Importing Important Libraries

Steps To Be Followed

1. Importing necessary Libraries
2. Creating S3 bucket
3. Mapping train And Test Data in S3
4. Mapping The path of the models in S3

In [10]:
```
import sagemaker
import boto3
from sagemaker.amazon.amazon_estimator import get_image_uri
from sagemaker.session import s3_input, Session
```

In [11]:
```
bucket_name = 'bankapplication' # <--- CHANGE THIS VARIABLE TO A UNIQUE NAME FOR YOUR
my_region = boto3.session.Session().region_name # set the region of the instance
print(my_region)
```

```
us-east-1
```

In [12]:
```
s3 = boto3.resource('s3')
try:
    if  my_region == 'us-east-1':
        s3.create_bucket(Bucket=bucket_name)
    print('S3 bucket created successfully')
except Exception as e:
    print('S3 error: ',e)
```

```
S3 bucket created successfully
```

In [13]:
```
# set an output path where the trained model will be saved
prefix = 'xgboost-as-a-built-in-algo'
output_path ='s3://{}/{}/output'.format(bucket_name, prefix)
print(output_path)
```

```
s3://bankapplication/xgboost-as-a-built-in-algo/output
```

Downloading The Dataset And Storing in S3

In [14]:
```
import pandas as pd
import urllib
try:
    urllib.request.urlretrieve ("https://d1.awsstatic.com/tmt/build-train-deploy-mach
    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.
```

In [7]:

```
### Train Test split

import numpy as np
train_data, test_data = np.split(model_data.sample(frac=1, random_state=1729), [int(0
print(train_data.shape, test_data.shape)
```

```
(28831, 61) (12357, 61)
```

In [15]:

```
### Saving Train And Test Into Buckets
## We start with Train Data
import os
pd.concat([train_data['y_yes'], train_data.drop(['y_no', 'y_yes'],
                                                axis=1)],
                                                axis=1).to_csv('train.csv', index=Fal
boto3.Session().resource('s3').Bucket(bucket_name).Object(os.path.join(prefix, 'train
s3_input_train = sagemaker.s3_input(s3_data='s3://{}/{}/train'.format(bucket_name, pr
```

```
's3_input' class will be renamed to 'TrainingInput' in SageMaker Python SDK v2.
```

In [16]:

```
# Test Data Into Buckets
pd.concat([test_data['y_yes'], test_data.drop(['y_no', 'y_yes'], axis=1)], axis=1).to
boto3.Session().resource('s3').Bucket(bucket_name).Object(os.path.join(prefix, 'test/
s3_input_test = sagemaker.s3_input(s3_data='s3://{}/{}/test'.format(bucket_name, pref
```

```
's3_input' class will be renamed to 'TrainingInput' in SageMaker Python SDK v2.
```

Building Models Xgboot- Inbuilt Algorithm

In [18]:

```
# this line automatically looks for the XGBoost image URI and builds an XGBoost conta
# specify the repo_version depending on your preference.
container = get_image_uri(boto3.Session().region_name,
                          'xgboost',
                          repo_version='1.0-1')
```

```
'get_image_uri' method will be deprecated in favor of 'ImageURIProvider' class in SageM
aker Python SDK v2.
```

In [25]:

```
# initialize hyperparameters
hyperparameters = {
        "max_depth":"5",
        "eta":"0.2",
        "gamma":"4",
        "min_child_weight":"6",
        "subsample":"0.7",
        "objective":"binary:logistic",
        "num_round":50
        }
```

In [26]:
```
# construct a SageMaker estimator that calls the xgboost-container
estimator = sagemaker.estimator.Estimator(image_name=container,
                                          hyperparameters=hyperparameters,
                                          role=sagemaker.get_execution_role(),
                                          train_instance_count=1,
                                          train_instance_type='ml.m5.2xlarge',
                                          train_volume_size=5, # 5 GB
                                          output_path=output_path,
                                          train_use_spot_instances=True,
                                          train_max_run=300,
                                          train_max_wait=600)
```

```
Parameter image_name will be renamed to image_uri in SageMaker Python SDK v2.
```

In [27]:
```
estimator.fit({'train': s3_input_train,'validation': s3_input_test})
```

```
2020-08-29 09:49:29 Starting - Starting the training job...
2020-08-29 09:49:31 Starting - Launching requested ML instances.........
2020-08-29 09:51:04 Starting - Preparing the instances for training...
2020-08-29 09:51:44 Downloading - Downloading input data
2020-08-29 09:51:44 Training - Downloading the training image..INFO:sagemaker-container
s:Imported framework sagemaker_xgboost_container.training
INFO:sagemaker-containers:Failed to parse hyperparameter objective value binary:logisti
c to Json.
Returning the value itself
INFO:sagemaker-containers:No GPUs detected (normal if no gpus installed)
INFO:sagemaker_xgboost_container.training:Running XGBoost Sagemaker in algorithm mode
INFO:root:Determined delimiter of CSV input is ','
INFO:root:Determined delimiter of CSV input is ','
INFO:root:Determined delimiter of CSV input is ','
[09:52:07] 28831x59 matrix with 1701029 entries loaded from /opt/ml/input/data/train?fo
rmat=csv&label_column=0&delimiter=,
INFO:root:Determined delimiter of CSV input is ','
[09:52:07] 12357x59 matrix with 729063 entries loaded from /opt/ml/input/data/validatio
n?format=csv&label_column=0&delimiter=,
INFO:root:Single node training.
INFO:root:Train matrix has 28831 rows
INFO:root:Validation matrix has 12357 rows
[09:52:07] WARNING: /workspace/src/learner.cc:328:
Parameters: { num_round } might not be used.

  This may not be accurate due to some parameters are only used in language bindings bu
t
  passed down to XGBoost core.  Or some parameters are not used but slip through this
  verification. Please open an issue if you find above cases.


[0]#011train-error:0.10079#011validation-error:0.10528
[1]#011train-error:0.09968#011validation-error:0.10456
[2]#011train-error:0.10017#011validation-error:0.10375
[3]#011train-error:0.09989#011validation-error:0.10310
[4]#011train-error:0.09996#011validation-error:0.10286
[5]#011train-error:0.09906#011validation-error:0.10261
[6]#011train-error:0.09930#011validation-error:0.10286
[7]#011train-error:0.09951#011validation-error:0.10261
[8]#011train-error:0.09920#011validation-error:0.10286
[9]#011train-error:0.09871#011validation-error:0.10294
[10]#011train-error:0.09868#011validation-error:0.10294
[11]#011train-error:0.09868#011validation-error:0.10326
[12]#011train-error:0.09854#011validation-error:0.10358
[13]#011train-error:0.09892#011validation-error:0.10342
[14]#011train-error:0.09850#011validation-error:0.10342
[15]#011train-error:0.09844#011validation-error:0.10326
[16]#011train-error:0.09857#011validation-error:0.10318
[17]#011train-error:0.09799#011validation-error:0.10318
[18]#011train-error:0.09816#011validation-error:0.10383
[19]#011train-error:0.09857#011validation-error:0.10383
[20]#011train-error:0.09830#011validation-error:0.10350
[21]#011train-error:0.09826#011validation-error:0.10318
[22]#011train-error:0.09847#011validation-error:0.10399
[23]#011train-error:0.09833#011validation-error:0.10407
[24]#011train-error:0.09812#011validation-error:0.10415
[25]#011train-error:0.09812#011validation-error:0.10399
[26]#011train-error:0.09774#011validation-error:0.10375
[27]#011train-error:0.09781#011validation-error:0.10375
[28]#011train-error:0.09781#011validation-error:0.10391
```

```
[29]#011train-error:0.09778#011validation-error:0.10367
[30]#011train-error:0.09781#011validation-error:0.10383
[31]#011train-error:0.09771#011validation-error:0.10358
[32]#011train-error:0.09743#011validation-error:0.10391
[33]#011train-error:0.09753#011validation-error:0.10342
[34]#011train-error:0.09767#011validation-error:0.10342
[35]#011train-error:0.09757#011validation-error:0.10350
[36]#011train-error:0.09757#011validation-error:0.10342
[37]#011train-error:0.09736#011validation-error:0.10342
[38]#011train-error:0.09750#011validation-error:0.10342
[39]#011train-error:0.09733#011validation-error:0.10350
[40]#011train-error:0.09705#011validation-error:0.10358
[41]#011train-error:0.09701#011validation-error:0.10383
[42]#011train-error:0.09712#011validation-error:0.10407
[43]#011train-error:0.09698#011validation-error:0.10375
[44]#011train-error:0.09733#011validation-error:0.10342
[45]#011train-error:0.09736#011validation-error:0.10367
[46]#011train-error:0.09746#011validation-error:0.10350
[47]#011train-error:0.09736#011validation-error:0.10358
[48]#011train-error:0.09712#011validation-error:0.10334
[49]#011train-error:0.09712#011validation-error:0.10318

2020-08-29 09:52:19 Uploading - Uploading generated training model
2020-08-29 09:52:19 Completed - Training job completed
Training seconds: 44
Billable seconds: 21
Managed Spot Training savings: 52.3%
```

Deploy Machine Learning Model As Endpoints

In [28]:

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

```
Parameter image will be renamed to image_uri in SageMaker Python SDK v2.
---------------!
```

Prediction of the Test Data

In [29]:

```
from sagemaker.predictor import csv_serializer
test_data_array = test_data.drop(['y_no', 'y_yes'], axis=1).values #load the data int
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
print(predictions_array.shape)
```

```
(12357,)
```

In [30]:

```
predictions_array
```

Out[30]:

```
array([0.05214286, 0.05660191, 0.05096195, ..., 0.03436061, 0.02942475,
       0.03715819])
```

In [31]:

```
cm = pd.crosstab(index=test_data['y_yes'], columns=np.round(predictions_array), rowna
tn = cm.iloc[0,0]; fn = cm.iloc[1,0]; tp = cm.iloc[1,1]; fp = cm.iloc[0,1]; p = (tp+t
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)*
print("{0:<16}{1:<1.0f}% ({2:<}){3:>7.0f}% ({4:<}) \n".format("Purchase", fn/(tn+fn)*
```

```
Overall Classification Rate: 89.7%

Predicted      No Purchase    Purchase
Observed
No Purchase    91% (10785)    34% (151)
Purchase        9% (1124)     66% (297)
```

Deleting The Endpoints

In [32]:

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

Out[32]:

```
[{'ResponseMetadata': {'RequestId': '2FF829102DC6DFD1',
   'HostId': 'mYPqeWyx3REoLIsQu2MVorzKLrlxES2n6Dcdr3PycVf1VkRCxicEewoPP8IxRguc5MGksLn
jynY=',
   'HTTPStatusCode': 200,
   'HTTPHeaders': {'x-amz-id-2': 'mYPqeWyx3REoLIsQu2MVorzKLrlxES2n6Dcdr3PycVf1VkRCxic
EewoPP8IxRguc5MGksLnjynY=',
    'x-amz-request-id': '2FF829102DC6DFD1',
    'date': 'Sat, 29 Aug 2020 10:21:27 GMT',
    'connection': 'close',
    'content-type': 'application/xml',
    'transfer-encoding': 'chunked',
    'server': 'AmazonS3'},
   'RetryAttempts': 0},
  'Deleted': [{'Key': 'xgboost-as-a-built-in-algo/train/train.csv'},
   {'Key': 'xgboost-as-a-built-in-algo/test/test.csv'},
   {'Key': 'xgboost-as-a-built-in-algo/output/sagemaker-xgboost-2020-08-29-09-49-29-0
15/output/model.tar.gz'}]}]
```

In [ ]: