

# PROJECT REPORT

TELECOM CUSTOMER CHURN PREDICTION POWERED BY AWS  
SAGEMAKER

NAME:Prabhanjan Kumar

PROJECT: Telecom Customer Churn Prediction

Powered By Aws Sagemaker

PROJECT DOMAIN: Machine Learning

# **CONTENTS:**

## **1.INTRODUCTION**

### **1.1 Abstract**

### **1.2 Overview**

## **2.LITERATURE SURVEY**

### **2.1 Existing problem**

### **2.2 Proposed solution**

## **3.THEORETICAL ANALYSIS**

### **3.1 Block diagram**

### **3.2 Hardware/Software designing**

## **4.EXPERIMENTAL INVESTIGATIONS**

## **5.RESULT**

## **6.ADVANTAGES & DISADVANTAGES**

## **7.APPLICATIONS**

## **8. FUTURE SCOPE**

## **9.CONCLUSION**

## **10.BIBLIOGRAPHY**

## **1.INTRODUCTION**

### **1.1 ABSTRACT:**

At recent years, estimating the churners before they leave has gained importance in environment of increased competition in company strategy. churners are tried to detect by using Machine Learning techniques. Attribute reductions are tried for decreasing the runtime and increasing achievement of models and performance was measured by using different classification method. In addition, outlier analysis is applied to dataset and then effects on classification results are examined. This classification methods are tested in two datasets which are taken from Telecommunication Companies. Recall and Precision Rates are used as performance criteria.

### **1.2 OVERVIEW**

The telecommunications sector has become one of the main industries in developed countries. The technical progress and the increasing number of operators raised the level of competition. Companies are working hard to survive in this competitive market depending on multiple strategies. Customers' churn is a considerable concern in service sectors with high competitive services. On the other hand, predicting the customers who are likely to leave the company will represent potentially large additional revenue source if it is done in the early phase .Many research confirmed that machine learning technology is highly efficient to predict this situation. This technique is applied through learning from previous data

## **2.LITERATURE SURVEY:**

### **2.1EXISTING PROBLEM:**

Customer churn is a major problem and one of the most important concerns for large companies. Telecommunication industry always suffers from very high

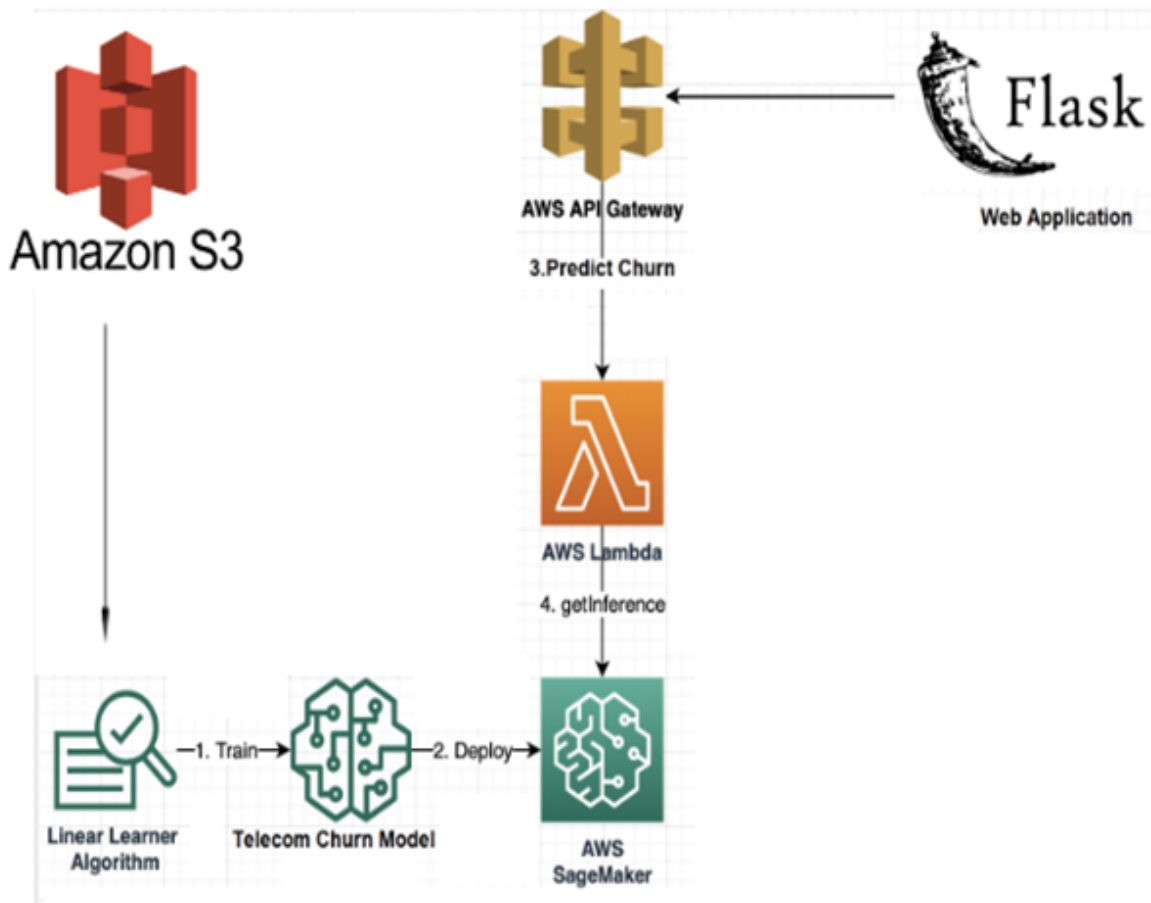
churn rates when one industry offers a better plan than the previous there is a high possibility of the customer churning from the present due to a better plan in such a scenario it is very difficult to avoid losses but through prediction, we can keep it to a minimal level. Due to the direct effect on the revenues of the companies, companies are seeking to develop means to predict potential customers to churn. Therefore, finding factors that increase customer churn is important to take necessary actions to reduce it.

## **2.2 PROPOSED SOLUTION:**

Churn prediction helps in identifying those customers who are likely to leave a company. The main contribution of our work is to develop a churn prediction model which assists telecom operators to predict customers who are most likely subject to churn. Build & Deploy a Machine Learning model to predict the customer churn using Amazon SageMaker and predictions can be obtained by using its Endpoint. Create a python - flask application that interacts with the model deployed on AWS Sagemaker with the help of AWS API Gateway and AWS Lambda Services.

## **3. THEORETICAL ANALYSIS:**

### **3.1. BLOCK DIAGRAM:**



### 3.2. SOFTWARE DESIGNING:

1. Amazon S3
2. AWS API Gateway
3. AWS Lambda
4. Amazon SageMaker
5. Python3
6. Flask integration

### 4.EXPERIMENTAL INVESTIGATIONS:

#### Aws Cloud:

Aws Cloud Provides Many Services Such as Sagemaker,lambda and Api Gateway,etc..

**Sagemaker:**

Amazon SageMaker is a fully managed service that provides every developer and data scientist with the ability to build, train, and deploy machine learning (ML) models quickly. SageMaker removes the heavy lifting from each step of the machine learning process to make it easier to develop high quality models.

**Lambda:**

With Lambda, you can run code for virtually any type of application or backend service - all

with zero administration. Just upload your code and Lambda takes care of everything required

to run and scale your code with high availability. You can set up your code to automatically

trigger from other AWS services

**Api Gateway:**

Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale. API developers can create APIs that access AWS or other web services, as well as data stored in the AWS Cloud. API Gateway creates RESTful APIs that are HTTP-based.

**5.RESULT:**

Student Dashboard x Pathways x Sample Form x +

127.0.0.1:5000

Apps Neural machine tra... MT Tab-delimited Bilin... Time series forecast... Language Translati... Intuitive Understan... Word Level English... Image captioning w... Text generation wit...

## Telecom Customer Churn Prediction

Enter Credit Score

Male ☐ Female ☐

Enter Age

Enter Tenure

France ☐ Spain ☐ German ☐

Enter Balance

Enter No Of Products

Enter Estimated Salary

Cr Card : Yes ☐ No ☐

Is Active Member Yes ☐ No ☐

Activate Windows  
Go to Settings to activate Windows.

Type here to search

13:49 07-10-2020

Student Dashboard x Pathways x Sample Form x +

127.0.0.1:5000

Apps Neural machine tra... MT Tab-delimited Bilin... Time series forecast... Language Translati... Intuitive Understan... Word Level English... Image captioning w... Text generation wit...

## Telecom Customer Churn Prediction

610

Male ☒ Female ☐

30

8

France ☐ Spain ☐ German ☒

50000

3

100003

Cr Card : Yes ☐ No ☒

Is Active Member Yes ☒ No ☐

Activate Windows  
Go to Settings to activate Windows.

Type here to search

13:51 07-10-2020

Student Dashboard x Pathways x Sample Form x +

127.0.0.1:5000

Apps Neural machine tra... MT Tab-delimited Bilin... Time series forecast... Language Translati... Intuitive Understan... Word Level English... Image captioning w... Text generation wit...

## Telecom Customer Churn Prediction

Result: Employee Is Likely To Leave

Enter Credit Score

Male ☐ Female ☐

Enter Age

Enter Tenure

France ☐ Spain ☐ German ☐

Enter Balance

Enter No Of Products

Enter Estimated Salary

Activate Windows  
Go to Settings to activate Windows.

Type here to search

13:50 07-10-2020

## 6.ADVANTAGES:

1. Easy to understand and efficient training algorithm(xgclassifier algorithm).
2. Always find a “good solution”

## 7.APPLICATIONS:

1. It is used in costumer churn prediction in telecom industries
2. It is used in costumer churn prediction in banking industries
3. It is used in CRM(Customer Relationship Management)

## 8.FUTURE SCOPE:

There are straightforward decision rules based models and complex classification models for churn forecast undertaking has been proposed in the writing. While these strategies are productive in playing out the churn forecast undertaking, they



require manual component designing procedure which tedious and blunder inclined. At the point when the outcome are not acquired at the right time we can't take the fundamental activities to abstain from churning so we need even more a logical answer for abstain from churning.

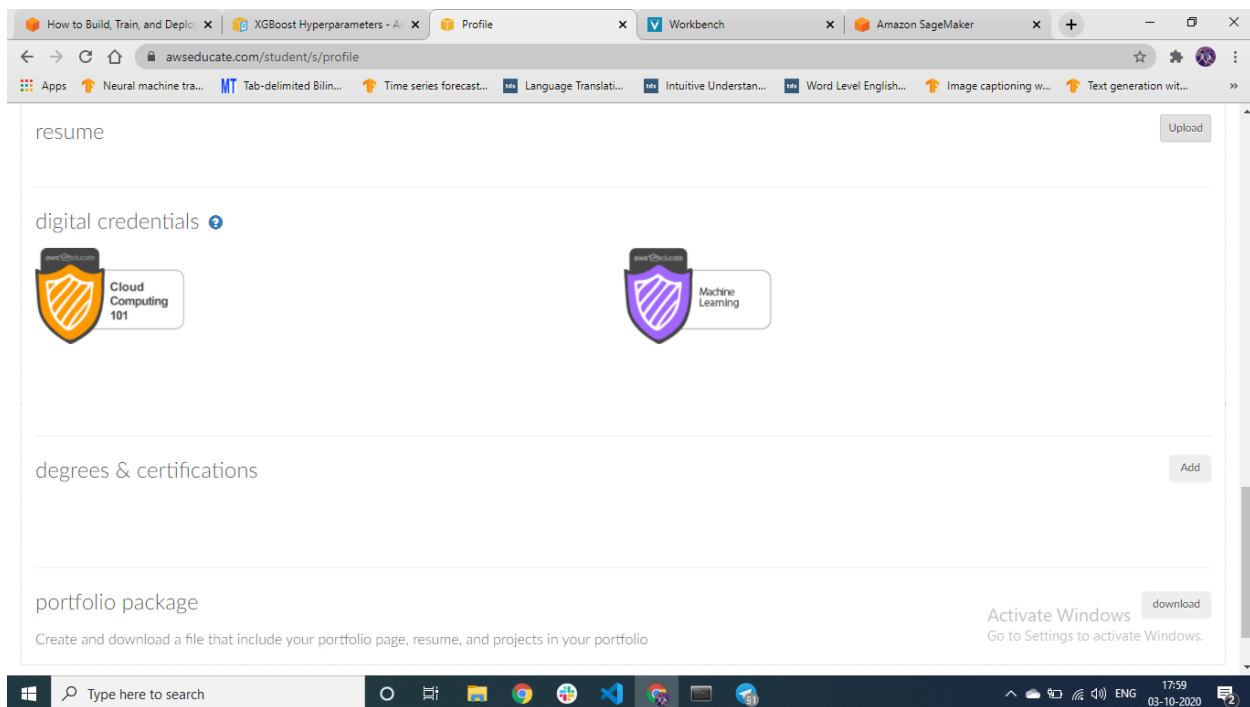
## **9.CONCLUSION:**

The importance of this type of research in the telecom market is to help companies make more profit. It has become known that predicting churn is one of the most important sources of income to telecom companies. We have applied feature engineering, effective feature transformation and selection approach to make the features ready for machine learning algorithms. The use of the Social Network Analysis features enhance the results of predicting the churn in telecom.

## **10.BIBILOGRAPHY:**

Qureshii SA, Rehman AS, Qamar AM, Kamal A, Rehman A. Telecommunication subscribers' churn prediction model using machine learning. In: Eighth international conference on digital information management.

## **Carrer Pathways:**



```
1 import numpy as np
2 import pandas as pd
3 import seaborn as sns
4 import matplotlib.pyplot as plt
```

```
1 dataset=pd.read_csv('telecom.csv')
2 dataset.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.8
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.5
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.5
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.6
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.1

```
1 dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
#   Column                Non-Null Count  Dtype
---  -
0   RowNumber              10000 non-null  int64
1   CustomerId             10000 non-null  int64
2   Surname                10000 non-null  object
3   CreditScore             10000 non-null  int64
4   Geography              10000 non-null  object
5   Gender                 10000 non-null  object
6   Age                    10000 non-null  int64
7   Tenure                  10000 non-null  int64
8   Balance                 10000 non-null  float64
9   NumOfProducts          10000 non-null  int64
10  HasCrCard               10000 non-null  int64
11  IsActiveMember          10000 non-null  int64
12  EstimatedSalary         10000 non-null  float64
13  Exited                  10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

```
1 dataset.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000

```
1 dataset.shape
```

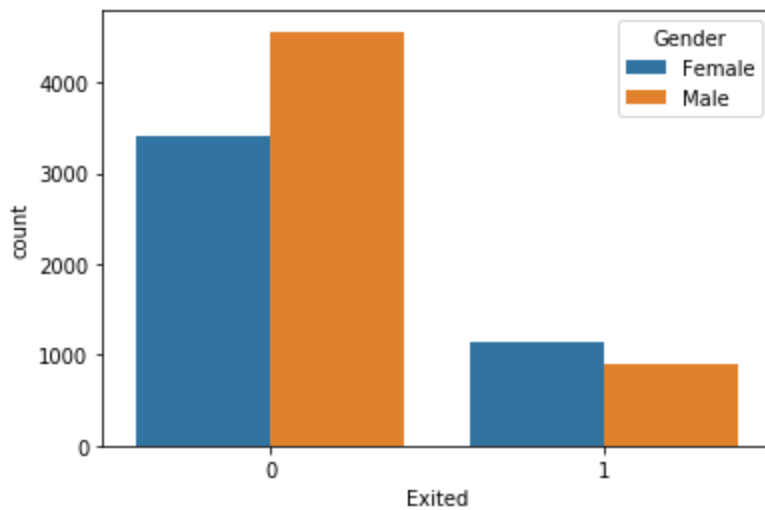
```
(10000, 14)
```

```
1 dataset.isnull().sum()
```

```
RowNumber      0
CustomerId      0
Surname         0
CreditScore     0
Geography       0
Gender          0
Age            0
Tenure         0
Balance        0
NumOfProducts  0
HasCrCard      0
IsActiveMember  0
EstimatedSalary 0
Exited         0
dtype: int64
```

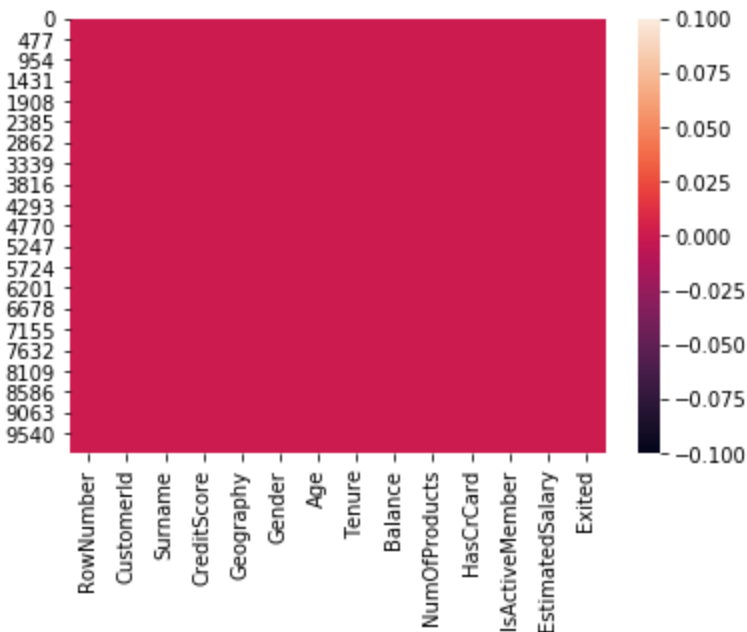
```
1 sns.countplot(x='Exited',hue='Gender',data=dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f09f27584e0>
```



```
1 sns.heatmap(dataset.isnull())
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f09f1ebe5c0>



```
1 dataset=dataset.drop(['RowNumber', 'CustomerId',  
    'Surname'],axis=1)  
2 dataset.head()
```

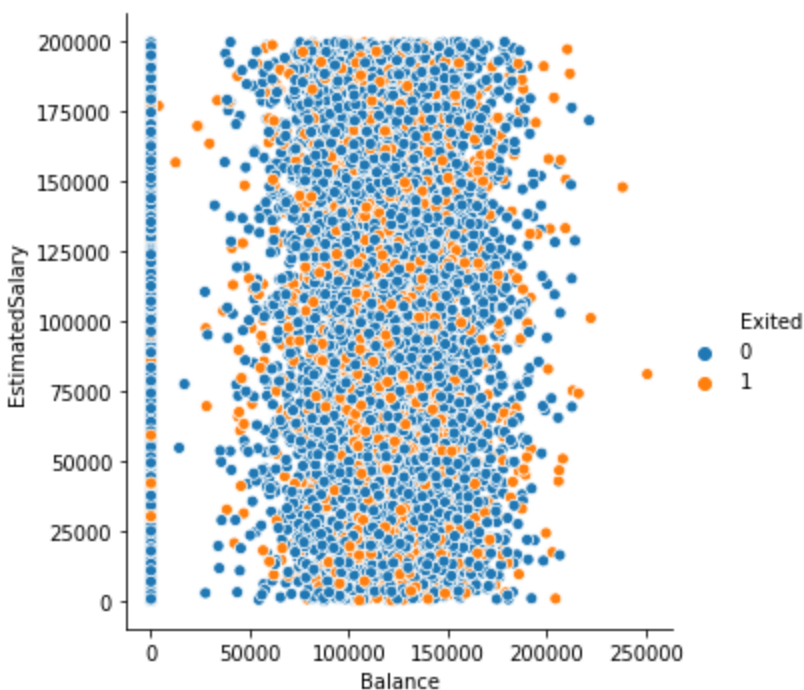
	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

```
1 from sklearn.preprocessing import LabelEncoder  
2 from sklearn.model_selection import train_test_split  
3 from sklearn.preprocessing import MinMaxScaler  
4 lb1=LabelEncoder()  
5 lb2=LabelEncoder()  
6 dataset['Gender']=lb1.fit_transform(dataset['Gender'])  
7 dataset['Geography']=lb2.fit_transform(dataset['Geography'])  
8 dataset.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	850	2	0	43	2	125510.82	1	1	1	79084.10	0

```
1 sns.relplot(x='Balance',y='EstimatedSalary',hue='Exited',data=dataset)
```

<seaborn.axisgrid.FacetGrid at 0x7f09f0bc90f0>



```
1 data_in=dataset.iloc[:, :-1]
2 data_out=dataset.iloc[:, -1]
3 sc=MinMaxScaler(feature_range=(0,1))
4 data_in=sc.fit_transform(data_in)
5 keys=dataset.keys()[ :-1]
6 dici={}
7 for i in range(len(keys)):
8     dici.update({keys[i]:data_in[:,i]})
9 final_data=pd.DataFrame(dici)
10 final_data.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	0.538	0.0	0.0	0.324324	0.2	0.000000	0.000000	1.0	1.0	0.506735
1	0.516	1.0	0.0	0.310811	0.1	0.334031	0.000000	0.0	1.0	0.562709
2	0.304	0.0	0.0	0.324324	0.8	0.636357	0.666667	1.0	0.0	0.569654
3	0.698	0.0	0.0	0.283784	0.1	0.000000	0.333333	0.0	0.0	0.469120
4	1.000	1.0	0.0	0.337838	0.2	0.500246	0.000000	1.0	1.0	0.395400

```
1 final_data=pd.concat([dataset.iloc[:,-1],final_data],axis=1)
2 final_data.head()
```

	Exited	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	0.538	0.0	0.0	0.324324	0.2	0.000000	0.000000	1.0	1.0	0.506735
1	0	0.516	1.0	0.0	0.310811	0.1	0.334031	0.000000	0.0	1.0	0.562709
2	1	0.304	0.0	0.0	0.324324	0.8	0.636357	0.666667	1.0	0.0	0.569654
3	0	0.698	0.0	0.0	0.283784	0.1	0.000000	0.333333	0.0	0.0	0.469120
4	0	1.000	1.0	0.0	0.337838	0.2	0.500246	0.000000	1.0	1.0	0.395400

```
1 import boto3,re,os,json,sagemaker
2 from sagemaker import get_execution_role
3 train,test=train_test_split(final_data,test_size=0.2)
4 role=get_execution_role()
5 my_region=boto3.session.Session().region_name
6 containers = {'us-west-2':
7               '433757028032.dkr.ecr.us-west-2.amazonaws.com/xgboost:lates
8               t',
9               'us-east-1':
10              '811284229777.dkr.ecr.us-east-1.amazonaws.com/xgboost:lates
11              t',
12              'us-east-2':
13              '825641698319.dkr.ecr.us-east-2.amazonaws.com/xgboost:lates
14              t',
15              'eu-west-1':
16              '685385470294.dkr.ecr.eu-west-1.amazonaws.com/xgboost:lates
17              t'}
18 prefix='sagemaker/Telecom'
19 bucket_name='buildathonproject1'
20 final_data.to_csv('train.csv',index=False,header=False)
21 boto3.Session().resource('s3').Bucket(bucket_name).Object(o
```

```

    s.path.join(prefix,'train/train.csv')).upload_file('train.csv')
14 s3_input_train=sagemaker.s3_input(s3_data='s3://{}/{}/train'
    '.format(bucket_name, prefix),content_type='csv')
15 sess=sagemaker.Session()
16 telecom_model=sagemaker.estimator.Estimator(containers[my_r
    egion],role,train_instance_count=1,train_instance_type='ml.
    m5.large',output_path='s3://{}/{}/output'.format(bucket_nam
    e,prefix),sagemaker_session=sess)
17 telecom_model.set_hyperparameters(max_depth=5,eta=0.2,gamma
    =4,min_child_weight=6,subsample=0.8,silent=0,objective='bin
    ary:logistic',num_round=100)
18 telecom_model.fit({'train':s3_input_train})

```

```

2020-09-30 07:50:56 Starting - Starting the training job...
2020-09-30 07:50:59 Starting - Launching requested ML instances.....
2020-09-30 07:52:21 Starting - Preparing the instances for training.....
2020-09-30 07:53:12 Downloading - Downloading input data...
2020-09-30 07:53:48 Training - Downloading the training image..Arguments: train
[2020-09-30:07:54:03:INFO] Running standalone xgboost training.
[2020-09-30:07:54:03:INFO] Path /opt/ml/input/data/validation does not exist!
[2020-09-30:07:54:03:INFO] File size need to be processed in the node: 0.99mb. Available memory size in the node: 172.35mb
[2020-09-30:07:54:03:INFO] Determined delimiter of CSV input is ','
[07:54:03] S3DistributionType set as FullyReplicated
[07:54:03] 10000x10 matrix with 100000 entries loaded from /opt/ml/input/data/train?format=csv&label_column=0&delimiter=,
[07:54:03] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 38 extra nodes, 6 pruned nodes, max_depth=5
[0]#011train-error:0.1448
[07:54:03] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 32 extra nodes, 12 pruned nodes, max_depth=5
[1]#011train-error:0.144
[07:54:03] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 38 extra nodes, 4 pruned nodes, max_depth=5
[2]#011train-error:0.144
[07:54:03] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 38 extra nodes, 6 pruned nodes, max_depth=5
[3]#011train-error:0.144
[07:54:03] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 38 extra nodes, 6 pruned nodes, max_depth=5

```

```

1 detector=telecom_model.deploy(initial_instance_count=1,inst
    ance_type='ml.m5.large')

```

Parameter image will be renamed to image\_uri in SageMaker Python SDK v2.

-----!

```

1 detector.endpoint

```

```
'xgboost-2020-09-30-07-50-56-779'
```

```

1 from sagemaker.predictor import csv_serializer
2 test_data_array=test.drop('Exited',axis=1).values #load the

```



```

data into an array
3 detector.content_type = 'text/csv' # set the data type for
  an inference
4 detector.serializer = csv_serializer # set the serializer
  type
5 predictions=detector.predict(test_data_array).decode('utf-8
  ') # predict!
6 predictions_array = np.fromstring(predictions[1:], sep=',')
7 print(predictions)

```

```

0.357669651508,0.141457110643,0.329746335745,0.026482027024,0.0712991580367,0.0263096913695,0.0375938378274,0.0215682908893,
0.0678913518786,0.115104779601,0.414443165064,0.015394564718,0.0659468099475,0.983116984367,0.564640462399,0.767752766609,0.2
11881354451,0.0556389167905,0.266310662031,0.0520670227706,0.0255261678249,0.0819280520082,0.121003270149,0.0237068850547,0.0
298531763256,0.400649666786,0.0279733166099,0.645623505116,0.144340574741,0.773866117001,0.0724500715733,0.0210663992912,0.66
0542488098,0.312410950661,0.805181801319,0.835865318775,0.0263171419501,0.255656123161,0.150213211775,0.17395016551,0.3088148
23627,0.0194727368653,0.290218800306,0.0748643428087,0.0298329666257,0.0253355037421,0.0449019707739,0.259558230639,0.0533759
184182,0.781935751438,0.0273960605264,0.157540291548,0.0343550741673,0.335181176662,0.0296571981162,0.091909609735,0.01418924
51793,0.157416403294,0.0391214042902,0.450152218342,0.292664647102,0.0937503576279,0.0576719790697,0.0443506836891,0.03734270
48326,0.155199185014,0.0502981394529,0.0710270255804,0.446701139212,0.572361409664,0.180645704269,0.7317096591,0.083087496459
5,0.0461247563362,0.0191855877638,0.0121706062928,0.0155159085989,0.364274173975,0.382801711559,0.412225365639,0.006050311494
62,0.11486851424,0.15652140975,0.0366217754781,0.130395650864,0.0441899485886,0.635700643063,0.0170651581138,0.0360826738179,
0.0455167926848,0.012677596882,0.0177081599832,0.0138631332666,0.66416233778,0.176358506083,0.531168580055,0.0239507853985,0.
0872323140502,0.837302923203,0.0105954483151,0.0460762195289,0.0309317186475,0.163216099143,0.601965308189,0.0419460423291,0.
0393777601421,0.0375836156309,0.017990315333,0.346629232168,0.064279101789,0.101827032864,0.265378087759,0.975147724152,0.070
0022429228,0.049429371953,0.0824607014656,0.144094198942,0.0924224555492,0.303251862526,0.215495213866,0.235083475709,0.02136
01496071,0.0220698155463,0.0728238001466,0.00614333618432,0.488911926746,0.10665551573,0.0739371702075,0.0133571783081,0.0279
69064191,0.697656035423,0.265094041824,0.0515954867005,0.082623578608,0.00672792643309,0.0578222945333,0.362552344799,0.03714
24034238,0.15151129663,0.172684624791,0.0699690058827,0.509014189243,0.287413448095,0.0959527418017,0.026997230947,0.52917981
1478,0.0673849955201,0.021628184244,0.0224863402545,0.546940028667,0.0326875001192,0.0189216658473,0.0947914049029,0.04943434

```

## LambdaFunction:

```

1 import io
2 import boto3
3 import json
4 import csv
5 def lambda_handler(event, context):
6     ENDPOINT_NAME = os.environ['environment_variable']
7     runtime= boto3.client('runtime.sagemaker')
8     print(ENDPOINT_NAME)
9     print("Received event: " , json.dumps(event, indent=2))
10    data = json.loads(json.dumps(event))
11    print("Data:",data)
12    payload = data['data']
13    print("Payload:",payload)
14    response =

```

```

runtime.invoke_endpoint(EndpointName=ENDPOINT_NAME,
15
    ContentType='text/csv',
16
                                Body=payload)
17     print(response)
18     result = json.loads(response['Body'].read().decode())
19     print(result)
20
21     if result>0.5:
22         return "P"
23     else:
24         return "N"

```

```

Response:
"p"

Request ID:
"566ea769-7ad5-4604-8ed9-1bbb57eae67b"

Function logs:
START RequestId: 566ea769-7ad5-4604-8ed9-1bbb57eae67b Version: $LATEST
xgboost-2020-09-30-07-50-56-779
Received event: {
  "data": "61,0,0,42.2,0.00,1,1,1,1348.88"
}
Data: {'data': '61,0,0,42.2,0.00,1,1,1,1348.88'}
Payload: 61,0,0,42.2,0.00,1,1,1,1348.88
{'ResponseMetadata': {'RequestId': 'a6c2f8a7-66e3-43f8-9b17-82f71d81e0c7', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': 'a6c2f8
0.760711729527

```

## UI:

### Hello.py

```

1 from flask import Flask,render_template,request,url_for
2 import requests
3 app=Flask(__name__)
4 @app.route('/',methods=['POST','GET'])
5 def hello():
6     if request.method=='POST':
7         cs=request.form['a']
8         geo=request.form['r2']
9         gen=request.form['r1']
10        age=request.form['b']
11        tenure=request.form['c']
12        balance=request.form['d']

```

```

13         np=request.form['e']
14         hc=request.form['r3']
15         im=request.form['r4']
16         es=request.form['f']
17         try:
18             cs=int(cs)
19             geo=int(geo)
20             gen=int(gen)
21             age=int(age)
22             tenure=int(tenure)
23             balance=float(balance)
24             np=int(np)
25             hc=int(hc)
26             im=int(im)
27             es=float(es)
28         except:
29             return
30         render_template('data.html',err_msg='Enter Valid Data')
31         url =
32         "https://qxk1s4orj6.execute-api.us-east-1.amazonaws.com/tel
33         ecom/"
34         payload = " {\\"data\\":\\" + str(cs) + ',' +
35         str(geo) + ',' + str(gen) + ',' + str(age) + ',' +
36         str(tenure) + ',' + str(balance) + ',' + str(np)
37         +','+str(hc)+ ','+str(im)+',' + str(es) + "\\" + \"}"
38
39         headers = {
40             'X-Amz-Content-Sha256':
41             'beaead3198f7da1e70d03ab969765e0821b24fc913697e929e726aeaeb
42             f0eba3',
43             'X-Amz-Date': '20200930T095337Z',
44             'Authorization': 'AWS4-HMAC-SHA256
45             Credential=ASIA4KDESJFDUSSQKJSG/20200930/us-east-1/execute-
46             api/aws4_request,
47             SignedHeaders=host;x-amz-content-sha256;x-amz-date,
48             Signature=b81935cc533d5efb8db465da9c12f4a3ed76ca80089dfc3ed

```

```

        ebdb39df4fe5f7c',
37         'Content-Type': 'text/plain'
38     }
39
40     response = requests.request("POST", url,
        headers=headers, data=payload)
41     response=response.text.encode('utf8')
42     response=str(response)
43     print(response)
44     result=response[3]
45     print(result)
46     if result=='N':
47         return render_template('data.html',result='Not
Likely to Leave')
48     else:
49         return
        render_template('data.html',result='Likely To Leave')
50     else:
51         return render_template('data.html')
52
53 if __name__ == '__main__':
54     app.run(debug=True)
55

```

### Data.html:

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <title>Sample Form</title>
5     <style>
6         .tel
7         {
8             width:300px;
9             height:25px;
10            background-color:#fffbfc;

```

```

11         border-style: ridge;
12         border-color:gray;
13         border-radius:6px;
14     }
15     body
16     {
17         font-family:sans-serif;
18     }
19     #sub
20     {
21         width:200px;
22         height:25px;
23         background-color:#9f6cff;
24         border-style: ridge;
25         border-color:gray;
26         border-radius:6px;
27     }
28 </style>
29 </head>
30 <body style="background-color:#615c4e">
31     <center>
32         <h1 style="color:red">Telecom Customer Churn
33         Prediction</h1>
34         <div style="background-color:#a8a883;">
35             <br/>
36             {% if result %}
37             <p style="color:red;font-size:30px;"> Result:
38             Employee Is {{result}}</p>
39             {% endif %}
40             {% if err_msg %}
41             <p
42             style="color:red;font-size:15px;">{{err_msg}}</p>
43             {% endif %}
44             <br/>
45             <form method="post" action="/">
46                 <input type="text" name="a" class="tel"

```

[illegible]

```
64 </body>
```

```
65 </html>
```