Importing important libraries

1) Import libraries
2) create s3 bucket
3) Mapping train and Test data in S3
4) Mapping the path of the models in S3

In [25]:
```
import sagemaker
import boto3 #for accessing s3 bucket
from sagemaker.amazon.amazon_estimator import get_image_uri
from sagemaker.session import s3_input, Session
```

In [30]:
```
bucketname = "Assignment1"
my_region = boto3.session.Session().region_name # set the region of the instance
print(my_region)
```

```
ap-south-1
```

In [32]:
```
s3 = boto3.resource('s3')
try:
    if my_region=="ap-south-1":
        s3.create_bucket(Bucket = bucketname)
    print("s3 bucket created successfully")
except Exception as e:
    print("S3 error: ", e)
```

```
s3 bucket created successfully
```

In [76]:
```
#Set an oputput path for saving the trained model
prefix = "xgboost-as-a-built-in-algo"
output_path = 's3://{}/{}/output'.format(bucketname,prefix)

output_path2 = 's3://{}/{}/output'.format("testbucketforassignone",prefix)

print(output_path)
```

```
s3://Assignment1/xgboost-as-a-built-in-algo/output
```

In [39]:
```
import pandas as pd
import urllib
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)
```

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

In [40]:
```
model_data.head()
```

Out[40]:

	age	campaign	pdays	previous	no_previous_contact	not_working	job_admin.	job_blue-collar	job_entrepreneur	job_housemaid	job_m
0	56	1	999	0	1	0	0	0	0	1	
1	57	1	999	0	1	0	0	0	0	0	
2	37	1	999	0	1	0	0	0	0	0	
3	40	1	999	0	1	0	1	0	0	0	
4	56	1	999	0	1	0	0	0	0	0	

In [41]:
```
model_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 41188 entries, 0 to 41187
Data columns (total 61 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   age                             41188 non-null  int64
 1   campaign                        41188 non-null  int64
 2   pdays                           41188 non-null  int64
 3   previous                        41188 non-null  int64
 4   no_previous_contact             41188 non-null  int64
 5   not_working                     41188 non-null  int64
 6   job_admin.                      41188 non-null  int64
 7   job_blue-collar                 41188 non-null  int64
 8   job_entrepreneur                41188 non-null  int64
 9   job_housemaid                   41188 non-null  int64
 10  job_management                  41188 non-null  int64
 11  job_retired                     41188 non-null  int64
 12  job_self-employed               41188 non-null  int64
 13  job_services                    41188 non-null  int64
 14  job_student                     41188 non-null  int64
 15  job_technician                  41188 non-null  int64
 16  job_unemployed                  41188 non-null  int64
 17  job_unknown                     41188 non-null  int64
 18  marital_divorced                41188 non-null  int64
 19  marital_married                 41188 non-null  int64
 20  marital_single                  41188 non-null  int64
 21  marital_unknown                 41188 non-null  int64
 22  education_basic.4y              41188 non-null  int64
 23  education_basic.6y              41188 non-null  int64
 24  education_basic.9y              41188 non-null  int64
 25  education_high.school           41188 non-null  int64
 26  education_illiterate            41188 non-null  int64
 27  education_professional.course   41188 non-null  int64
 28  education_university.degree     41188 non-null  int64
 29  education_unknown               41188 non-null  int64
 30  default_no                      41188 non-null  int64
 31  default_unknown                 41188 non-null  int64
 32  default_yes                     41188 non-null  int64
 33  housing_no                      41188 non-null  int64
 34  housing_unknown                 41188 non-null  int64
 35  housing_yes                     41188 non-null  int64
 36  loan_no                         41188 non-null  int64
 37  loan_unknown                    41188 non-null  int64
 38  loan_yes                        41188 non-null  int64
 39  contact_cellular                41188 non-null  int64
 40  contact_telephone               41188 non-null  int64
 41  month_apr                       41188 non-null  int64
 42  month_aug                       41188 non-null  int64
 43  month_dec                       41188 non-null  int64
 44  month_jul                       41188 non-null  int64
 45  month_jun                       41188 non-null  int64
 46  month_mar                       41188 non-null  int64
 47  month_may                       41188 non-null  int64
 48  month_nov                       41188 non-null  int64
 49  month_oct                       41188 non-null  int64
 50  month_sep                       41188 non-null  int64
 51  day_of_week_fri                 41188 non-null  int64
 52  day_of_week_mon                 41188 non-null  int64
 53  day_of_week_thu                 41188 non-null  int64
 54  day_of_week_tue                 41188 non-null  int64
 55  day_of_week_wed                 41188 non-null  int64
 56  poutcome_failure                41188 non-null  int64
 57  poutcome_nonexistent            41188 non-null  int64
 58  poutcome_success                41188 non-null  int64
 59  y_no                            41188 non-null  int64
 60  y_yes                           41188 non-null  int64
dtypes: int64(61)
memory usage: 19.5 MB
```

In [43]:
```
#train_test_split
import numpy as np
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)
```

In [56]:
```
# saving the train and test data into the bucket
import os
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("testbucketforassignone").Object(os.path.join(prefix, 'train/train.csv'
)).upload_file('train.csv')
s3_input_train = sagemaker.TrainingInput(s3_data='s3://{}/{}/train'.format("testbucketforassignone", prefix),
content_type='csv')
```

In [58]:
```
#Test Data Into Buckets
pd.concat([test_data['y_yes'], test_data.drop(['y_no', 'y_yes'], axis=1)], axis=1).to_csv('test.csv', index=F
alse, header=False)
boto3.Session().resource('s3').Bucket("testbucketforassignone").Object(os.path.join(prefix, 'test/test.csv'))
.upload_file('test.csv')
s3_input_test = sagemaker.TrainingInput(s3_data='s3://{}/{}/test'.format("testbucketforassignone", prefix), c
ontent_type='csv')
```

Building a Model (XGBOOST-inbuilt)

In [70]:
```
from sagemaker import image_uris
container = sagemaker.image_uris.retrieve("xgboost", boto3.Session().region_name, "1.2-1")

#container = get_image_uri(boto3.Session().region_name, 'xgboost', repo_version = '1.0-1')
```

In [71]:
```
# 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 [77]:
```
# construct a SageMaker estimator that calls the xgboost-container
estimator = sagemaker.estimator.Estimator(image_uri=container,
                                          hyperparameters=hyperparameters,
                                          role=sagemaker.get_execution_role(),
                                          instance_count=1,
                                          instance_type='ml.m5.2xlarge',
                                          volume_size=5, # 5 GB
                                          output_path=output_path2,
                                          use_spot_instances=True,
                                          max_run=300,
                                          max_wait=600)
```

Training data

In [78]:
```
estimator.fit ({'train': s3_input_train,'validation': s3_input_test})
```

```
2021-11-01 22:52:39 Starting - Starting the training job...
2021-11-01 22:53:03 Starting - Launching requested ML instancesProfilerReport-1635807159: InProgress
......
2021-11-01 22:54:03 Starting - Preparing the instances for training......
2021-11-01 22:55:04 Downloading - Downloading input data...
2021-11-01 22:55:24 Training - Downloading the training image..[2021-11-01 22:55:51.173 ip-10-0-71-154.ap-sou
th-1.compute.internal:1 INFO utils.py:27] RULE_JOB_STOP_SIGNAL_FILENAME: None
INFO:sagemaker-containers:Imported framework sagemaker_xgboost_container.training
INFO:sagemaker-containers:Failed to parse hyperparameter objective value binary:logistic 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 ','
INFO:root:Determined delimiter of CSV input is ','
INFO:root:Single node training.
INFO:root:Train matrix has 28831 rows and 59 columns
INFO:root:Validation matrix has 12357 rows
[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

2021-11-01 22:56:04 Uploading - Uploading generated training model
2021-11-01 22:56:04 Completed - Training job completed
Training seconds: 61
Billable seconds: 19
Managed Spot Training savings: 68.9%
```

Deploy machine learning model as Endpoints

In [79]:
```
xgb_predictor = estimator.deploy(initial_instance_count=1,instance_type='ml.m4.xlarge')
```

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

Prediction using Test data

In [82]:
```
from sagemaker.predictor import csv_serializer
test_data_array = test_data.drop(['y_no', 'y_yes'], axis=1).values #load the data into an array
#xgb_predictor.content_type = '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)
```

```
The csv_serializer has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.

(12357,)
```

In [83]:
```
predictions_array
```

Out[83]:
```
array([0.05214286, 0.05660191, 0.05096195, ..., 0.03436061, 0.02942475,
       0.03715819])
```

In [85]:
```
cm = pd.crosstab(index=test_data['y_yes'], columns=np.round(predictions_array), rownames=['Observed'], colnam
es=['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: 89.7%

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

In [ ]:
```
#deleting the end points
sagemaker.Session().delete_endpoint(xgb_predictor.endpoint)
bucket_to_delete = boto3.resource('s3').Bucket("testbucketforassignone")
bucket_to_delete.objects.all().delete()
```