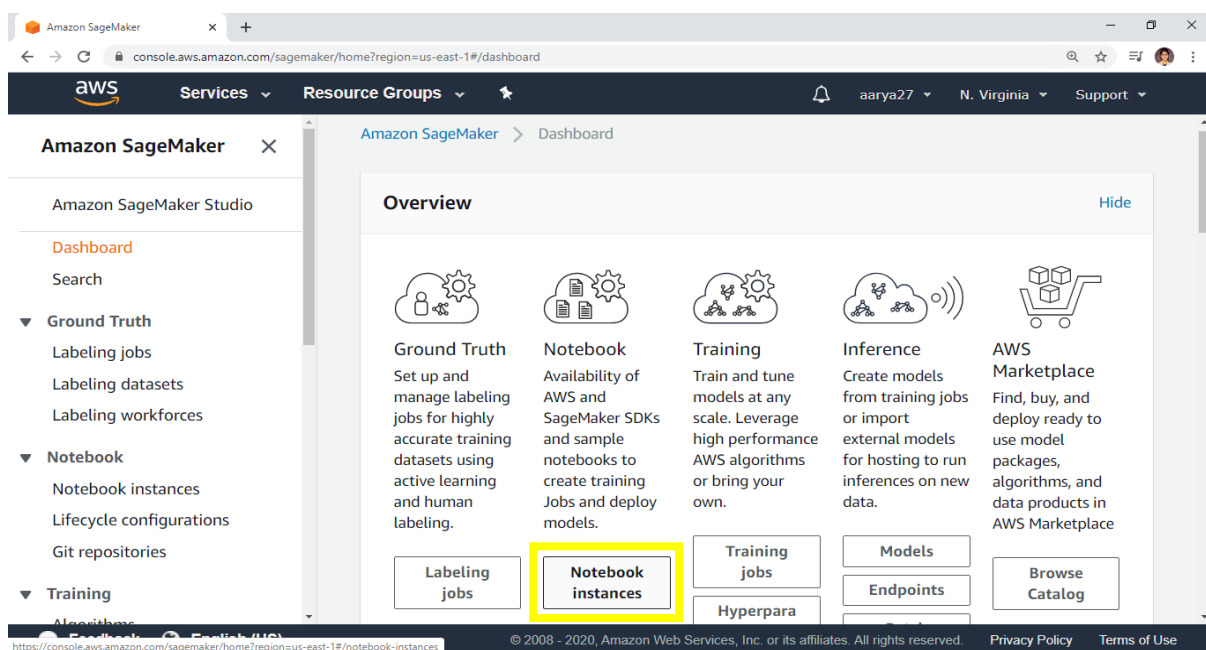
Navigate to the Amazon SageMaker [Console](#). Begin typing *SageMaker* in the search bar and select **Amazon SageMaker** to open the service console.



II. CREATE AN AMAZON SAGEMAKER NOTEBOOK INSTANCE

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

- a. From the Amazon SageMaker dashboard, select **Notebook instances**.



b. On the **Create notebook instance** page, enter a name in the **Notebook instance name** field.

This lab uses *MySageMakerProject* 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**.

To enable the notebook instance to access and securely upload data to Amazon S3, an IAM role must be specified. In the **IAM role** field, choose **Create a new role** to have Amazon SageMaker 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

console.aws.amazon.com/sagemaker/home?region=us-east-1#/notebook-instances/create

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

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

Notebook instance type

Elastic Inference [Learn more](#)

► Additional configuration

Permissions and encryption

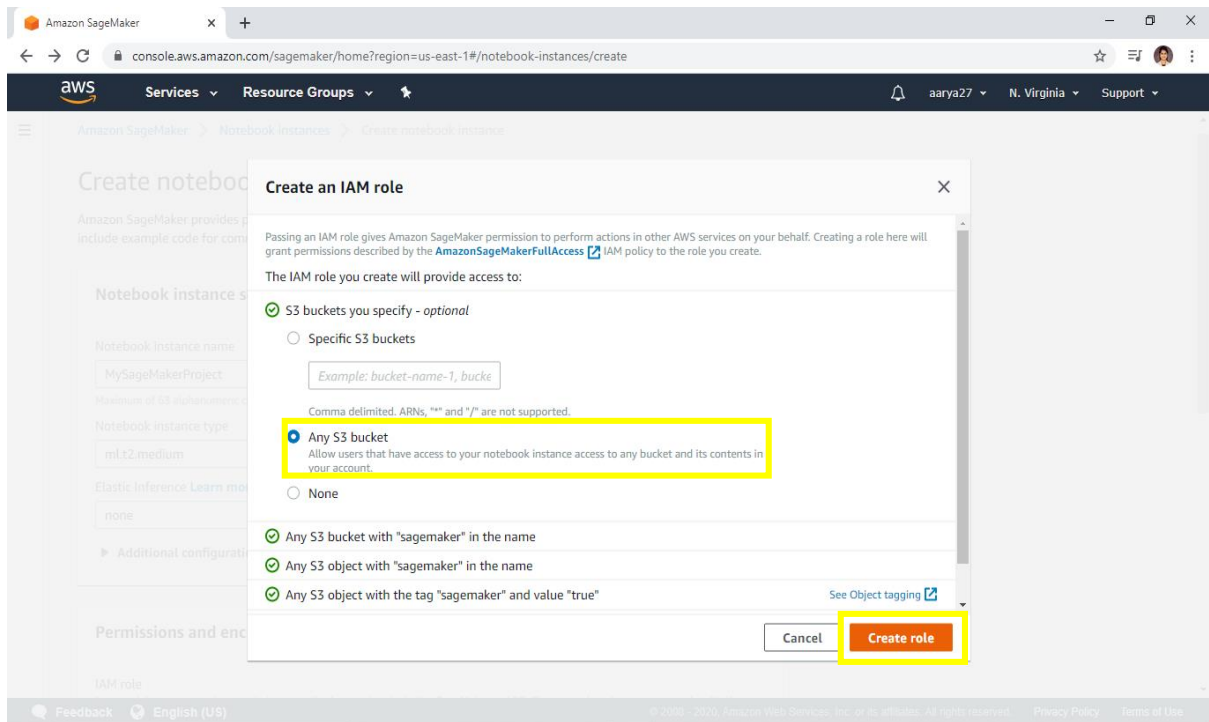
IAM role
Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the [AmazonSageMakerFullAccess](#) IAM policy attached.

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

Feedback English (US)

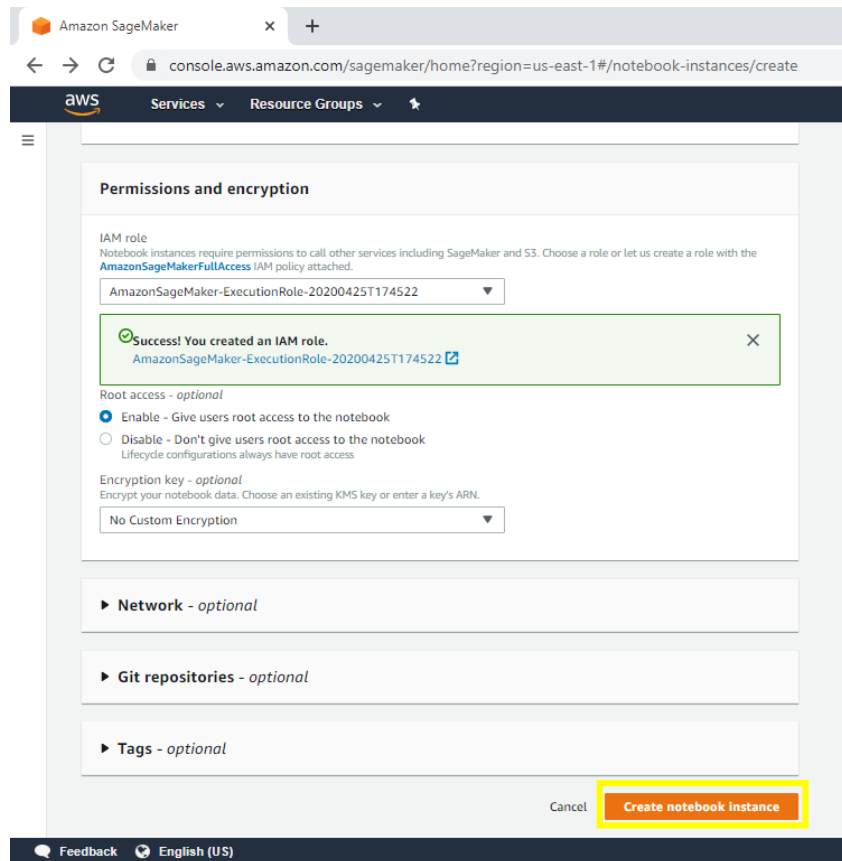
c. In the **Create an IAM role** box, select **Any S3 bucket**. This allows your Amazon SageMaker instance to access all S3 buckets in your account. Later in this tutorial, you'll be creating a new S3 bucket. However, if you have a bucket you'd want to use instead, select **Specific S3 buckets** and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

Choose **Create role**.



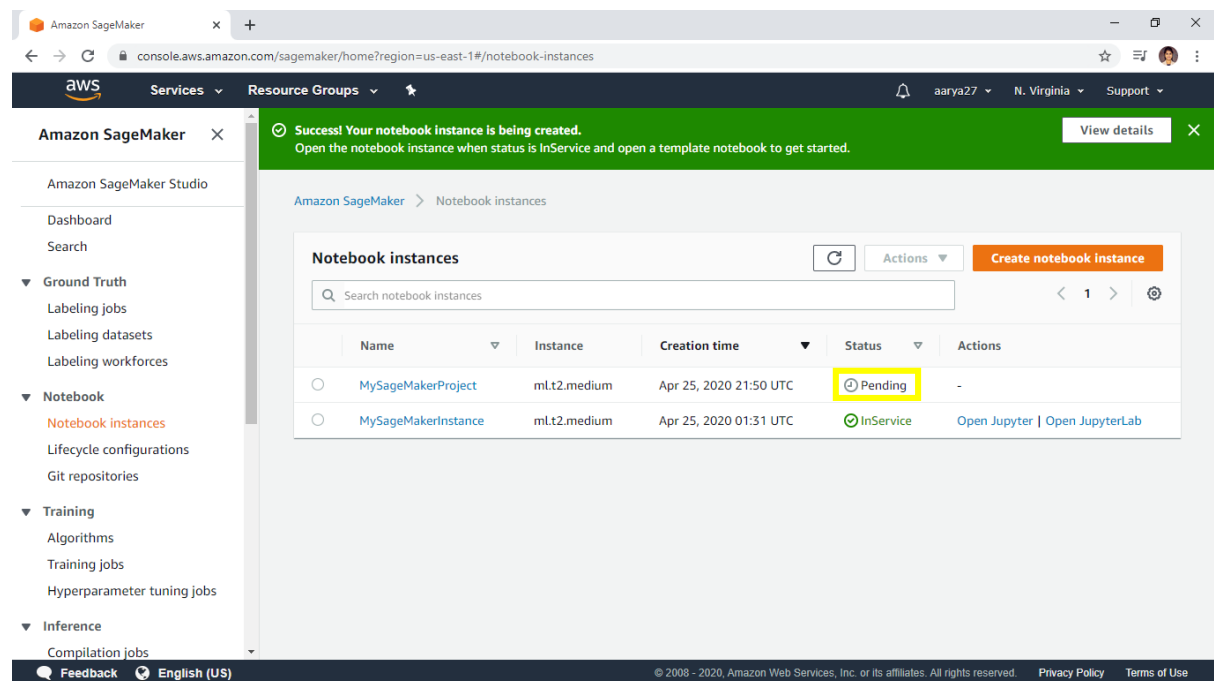
d. Notice that Amazon SageMaker created a role called *AmazonSageMaker-ExecutionRole-**** for you.

For this tutorial, we will use the default values for the other fields. Choose **Create notebook instance**.



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

Your notebook instance should transition from Pending to InService status in less than two minutes.

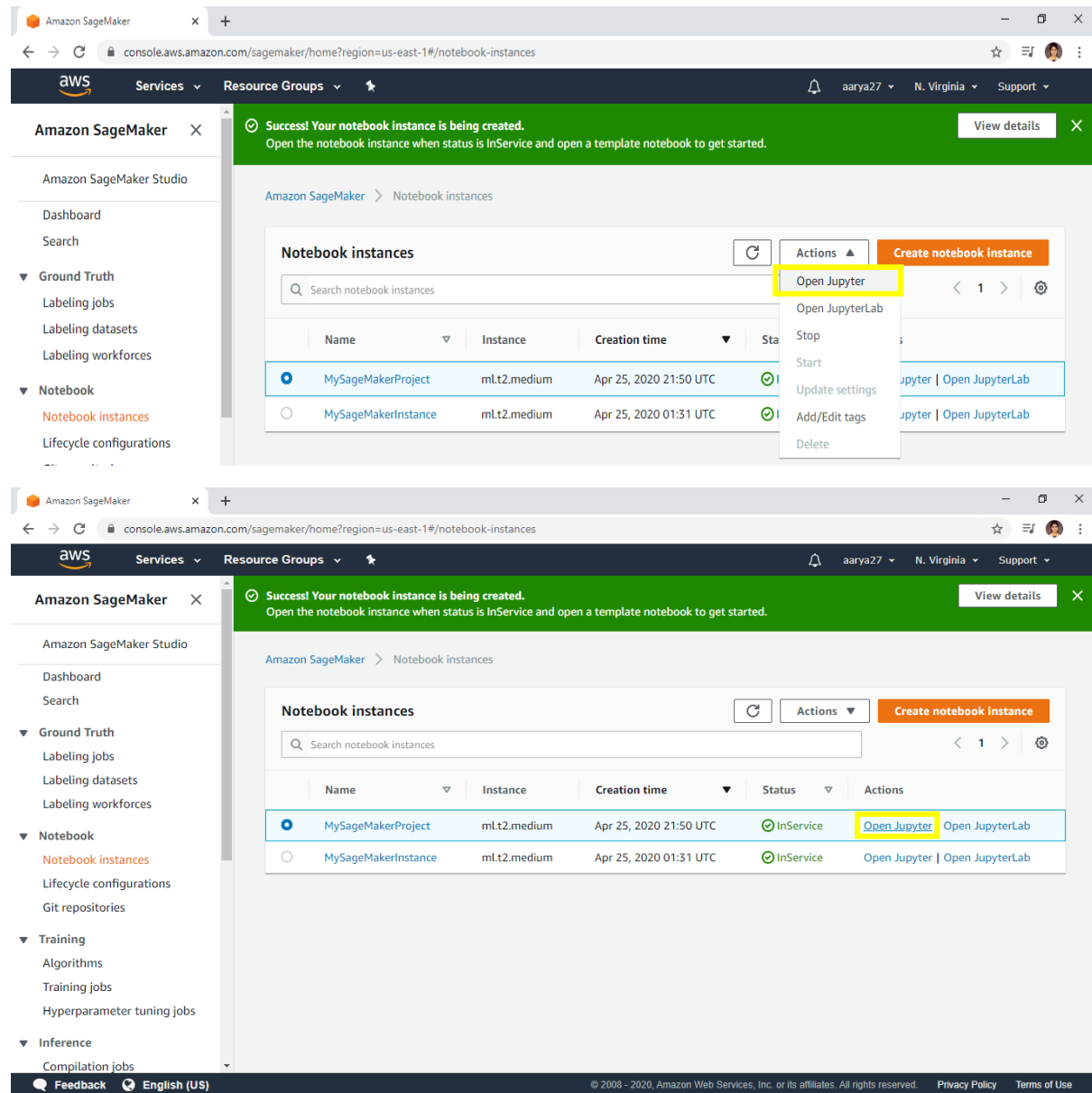


III. PREPARE THE DATA

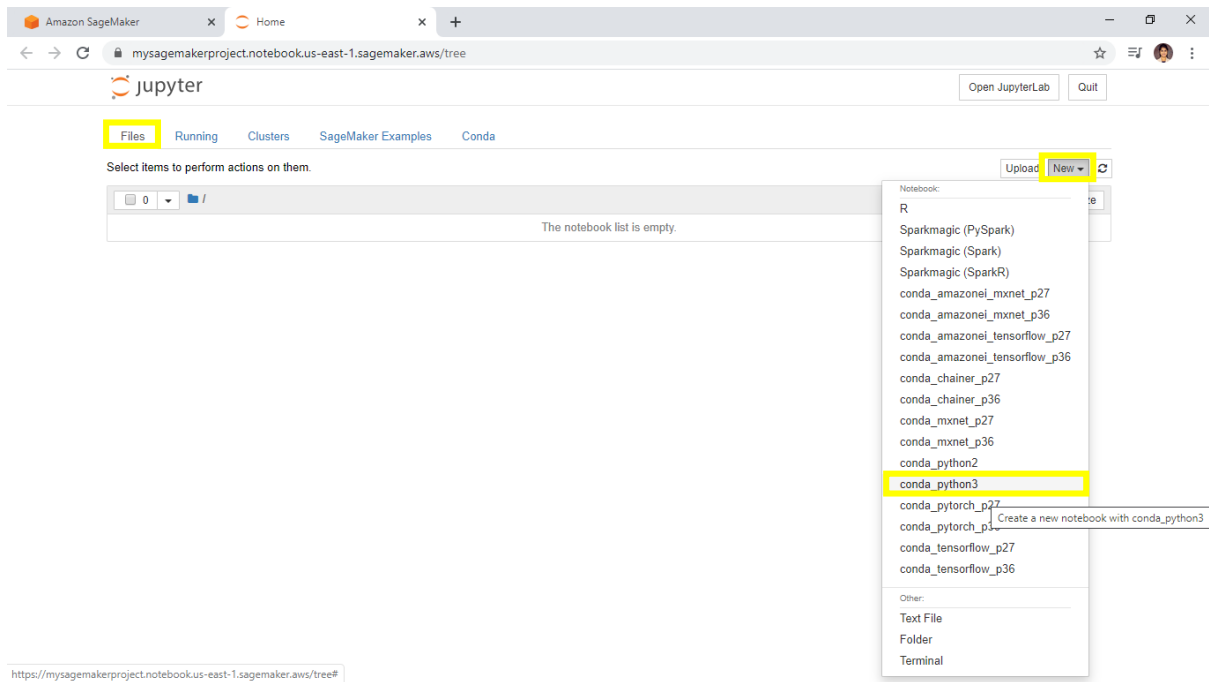
In this step you will use your Amazon SageMaker notebook to preprocess the data that you need to train your machine learning model.

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

After the status is **InService**, select *MySageMakerProject* and open it using the **Actions** drop down menu, or by choosing **Open Jupyter** next to the **InService** status [3].



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

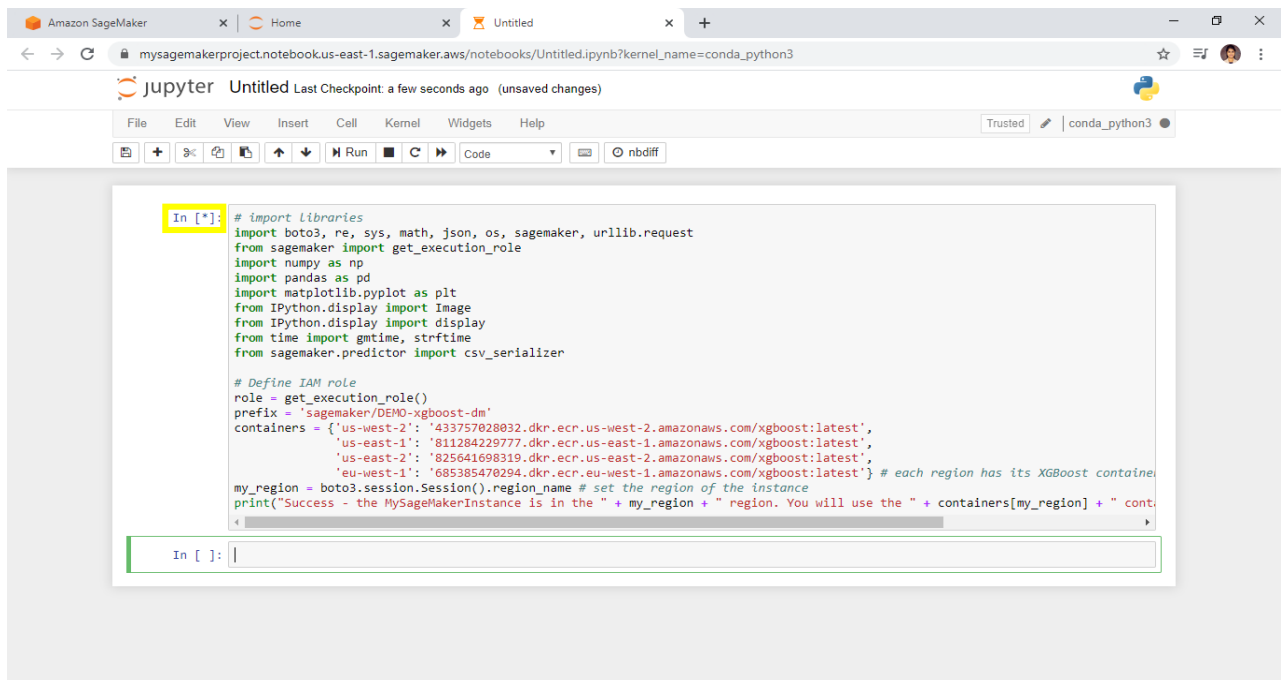


c. To prepare the data, train the ML model, and deploy it, you will need to import some libraries and define a few environment variables in your Jupyter notebook environment. Copy the following code into the code cell in your instance and select **Run** [3].

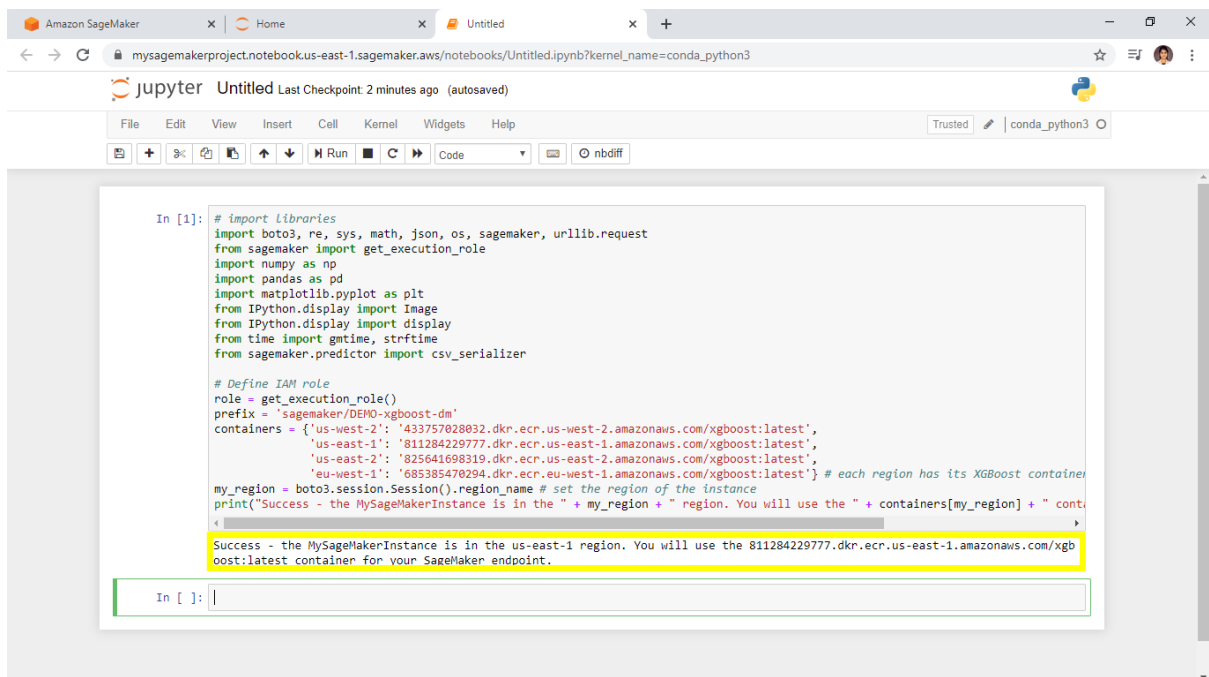
While the code runs, an * appears between the square brackets as pictured in the first screenshot to the right.

```
# import libraries
import boto3, re, sys, math, json, os, sagemaker, urllib.request
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 container
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] + " container for your SageMaker endpoint.")
```



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.



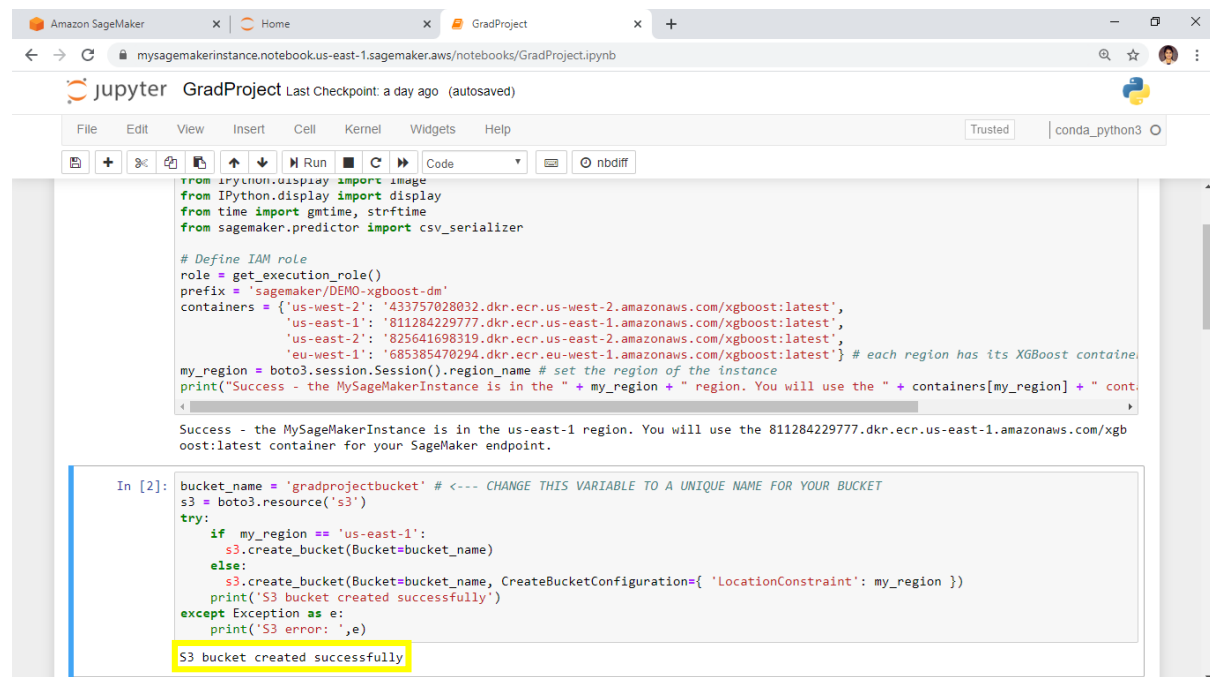
You can rename the notebook by clicking on 'Untitled' and replacing it with your desired name for the notebook. I have renamed the notebook as 'GradProject'.

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

Copy the following code into the next code cell in your notebook and change the name of the S3 bucket to make it unique. S3 bucket names must be globally unique and have [some other restrictions and limitations](#).

Select **Run**. If you don't receive a success message, change the bucket name and try again [3].

```
bucket_name = 'your-s3-bucket-name' # <--- 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)
```



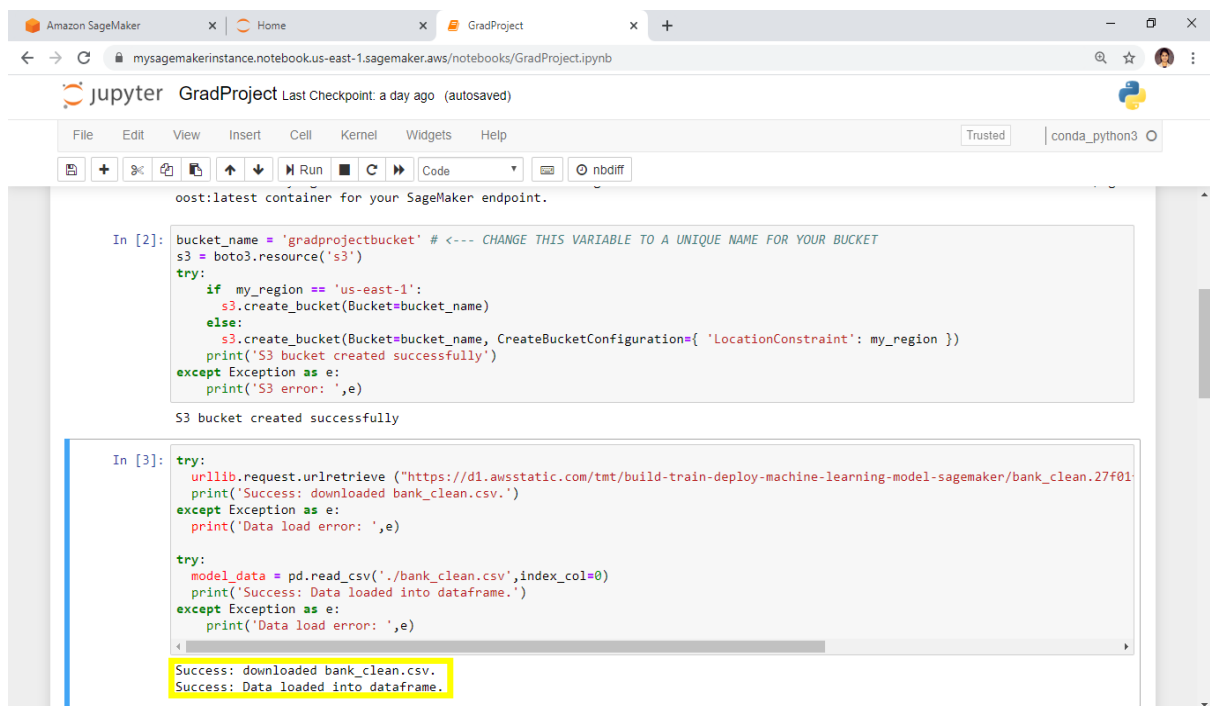
e. Next, you need to **download the data** to your Amazon SageMaker instance and **load it into a dataframe**. Copy and **Run** the following code [3]:

```
try:
    urllib.request.urlretrieve ("https://d1.awsstatic.com/tmt/build-train-deploy-machine-
learning-model-sagemaker/bank_clean.27f01fbbdf43271788427f3682996ae29ceca05d.csv",
"bank_clean.csv")

    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)
```



f. 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 [3].

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 [3]:

```
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)
```

OUTPUT:

```
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)
```

IV. TRAIN THE MODEL FROM THE DATA

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

a. 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 [3].

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

```
pd.concat([train_data['y_yes'], train_data.drop(['y_no', 'y_yes'], axis=1)],
axis=1).to_csv('train.csv', index=False, header=False)

boto3.Session().resource('s3').Bucket(bucket_name).Object(os.path.join(prefix,
'train/train.csv')).upload_file('train.csv')

s3_input_train = sagemaker.s3_input(s3_data='s3://{}/{}/train'.format(bucket_name,
prefix), content_type='csv')
```

b. Next, you need to set up the Amazon SageMaker session, create an instance of the XGBoost model (an estimator), and define the model's hyperparameters. Copy the following code into a new code cell and select **Run** [3]:

```
sess = sagemaker.Session()

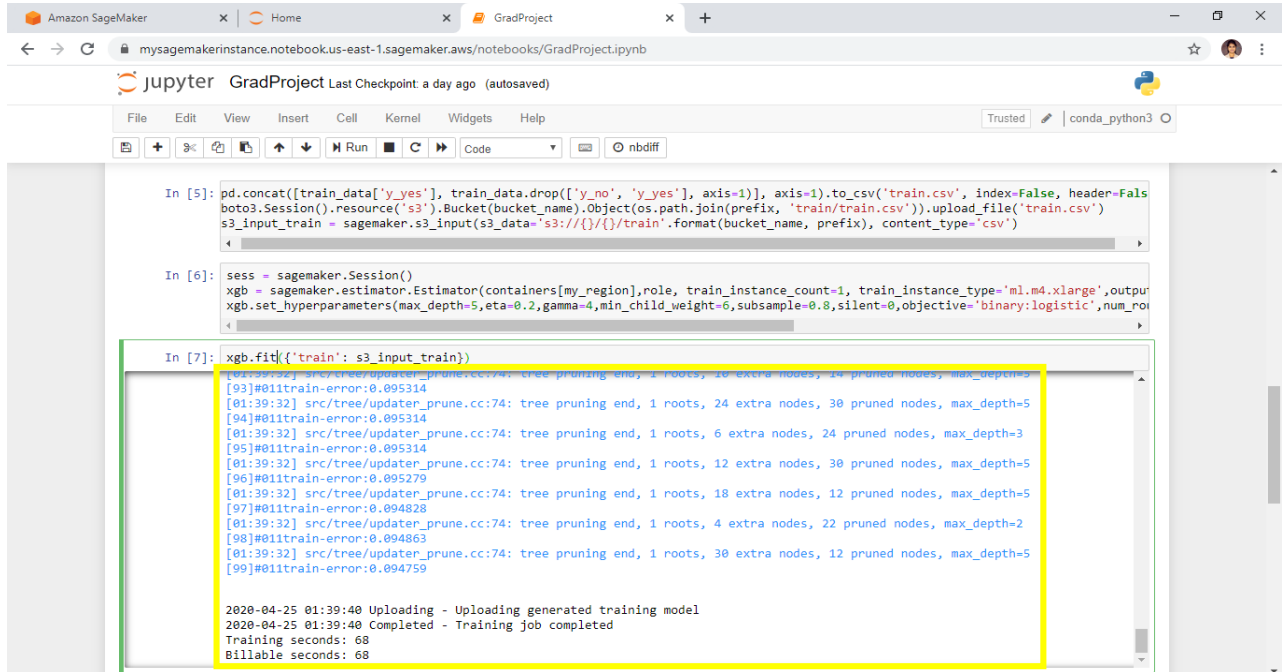
xgb = sagemaker.estimator.Estimator(containers[my_region],role, train_instance_count=1,
train_instance_type='ml.m4.xlarge',output_path='s3://{}/{}/output'.format(bucket_name,
prefix),sagemaker_session=sess)

xgb.set_hyperparameters(max_depth=5,eta=0.2,gamma=4,min_child_weight=6,subsample=0.8,
silent=0,objective='binary:logistic',num_round=100)
```

c. 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** [3].

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

After a few minutes, you should start to see the training logs being generated as follows:



```
In [5]: pd.concat([train_data['y_yes'], train_data.drop(['y_no', 'y_yes'], axis=1)], axis=1).to_csv('train.csv', index=False, header=False)
boto3.Session().resource('s3').Bucket(bucket_name).Object(os.path.join(prefix, 'train/train.csv')).upload_file('train.csv')
s3_input_train = sagemaker.s3_input(s3_data='s3://{}/{}/train'.format(bucket_name, prefix), content_type='csv')

In [6]: sess = sagemaker.Session()
xgb = sagemaker.estimator.Estimator(containers[my_region], role, train_instance_count=1, train_instance_type='ml.m4.xlarge', output
xgb.set_hyperparameters(max_depth=5, eta=0.2, gamma=4, min_child_weight=6, subsample=0.8, silent=0, objective='binary:logistic', num_rou

In [7]: xgb.fit({'train': s3_input_train})
[01:39:32] 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
[01:39:32] 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
[01:39:32] 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
[01:39:32] 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
[01:39:32] 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
[01:39:32] 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
[01:39:32] 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

2020-04-25 01:39:40 Uploading - Uploading generated training model
2020-04-25 01:39:40 Completed - Training job completed
Training seconds: 68
Billable seconds: 68
```

V. 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.

a. 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** [3]:

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

OUTPUT:

```
In [8]: xgb_predictor = xgb.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge')
```

```
-----!
```

b. 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** [3]:

```
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)
```

OUTPUT:

```
In [9]: 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,)
```

VI. EVALUATE THE PERFORMANCE

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

a. 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 [3].

```
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))

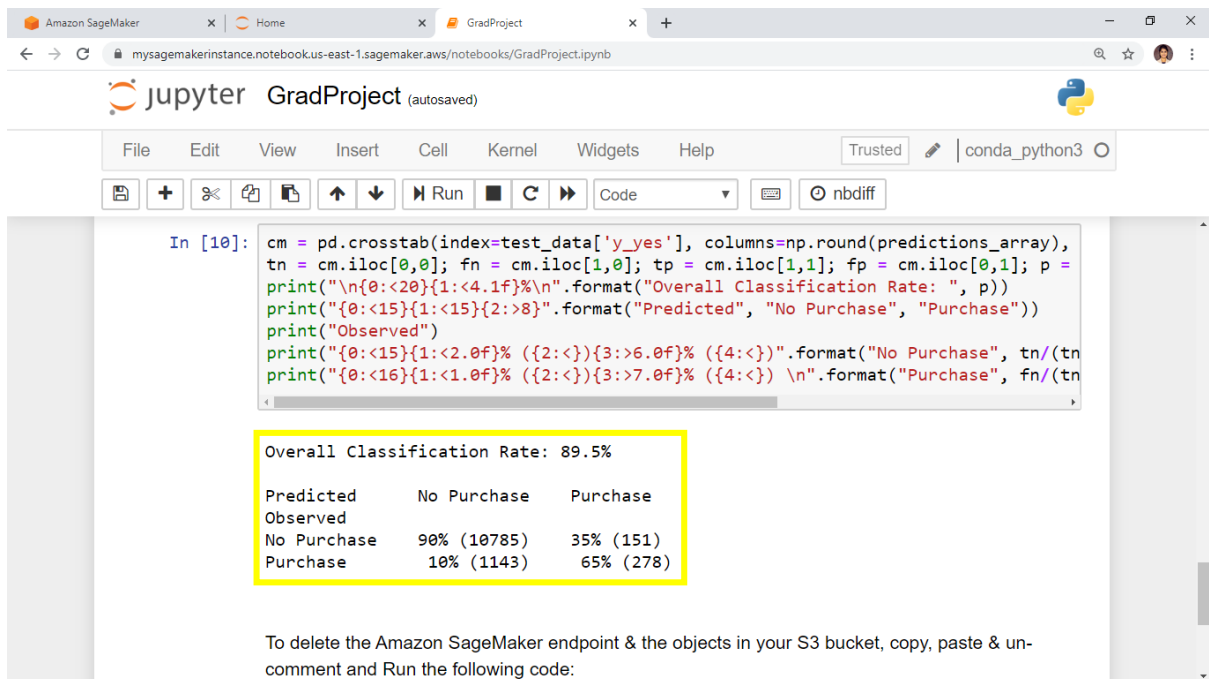
print("{0:<15}{1:<15}{2:>8}".format("Predicted", "No Purchase", "Purchase"))

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))
```

OUTPUT:



The screenshot shows a Jupyter Notebook interface with the title 'GradProject (autosaved)'. The code cell 'In [10]:' contains a pandas crosstab analysis of predictions versus actual values. The output displays the 'Overall Classification Rate: 89.5%' and a confusion matrix table.

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

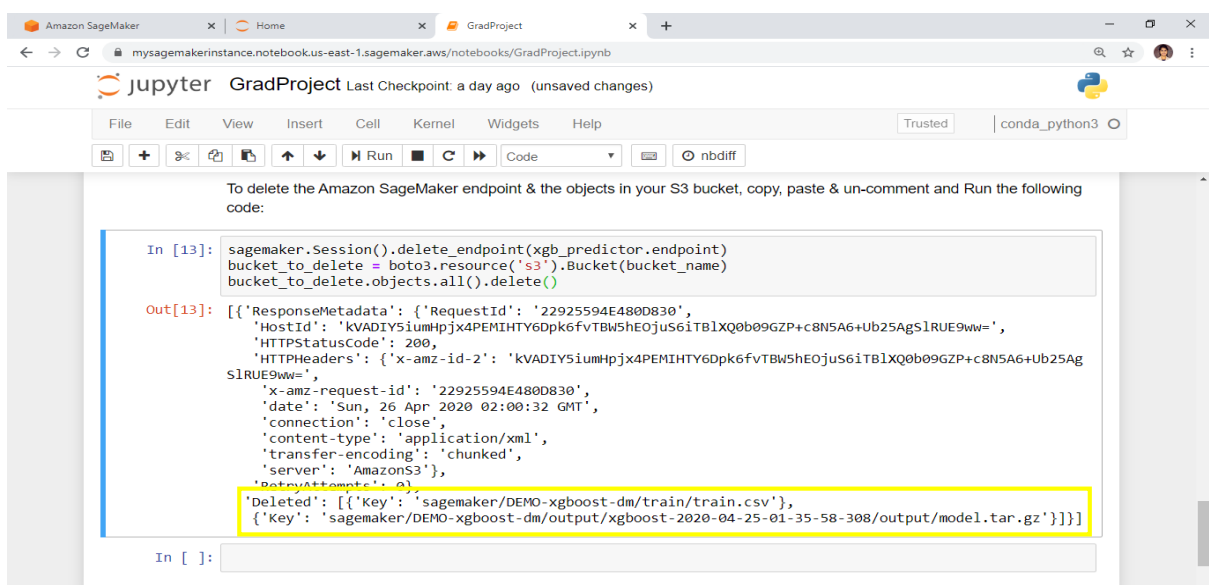
Below the table, a text instruction reads: 'To delete the Amazon SageMaker endpoint & the objects in your S3 bucket, copy, paste & un-comment and Run the following code:'

VII. TERMINATE YOUR RESOURCES

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

a. To delete the Amazon SageMaker endpoint and the objects in your S3 bucket, copy, paste and **Run** the following code [3]:

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



The screenshot shows the same Jupyter Notebook interface. The code cell 'In [13]:' contains the deletion code. The output 'Out[13]:' shows a detailed JSON response from the SageMaker API, including metadata, status, and a list of deleted objects. The 'Deleted' list is highlighted with a yellow box.

```
"Deleted": [{"key": "sagemaker/DEMO-xgboost-dm/train/train.csv"}, {"key": "sagemaker/DEMO-xgboost-dm/output/xgboost-2020-04-25-01-35-58-308/output/model.tar.gz"}]]
```


CONCLUSION:

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 [3].

OUTPUT

MySageMakerProject Notebook - [MySageMakerProject](#)

REFERENCES:

1. <https://thebestschools.org/magazine/future-meaningful-work/>
2. <https://aws.amazon.com/machine-learning/>
3. <https://aws.amazon.com/getting-started/hands-on/build-train-deploy-machine-learning-model-sagemaker/>