FLIP ROBO

Micro Credit Loan

Submitted by:

Amruta Shah

ACKNOWLEDGMENT

A unique opportunity like this comes very rarely. It is indeed a pleasure for me to have worked on this project.

The satisfaction that accompanies the successful completion of this project is incomplete without the mention of the people and references whose guidance has made it possible for me to complete this project.

I am grateful to my internship company **Flip Robo Technologies** with its ideals and inspiration for providing me with facilities that has made this project a success.

Also, I am grateful to my institute “Data Trained” for providing the opportunity to work as an intern in “Flip Robo Technologies” and giving me the great knowledge to complete this project.

I like to acknowledge the effort put in by our SME Khushboo Garg helping me understand the relevant concepts related to Data pre-processing and providing guidance when necessary.

Also based on below references I am able to encourage my knowledge time to time to improve my knowledge.

References:

- XGBClassifier
- Hyper-parameter Tuning
- SMOTE
- Getting started with XGBoost
- scikit

INTRODUCTION

- Business Problem Framing

  There are many poor families living in remote areas with not much sources of income, Since the communication via Telephone is important and how it affects a person's life, this project provides their services to low income families and poor customers that can help them in the need of hour by providing Micro credit loan to telephone number.

- Conceptual Background of the Domain Problem

  This project can very useful for the poor customers and also for the people who needs balance in urgent. Since we all have already used the mobile financial services which was provided by Airtel, this can be very useful for the people.

- Review of Literature

  Many microfinance institutions, experts and donors are supporting the idea of using mobile financial services which they feel are more convenient and efficient, and cost saving, than the traditional hightouch model used since long for the purpose of delivering microfinance services. Though, the MFI industry is primarily focusing on low-income families and are very useful in such areas, the implementation of MFS has been uneven with both significant challenges and successes. Today, microfinance is widely accepted as a poverty-reduction tool, representing $70 billion in outstanding loans and a global outreach of 200 million clients.

- Motivation for the Problem Undertaken

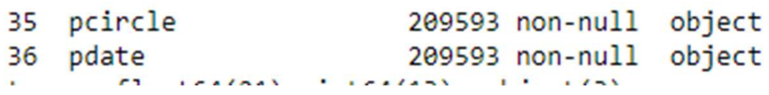
  Motivation behind this project is that mainly focuses on poor families living in remote areas with not much sources of income, Since the communication via Telephone is important and how it affects a person's life, this project provides the services to low income families and poor customers that can help them in the need of hour by providing Micro credit loan to telephone number.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem
  Mathematical Summary:
  Dimensions of Dataset: There are 36 columns and 209593 rows in this dataset.
  Null Values: There are no null values in this dataset.
  Skewness: Skewness is present in almost every column Statistical.
  Summary: Standard deviation is very high in most of the columns.
  Minimum negative values are present in the dataset.
  There is lot of difference between mean and 50th percentile, which means data is skewed There is lot of difference between 75th percentile and max, which means there are outliers.

- Data Sources and their formats
  Data Sources: The sample data is provided to us from our client database. In order to improve the selection of customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers. In this Project, Label ‘1’ indicates that the loan has been paid within 5 days i.e. Non- defaulter, while, Label ‘0’ indicates that the loan has not been paid within 5 days i.e. defaulter.

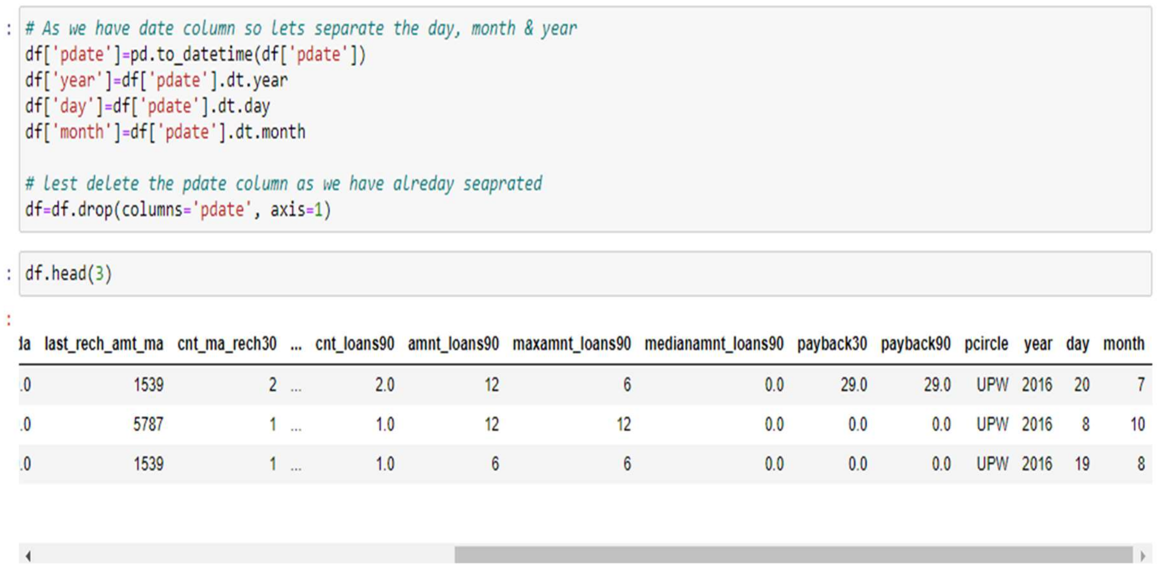
  Data Formats: Pcircal and date columns are object datatype which we need to encode & pre-process.



```
35  pcircle              209593 non-null  object
36  pdate                209593 non-null  object
```

- Data Pre-processing Done

First of all I have dropped irrelevant columns 'unknown','msisdn'. Then check for the duplicates some duplicates are found which is delated. After dropping the duplicated the shape of dataset is (rows=209562, columns=35).
Moving further as we have date columns so I have separated them into year, day & month basis as shown below.

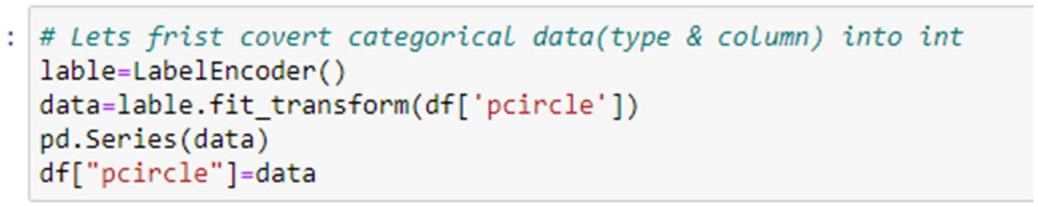


```
# As we have date column so lets separate the day, month & year
df['pdate']=pd.to_datetime(df['pdate'])
df['year']=df['pdate'].dt.year
df['day']=df['pdate'].dt.day
df['month']=df['pdate'].dt.month

# lest delete the pdate column as we have alreday seaprated
df=df.drop(columns='pdate', axis=1)
```

```
df.head(3)
```

```
la  last_rech_amt_ma  cnt_ma_rech30  ...  cnt_loans90  amnt_loans90  maxamnt_loans90  medianamnt_loans90  payback30  payback90  pcircle  year  day  month
.0              1539              2  ...          2.0            12                6                 0.0       29.0       29.0      UPW  2016   20      7
.0              5787              1  ...          1.0            12               12                 0.0        0.0        0.0      UPW  2016    8     10
.0              1539              1  ...          1.0             6                6                 0.0        0.0        0.0      UPW  2016   19      8
```

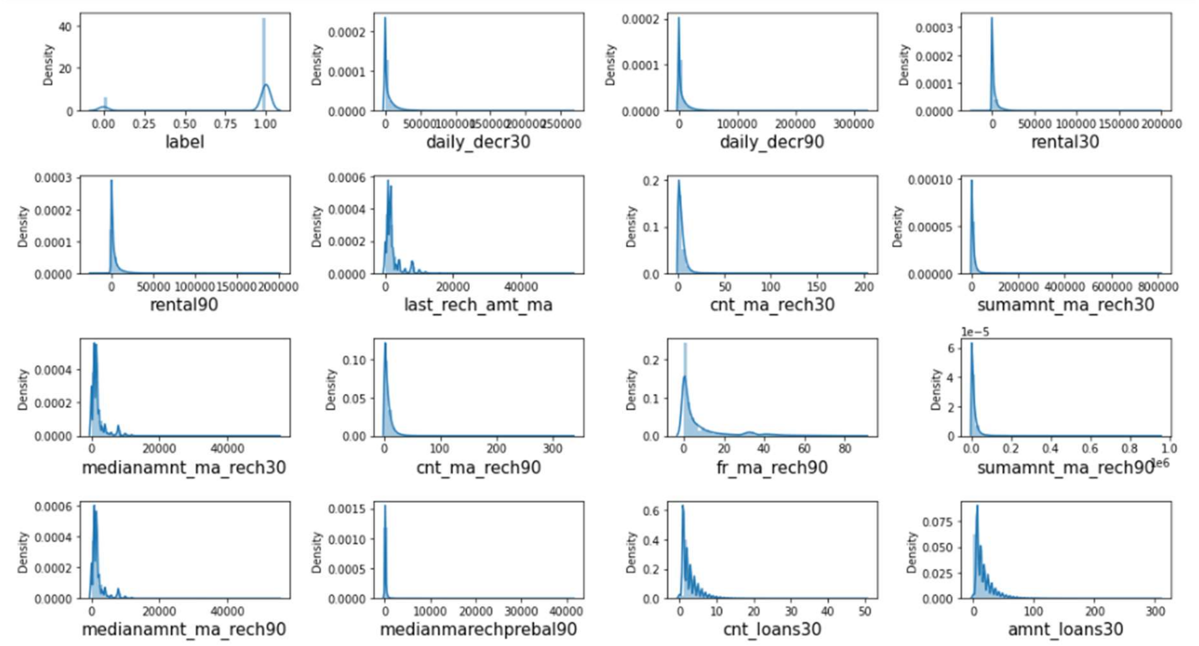
As our dataset has objective type data”pcircle” so covert them into int by using Encoding method as shown.



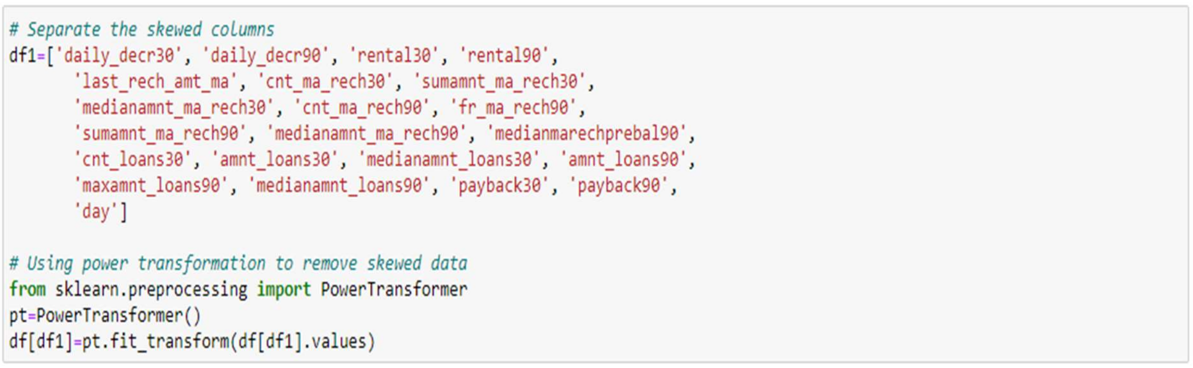
```
# Lets frist covert categorical data(type & column) into int
lable=LabelEncoder()
data=lable.fit_transform(df['pcircle'])
pd.Series(data)
df["pcircle"]=data
```

Columns dropped based on Correlation:
Removed these 14 columns which are correlation less than 0.01 with the target variable.
['month','cnt_loans90','cnt_da_rech30','last_rech_date_ma','cnt_da_rech90','last_rech_date_da','fr_ma_rech30','maxamnt_loans30','fr_da_rech30','aon','medianmarechprebal30','fr_da_rech90','pcircle','year']

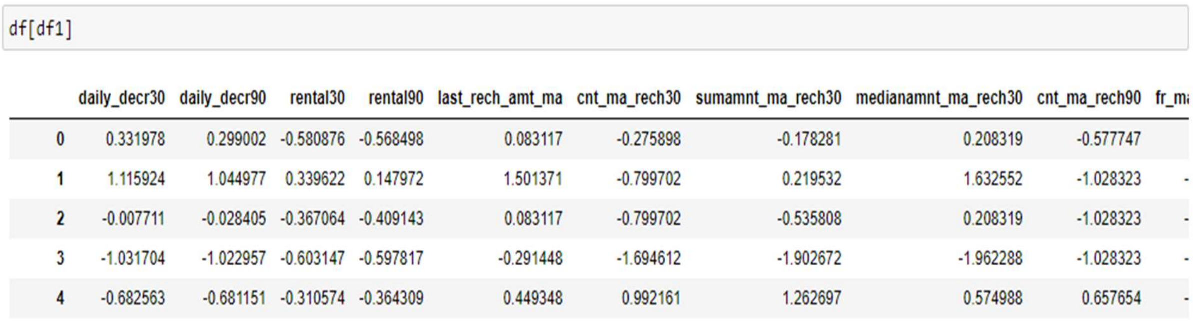
Removing skewness based on the distribution graph as below by using Power Transformation.



As we see there are almost in every independent variable has skewed data. As we see the skewed data is present as per thumb rule +0.5/-0.5 weight let’s work on right skewed data by using power transform. As label is our target variable so we are not using the same.



```
# Separate the skewed columns
df1=['daily_decr30', 'daily_decr90', 'rental30', 'rental90',
     'last_rech_amt_ma', 'cnt_ma_rech30', 'sumamnt_ma_rech30',
     'medianamnt_ma_rech30', 'cnt_ma_rech90', 'fr_ma_rech90',
     'sumamnt_ma_rech90', 'medianamnt_ma_rech90', 'medianmarechprebal90',
     'cnt_loans30', 'amnt_loans30', 'medianamnt_loans30', 'amnt_loans90',
     'maxamnt_loans90', 'medianamnt_loans90', 'payback30', 'payback90',
     'day']

# Using power transformation to remove skewed data
from sklearn.preprocessing import PowerTransformer
pt=PowerTransformer()
df[df1]=pt.fit_transform(df[df1].values)
```

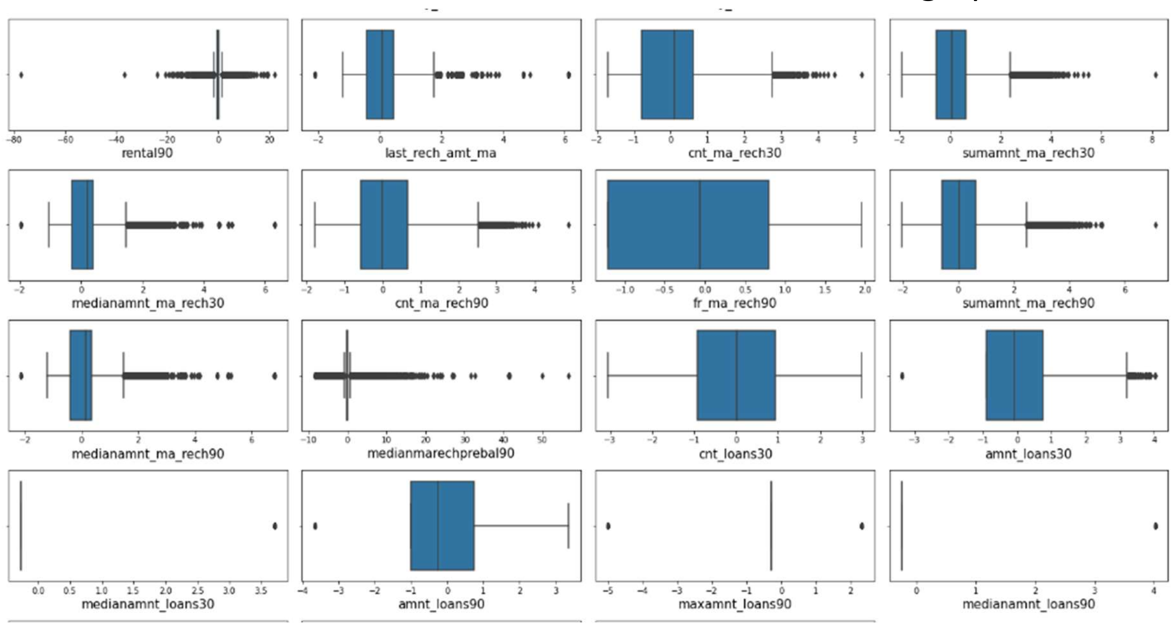


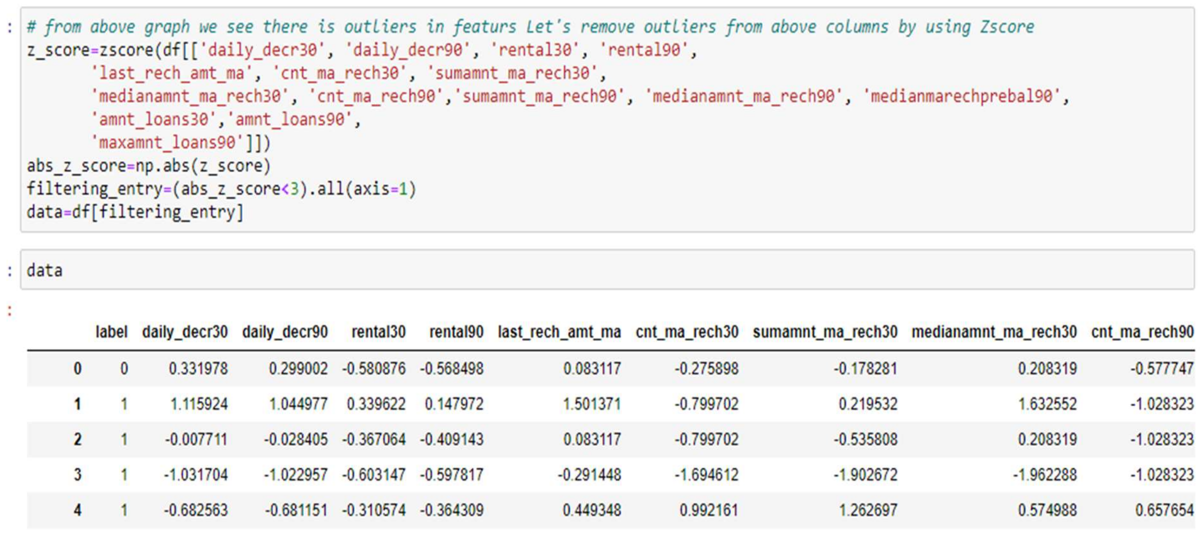
```
df[df1]
```

	daily_decr30	daily_decr90	rental30	rental90	last_rech_amt_ma	cnt_ma_rech30	sumamnt_ma_rech30	medianamnt_ma_rech30	cnt_ma_rech90	fr_ma
0	0.331978	0.299002	-0.580876	-0.568498	0.083117	-0.275898	-0.178281	0.208319	-0.577747	
1	1.115924	1.044977	0.339622	0.147972	1.501371	-0.799702	0.219532	1.632552	-1.028323	-
2	-0.007711	-0.028405	-0.367064	-0.409143	0.083117	-0.799702	-0.535808	0.208319	-1.028323	-
3	-1.031704	-1.022957	-0.603147	-0.597817	-0.291448	-1.694612	-1.902672	-1.962288	-1.028323	-
4	-0.682563	-0.681151	-0.310574	-0.364309	0.449348	0.992161	1.262697	0.574988	0.657654	-

Removing the Outliers:

As our dataset has maximum outliers as shown in below graph.





```
# from above graph we see there is outliers in featurs Let's remove outliers from above columns by using Zscore
z_score=zscore(df[['daily_decr30', 'daily_decr90', 'rental30', 'rental90',
        'last_rech_amt_ma', 'cnt_ma_rech30', 'sumamnt_ma_rech30',
        'medianamnt_ma_rech30', 'cnt_ma_rech90','sumamnt_ma_rech90', 'medianamnt_ma_rech90', 'medianmarechprebal90',
        'amnt_loans30','amnt_loans90',
        'maxamnt_loans90']])
abs_z_score=np.abs(z_score)
filtering_entry=(abs_z_score<3).all(axis=1)
data=df[filtering_entry]
```

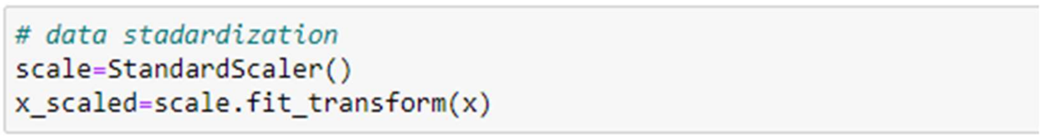
```
data
```

	label	daily_decr30	daily_decr90	rental30	rental90	last_rech_amt_ma	cnt_ma_rech30	sumamnt_ma_rech30	medianamnt_ma_rech30	cnt_ma_rech90
0	0	0.331978	0.299002	-0.580876	-0.568498	0.083117	-0.275898	-0.178281	0.208319	-0.577747
1	1	1.115924	1.044977	0.339622	0.147972	1.501371	-0.799702	0.219532	1.632552	-1.028323
2	1	-0.007711	-0.028405	-0.367064	-0.409143	0.083117	-0.799702	-0.535808	0.208319	-1.028323
3	1	-1.031704	-1.022957	-0.603147	-0.597817	-0.291448	-1.694612	-1.902672	-1.962288	-1.028323
4	1	-0.682563	-0.681151	-0.310574	-0.364309	0.449348	0.992161	1.262697	0.574988	0.657654

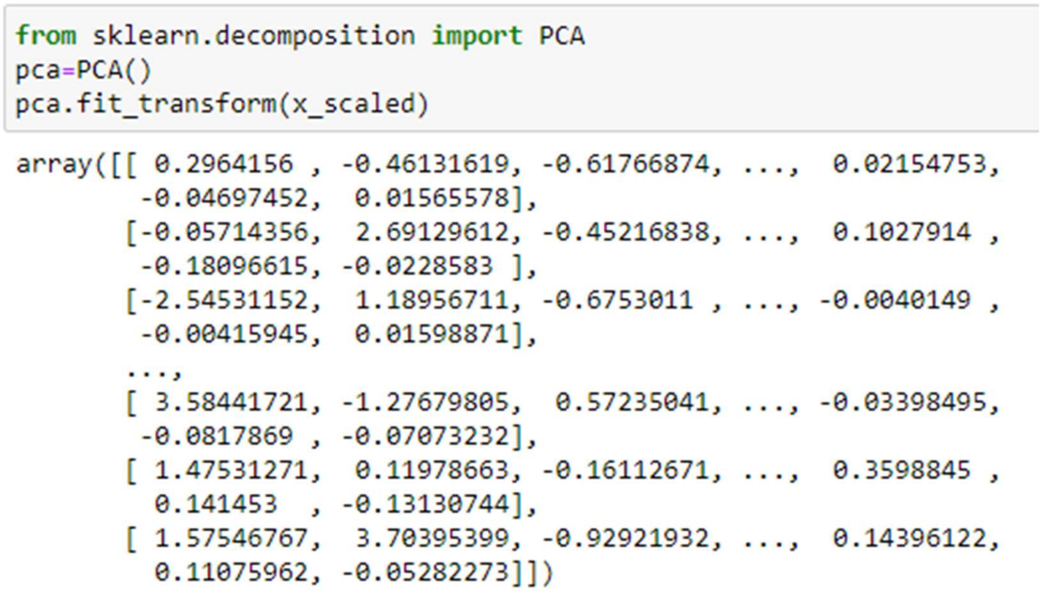
Scaling the data:

Scaled the data using StandardScaler method.

Also the PCA is used to reduce the dimension and selected features based on the PCA plot at 95 % variance factor.



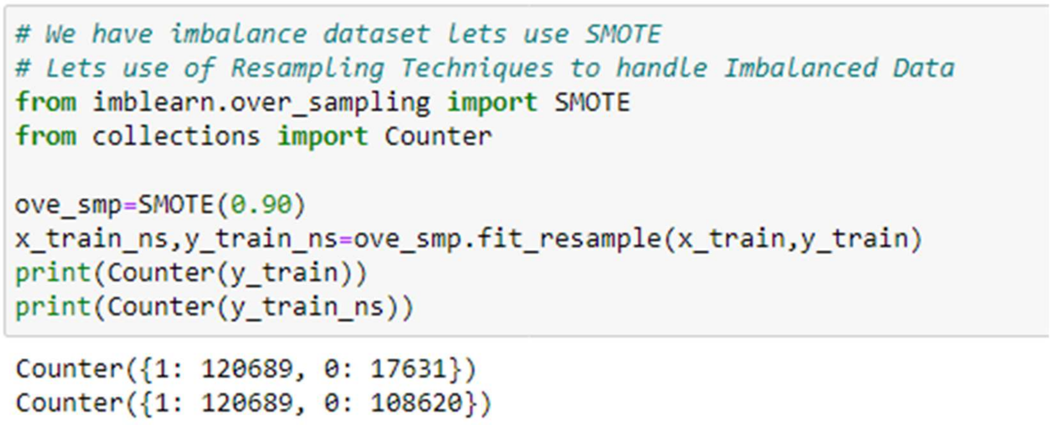
```
# data stadardization
scale=StandardScaler()
x_scaled=scale.fit_transform(x)
```



```
from sklearn.decomposition import PCA
pca=PCA()
pca.fit_transform(x_scaled)
```

```
array([[ 0.2964156 , -0.46131619, -0.61766874, ...,  0.02154753,
        -0.04697452,  0.01565578],
       [-0.05714356,  2.69129612, -0.45216838, ...,  0.1027914 ,
        -0.18096615, -0.0228583 ],
       [-2.54531152,  1.18956711, -0.6753011 , ..., -0.0040149 ,
        -0.00415945,  0.01598871],
       ...,
       [ 3.58441721, -1.27679805,  0.57235041, ..., -0.03398495,
        -0.0817869 , -0.07073232],
       [ 1.47531271,  0.11978663, -0.16112671, ...,  0.3598845 ,
         0.141453  , -0.13130744],
       [ 1.57546767,  3.70395399, -0.92921932, ...,  0.14396122,
         0.11075962, -0.05282273]])
```

Handing Class imbalance problem:
I used Oversampling method, at last I used SMOTE method to handle the class imbalance problem.



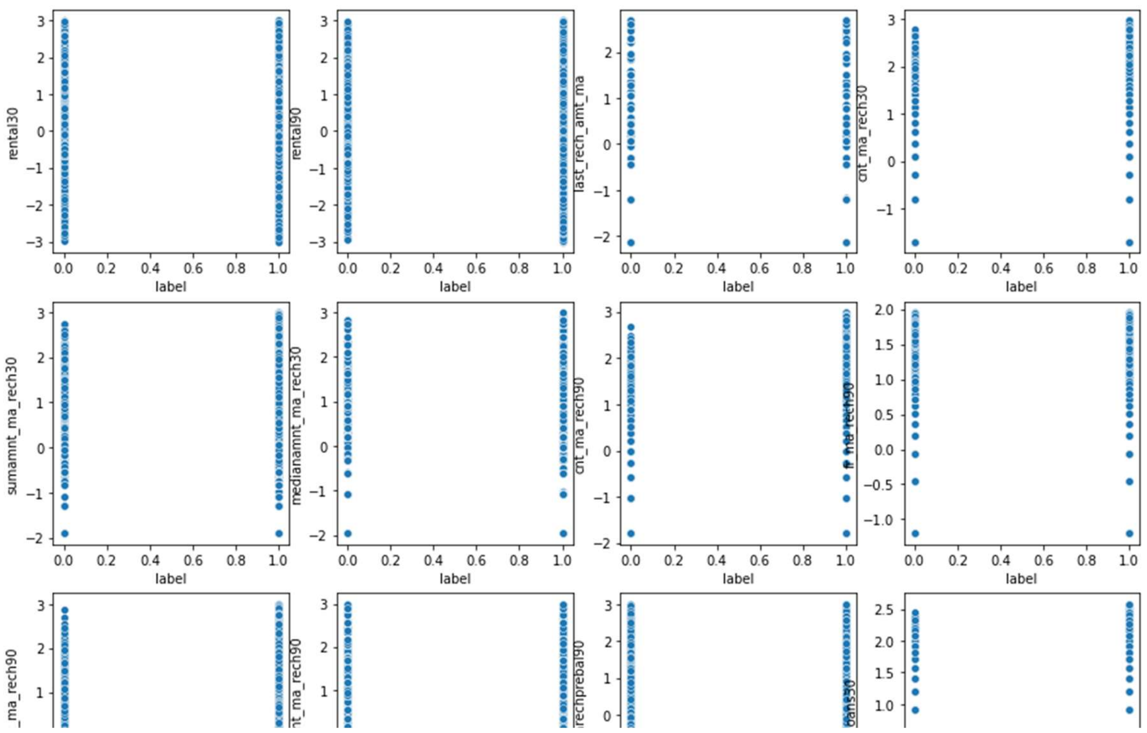
```
# We have imbalance dataset lets use SMOTE
# Lets use of Resampling Techniques to handle Imbalanced Data
from imblearn.over_sampling import SMOTE
from collections import Counter

ove_smp=SMOTE(0.90)
x_train_ns,y_train_ns=ove_smp.fit_resample(x_train,y_train)
print(Counter(y_train))
print(Counter(y_train_ns))
```

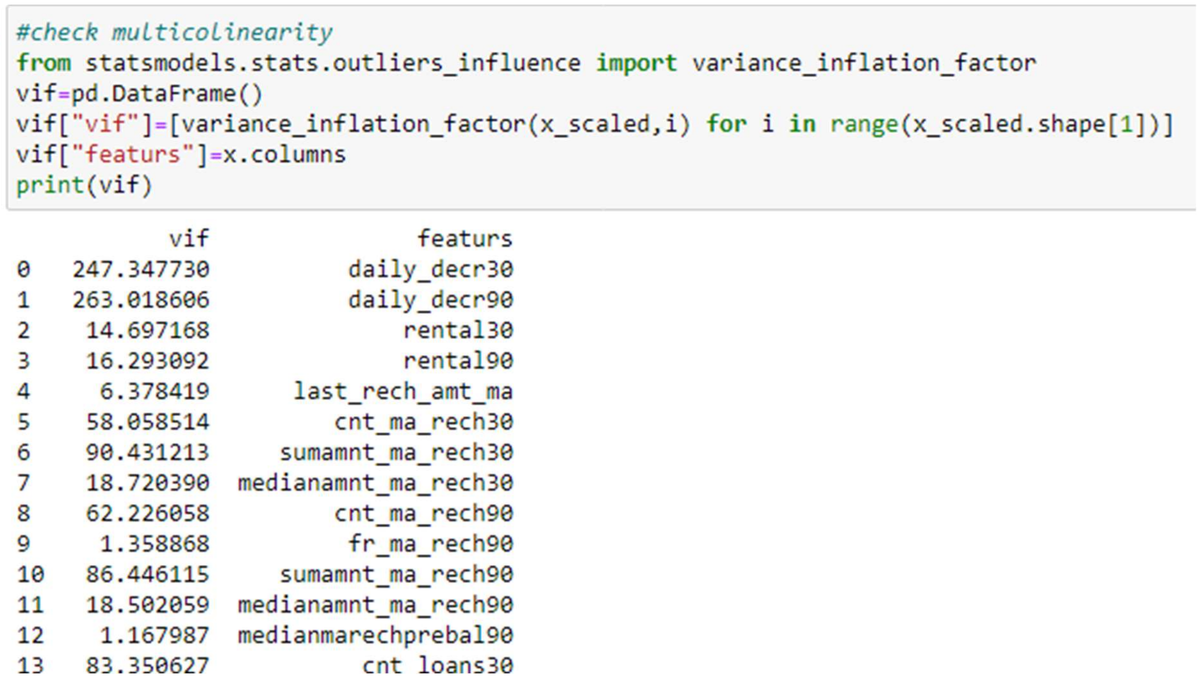
```
Counter({1: 120689, 0: 17631})
Counter({1: 120689, 0: 108620})
```

- Data Inputs- Logic- Output Relationships

  Multicollinearity:

  To check the Input output relation, I have plotted the distribution plot and also use VIF technique to check the multicollinearity. As shown in below screenshot. And detected our problem is multicollinearity problem.





```
#check multicolinearity
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif["vif"]=[variance_inflation_factor(x_scaled,i) for i in range(x_scaled.shape[1])]
vif["featurs"]=x.columns
print(vif)
```

```
           vif              featurs
0   247.347730          daily_decr30
1   263.018606          daily_decr90
2    14.697168              rental30
3    16.293092              rental90
4     6.378419      last_rech_amt_ma
5    58.058514         cnt_ma_rech30
6    90.431213     sumamnt_ma_rech30
7    18.720390  medianamnt_ma_rech30
8    62.226058         cnt_ma_rech90
9     1.358868          fr_ma_rech90
10   86.446115     sumamnt_ma_rech90
11   18.502059  medianamnt_ma_rech90
12    1.167987  medianmarechprebal90
13   83.350627           cnt loans30
```

- State the set of assumptions (if any) related to the problem under consideration

This Dataset is class imbalanced. 87.5% of data is class ‘1’ 12.5% of data is class ‘0.

- Hardware and Software Requirements and Tools Used

Machine: Can use a laptop/desktop.

Operating system: Windows 11, Mac OS X 10.9 Mavericks or Higher

RAM & Processor: 4 GB+ RAM, i3 5th Generation 2.2 Ghz or equivalent/higher .

Tools Used: Jupyter Notebook, Microsoft Excel.

Libraries Used :

```
import pandas as pd       # for data manipulation

import numpy as np        # for mathematical calculations

import seaborn as sns      # for data visualization

import matplotlib.pyplot as plt  #for graphical analysis

%matplotlib inline

from scipy.stats import zscore # to remove outliers

from sklearn.preprocessing import StandardScaler  # for normalize the model

from sklearn.preprocessing import LabelEncoder  # to convert object into int

from sklearn.model_selection import train_test_split  # for train and test model

import warnings                   # to ignore any warnings

warnings.filterwarnings("ignore")

from sklearn import metrics  # for model evaluation

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  I dropped the 14 columns whose correlation with target variable is less than 0.01, Since they don't impact the result.

  I have dropped the duplicated to avoid impact of the same on result.

  Also, dropped the multicollinearity columns to get the better result.

  Reduced the skewness using power transform method.

  Used SMOTE method to deal with class imbalance problem.

  Removed the Outliers by using zscore method to build best model.

- Testing of Identified Approaches (Algorithms)

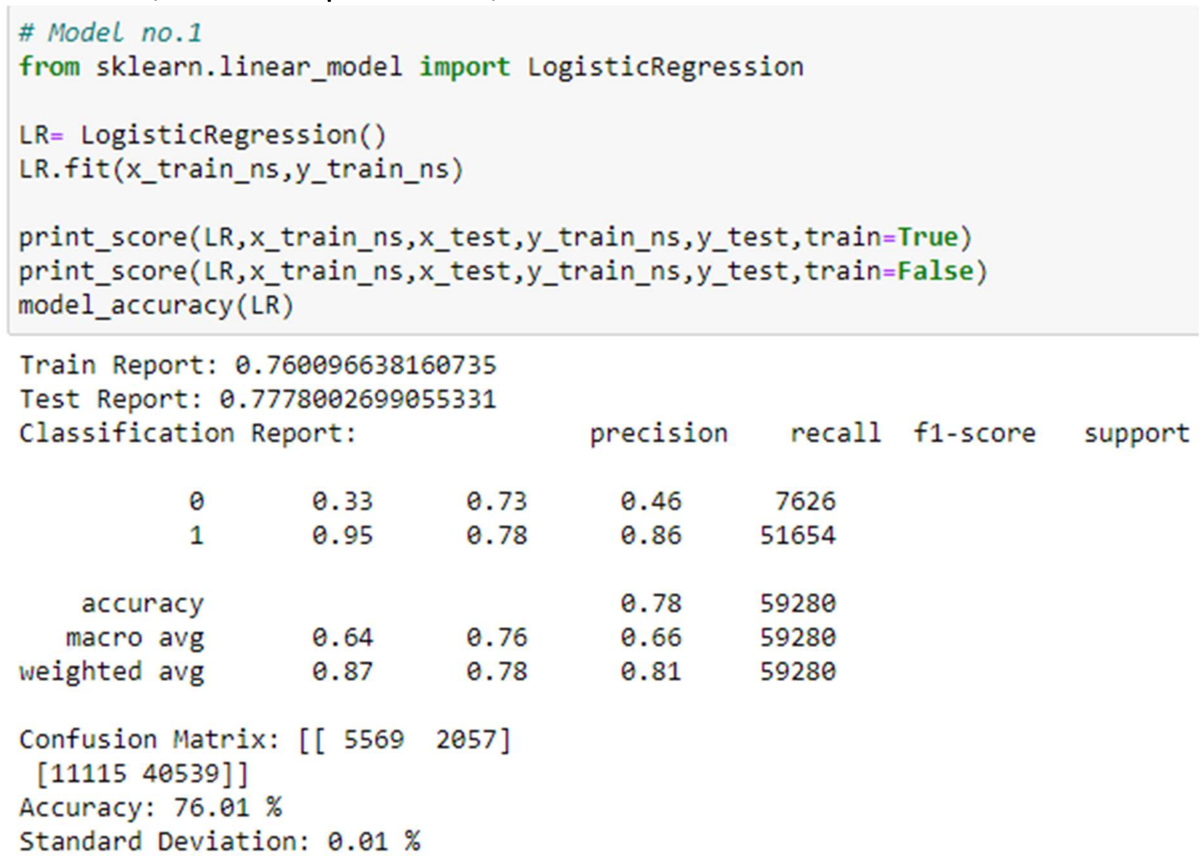
  I have selected 5 models for the prediction which is used for the training and testing.

  1) Logistic Regression from sklearn.linear_model
  2) DecisionTreeClassifier from sklearn.tree
  3) RandomForestClassifier from sklearn.ensemble
  4) ADABoostClassifier from sklearn.ensemble
  5) XGBClassifier from XGBoost

- Run and Evaluate selected models

1) **Logistic Regression from sklearn.linear_model:**
   Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is binary, which means there would be only two possible classes 1 (stands for success/yes) or 0 (stands for failure/no). Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that

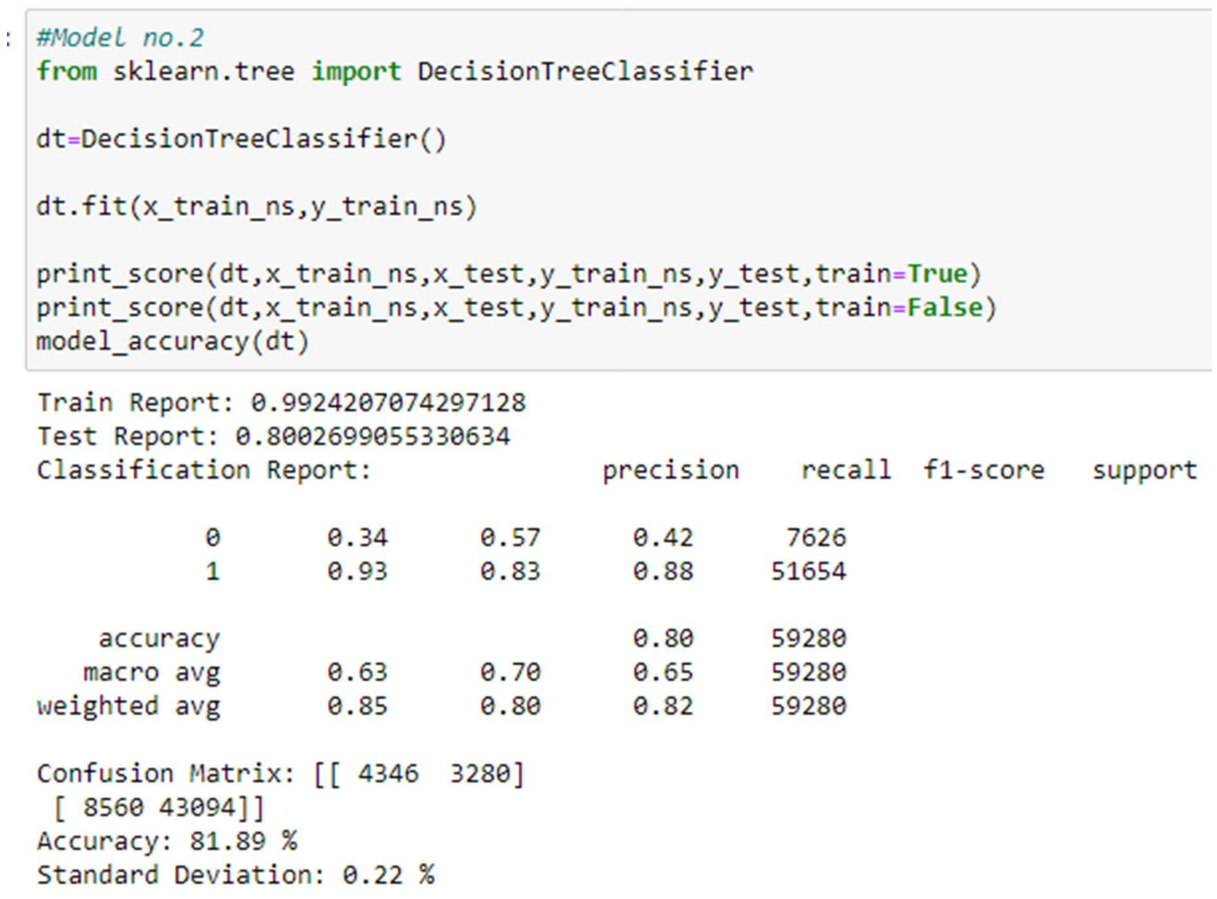
can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.



```python
# Model no.1
from sklearn.linear_model import LogisticRegression

LR= LogisticRegression()
LR.fit(x_train_ns,y_train_ns)

print_score(LR,x_train_ns,x_test,y_train_ns,y_test,train=True)
print_score(LR,x_train_ns,x_test,y_train_ns,y_test,train=False)
model_accuracy(LR)
```

```
Train Report: 0.760096638160735
Test Report: 0.7778002699055331
Classification Report:                precision    recall  f1-score   support

           0       0.33      0.73      0.46      7626
           1       0.95      0.78      0.86     51654

    accuracy                           0.78     59280
   macro avg       0.64      0.76      0.66     59280
weighted avg       0.87      0.78      0.81     59280

Confusion Matrix: [[ 5569  2057]
 [11115 40539]]
Accuracy: 76.01 %
Standard Deviation: 0.01 %
```

2) **DecisionTreeClassifier from sklearn.tree:**
Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. The two main entities of a tree are decision nodes, where the data is split and leaves, where we get the outcome.

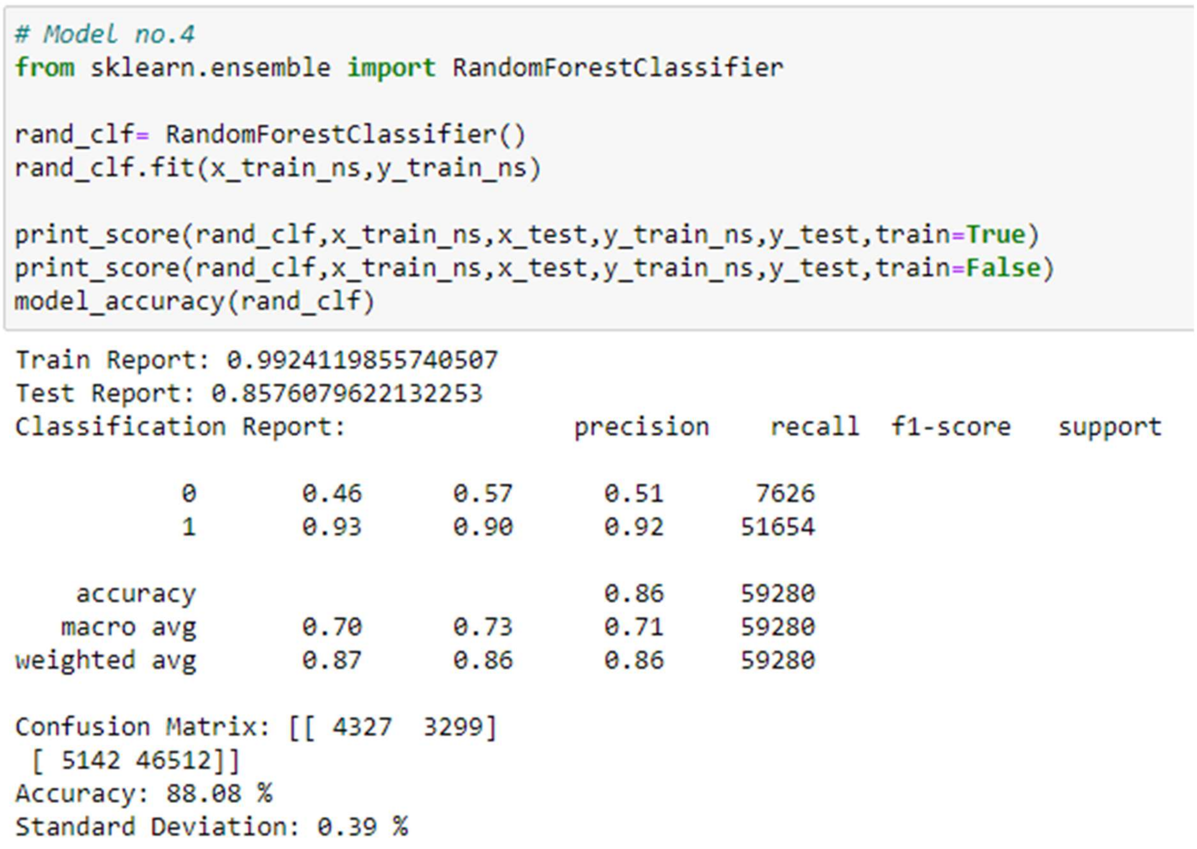


```
#Model no.2
from sklearn.tree import DecisionTreeClassifier

dt=DecisionTreeClassifier()

dt.fit(x_train_ns,y_train_ns)

print_score(dt,x_train_ns,x_test,y_train_ns,y_test,train=True)
print_score(dt,x_train_ns,x_test,y_train_ns,y_test,train=False)
model_accuracy(dt)
```

```
Train Report: 0.9924207074297128
Test Report: 0.8002699055330634
Classification Report:               precision    recall  f1-score   support

           0       0.34      0.57      0.42      7626
           1       0.93      0.83      0.88     51654

    accuracy                           0.80     59280
   macro avg       0.63      0.70      0.65     59280
weighted avg       0.85      0.80      0.82     59280

Confusion Matrix: [[ 4346  3280]
 [ 8560 43094]]
Accuracy: 81.89 %
Standard Deviation: 0.22 %
```

3) **RandomForestClassifier from sklearn.ensemble:**

As we know that a forest is made up of trees and more trees means more robust forest. Similarly, a random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

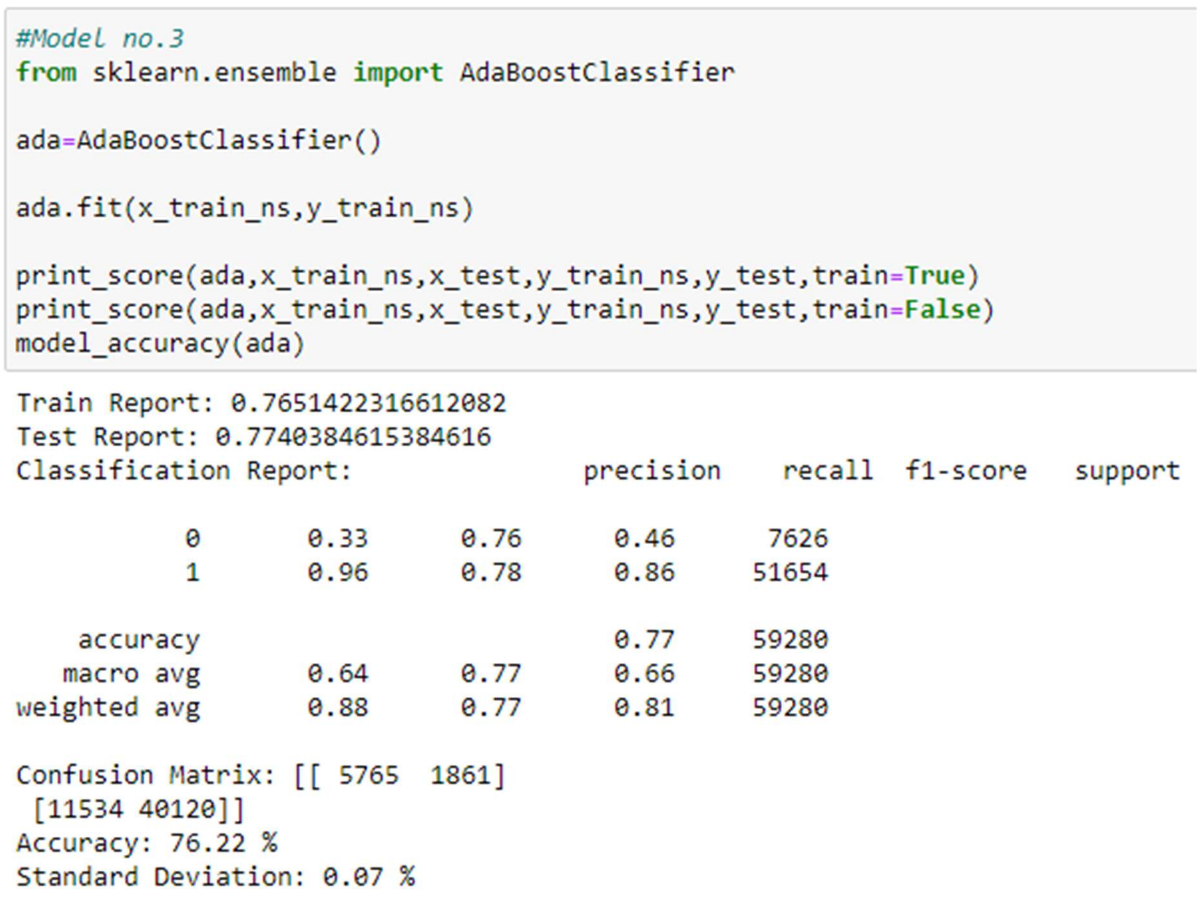


```
# Model no.4
from sklearn.ensemble import RandomForestClassifier

rand_clf= RandomForestClassifier()
rand_clf.fit(x_train_ns,y_train_ns)

print_score(rand_clf,x_train_ns,x_test,y_train_ns,y_test,train=True)
print_score(rand_clf,x_train_ns,x_test,y_train_ns,y_test,train=False)
model_accuracy(rand_clf)
```

```
Train Report: 0.9924119855740507
Test Report: 0.8576079622132253
Classification Report:                precision    recall  f1-score   support

           0       0.46      0.57      0.51      7626
           1       0.93      0.90      0.92     51654

    accuracy                           0.86     59280
   macro avg       0.70      0.73      0.71     59280
weighted avg       0.87      0.86      0.86     59280

Confusion Matrix: [[ 4327  3299]
 [ 5142 46512]]
Accuracy: 88.08 %
Standard Deviation: 0.39 %
```

4) ADABoostClassifier:

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

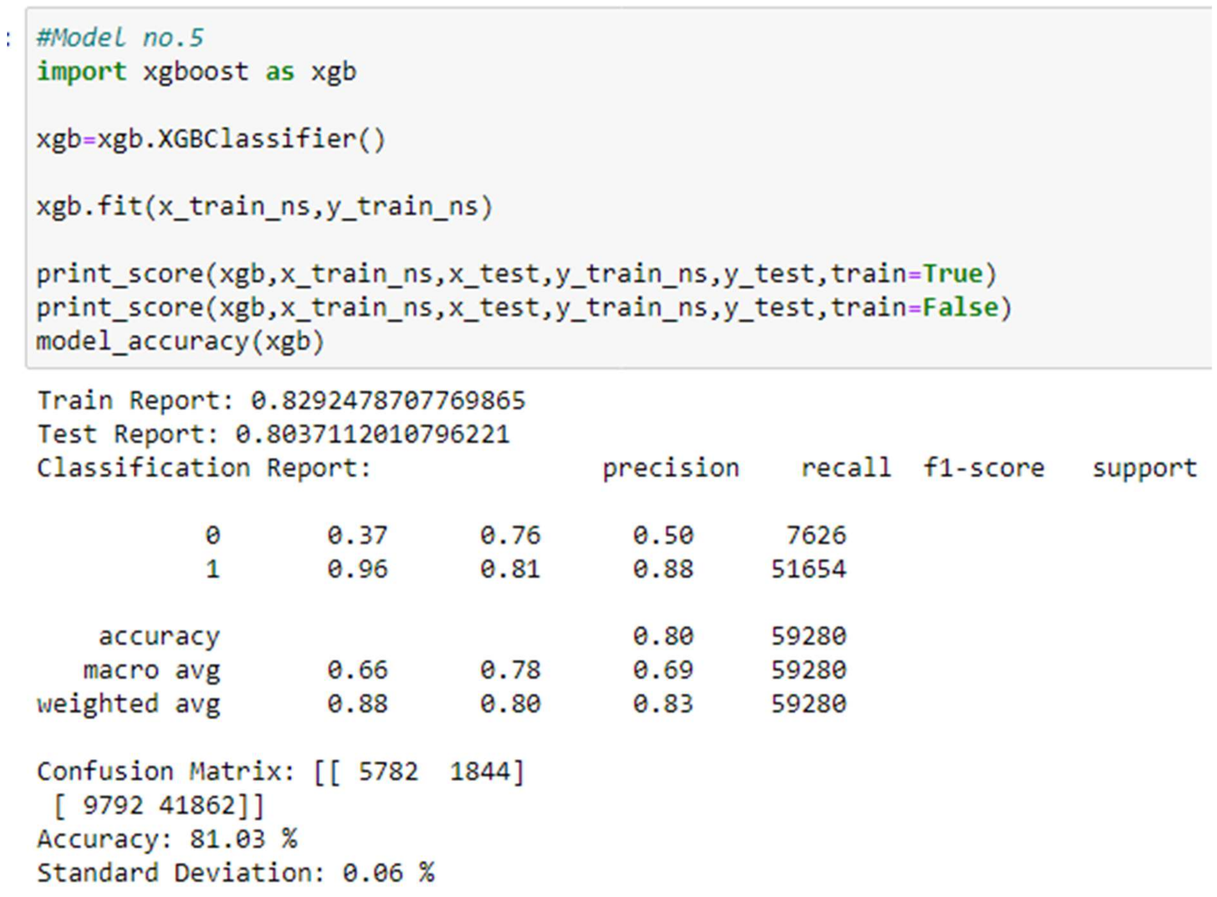


```
#Model no.3
from sklearn.ensemble import AdaBoostClassifier

ada=AdaBoostClassifier()

ada.fit(x_train_ns,y_train_ns)

print_score(ada,x_train_ns,x_test,y_train_ns,y_test,train=True)
print_score(ada,x_train_ns,x_test,y_train_ns,y_test,train=False)
model_accuracy(ada)
```

```
Train Report: 0.7651422316612082
Test Report: 0.7740384615384616
Classification Report:              precision    recall  f1-score   support

           0       0.33      0.76      0.46      7626
           1       0.96      0.78      0.86     51654

    accuracy                           0.77     59280
   macro avg       0.64      0.77      0.66     59280
weighted avg       0.88      0.77      0.81     59280

Confusion Matrix: [[ 5765  1861]
 [11534 40120]]
Accuracy: 76.22 %
Standard Deviation: 0.07 %
```

5) **XGBClassifier from XGBoost:**

XGBoost is short for "eXtreme Gradient Boosting." The "eXtreme" refers to speed enhancements such as parallel computing and cache awareness that makes XGBoost approximately 10 times faster than traditional Gradient Boosting. In addition, XGBoost includes a unique split-finding algorithm to optimise trees, along with built-in regularisation that reduces over-fitting. Generally speaking, XGBoost is a faster, more accurate version of Gradient Boosting.



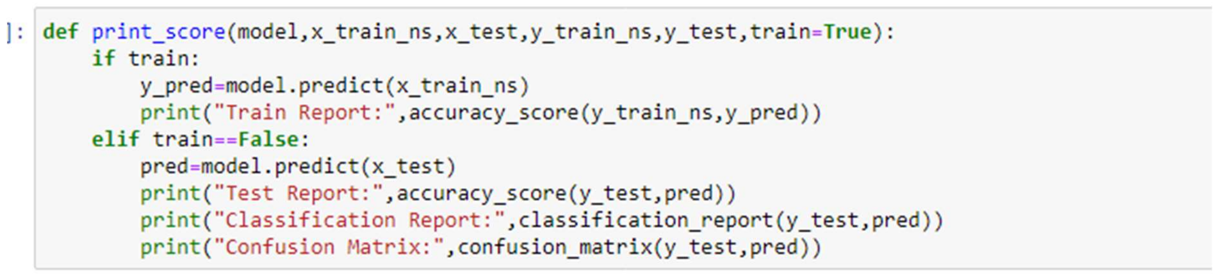
```
#Model no.5
import xgboost as xgb

xgb=xgb.XGBClassifier()

xgb.fit(x_train_ns,y_train_ns)

print_score(xgb,x_train_ns,x_test,y_train_ns,y_test,train=True)
print_score(xgb,x_train_ns,x_test,y_train_ns,y_test,train=False)
model_accuracy(xgb)
```

```
Train Report: 0.8292478707769865
Test Report: 0.8037112010796221
Classification Report:               precision    recall  f1-score   support

           0       0.37      0.76      0.50      7626
           1       0.96      0.81      0.88     51654

    accuracy                           0.80     59280
   macro avg       0.66      0.78      0.69     59280
weighted avg       0.88      0.80      0.83     59280

Confusion Matrix: [[ 5782  1844]
 [ 9792 41862]]
Accuracy: 81.03 %
Standard Deviation: 0.06 %
```

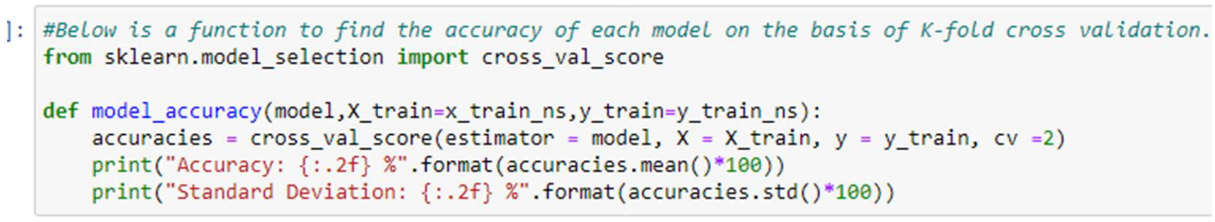
- Key Metrics for success in solving problem under consideration

  I have used accuracy score, ROC AUC Curve, classification report & confusion matrix as metrices.

  We got **our** best model looking at accuracy, ROC AUC Curve & confusion matrix is **XGBClassifier** with Kfold cross validation method with the accuracy score of 86%.



```
]: def print_score(model,x_train_ns,x_test,y_train_ns,y_test,train=True):
       if train:
           y_pred=model.predict(x_train_ns)
           print("Train Report:",accuracy_score(y_train_ns,y_pred))
       elif train==False:
           pred=model.predict(x_test)
           print("Test Report:",accuracy_score(y_test,pred))
           print("Classification Report:",classification_report(y_test,pred))
           print("Confusion Matrix:",confusion_matrix(y_test,pred))
```

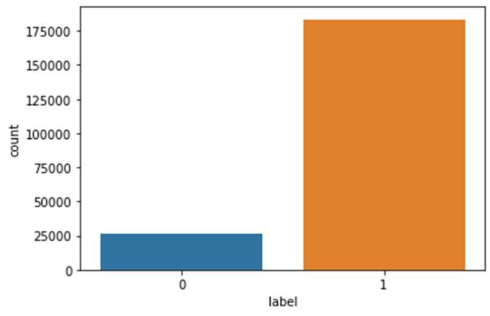


```
]: #Below is a function to find the accuracy of each model on the basis of K-fold cross validation.
   from sklearn.model_selection import cross_val_score

   def model_accuracy(model,X_train=x_train_ns,y_train=y_train_ns):
       accuracies = cross_val_score(estimator = model, X = X_train, y = y_train, cv =2)
       print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
       print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

- Visualizations

Univariate Plots

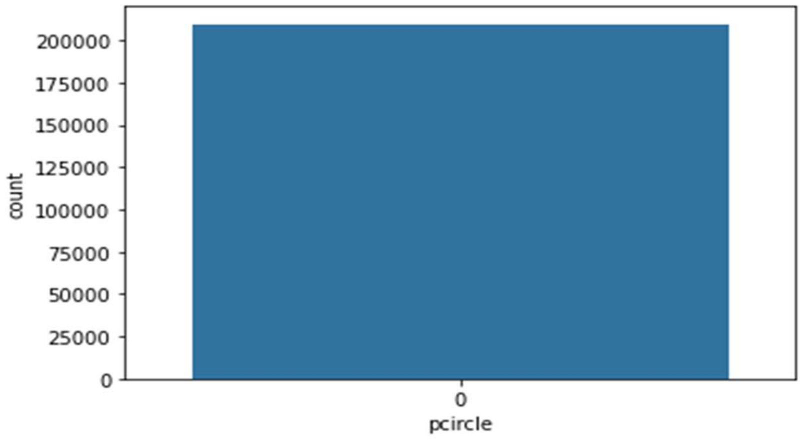
```
: #plot each class frequency
  sns.countplot(x='label',data=df)
  plt.show()
  print(df['label'].value_counts())
```



```
1    183429
0     26133
Name: label, dtype: int64
```

From above graph we can see that the amount of people who has paid back the credit amount within 5 days of issuing the loan is 183429 and those who not paid within 5 days are 26133.

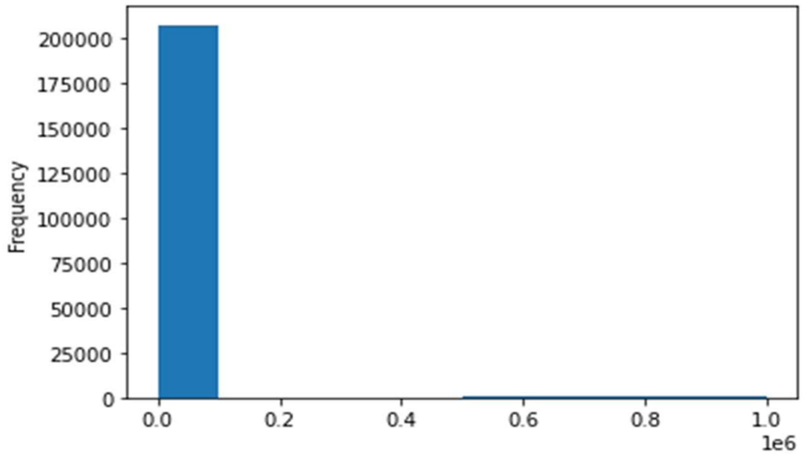
```
#plot each class frequency
sns.countplot(x='pcircle',data=df)
plt.show()
print(df['pcircle'].value_counts())
```



```
0    209562
Name: pcircle, dtype: int64
```

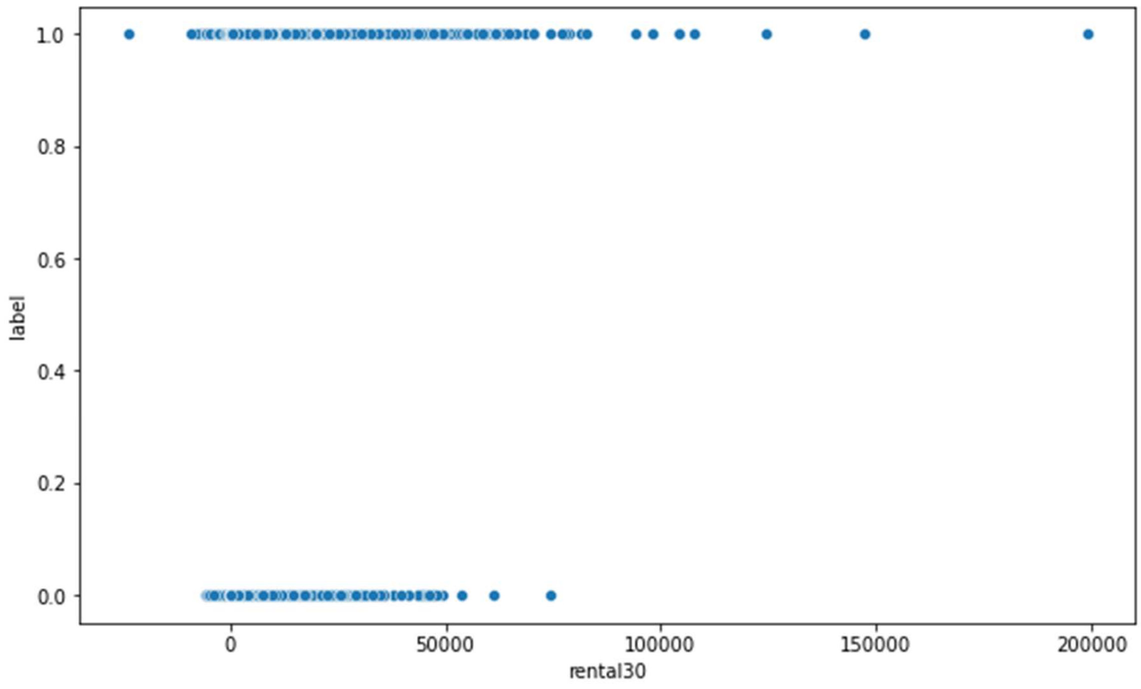
As we see there is only one telecom circle

```
#plot each class frequency
df['aon'].plot.hist()
plt.show()
print(df['aon'].value_counts())
```



Bivariate Plot

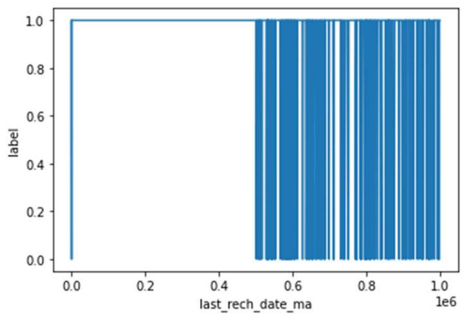
```
#Bivariant graph
plt.figure(figsize =(10, 6))
sns.scatterplot(x ='rental30', y ='label', data = df)
plt.show()
print(df.groupby("label")['rental30'].value_counts())
```



```
label  rental30
0      0.00          3626
```

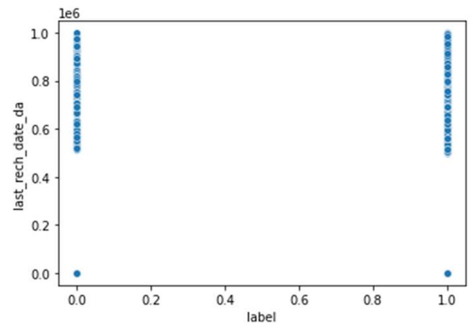
From graph we can see that the success of user paid back the credit amount within 5 days of issuing the loan has above 100000 (In Indonesian Rupiah) Average main account balance over last 30 days

```
#Bivariant graph
plt.figure(figsize =(6, 4))
sns.lineplot(x ='last_rech_date_ma', y ='label', data = df)
plt.show()
```



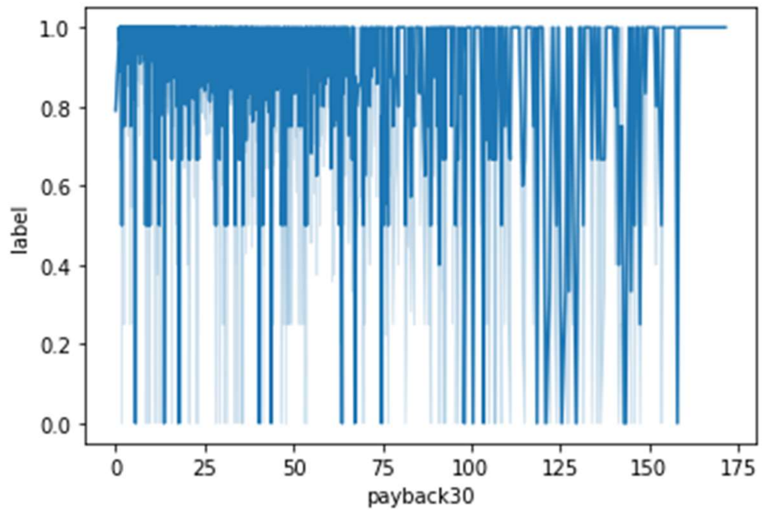
From graph we can see the those who has paid back credit amount within 5 days those user has at least once recharge of main account, similarly those who have not paid they have not recharge there main account.

```
#Bivariant graph
plt.figure(figsize =(6, 4))
sns.scatterplot(y ='last_rech_date_da', x ='label', data = df)
plt.show()
```



From graph we can see there are similar user those who paid the credit amount and recharge the data account & those who not paid the credit amount within 5 days who recharge the data account.

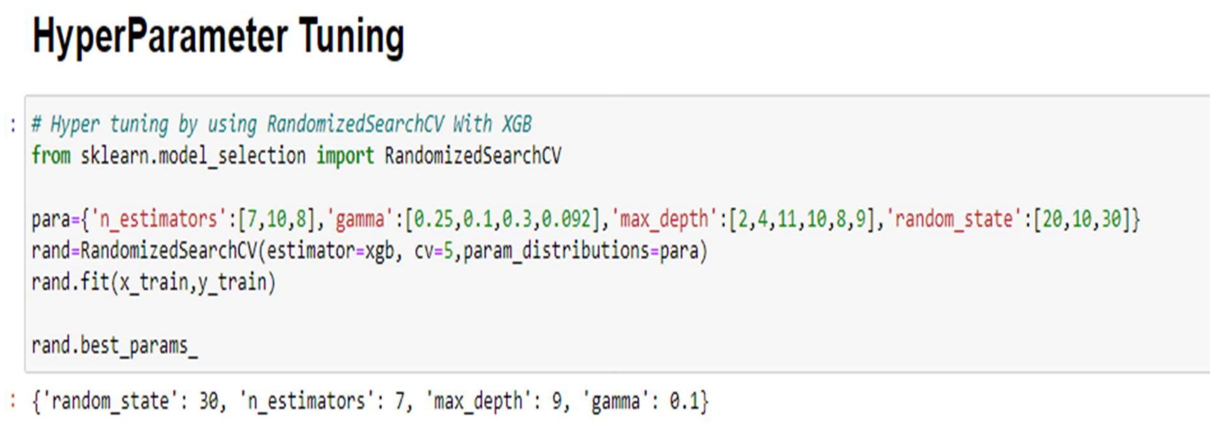
```
#Bivariant graph
plt.figure(figsize =(6, 4))
sns.lineplot(x ='payback30', y ='label', data = df)
plt.show()
```



From graph we can see target variable & payback30 feature has linear relation.

- Interpretation of the Results

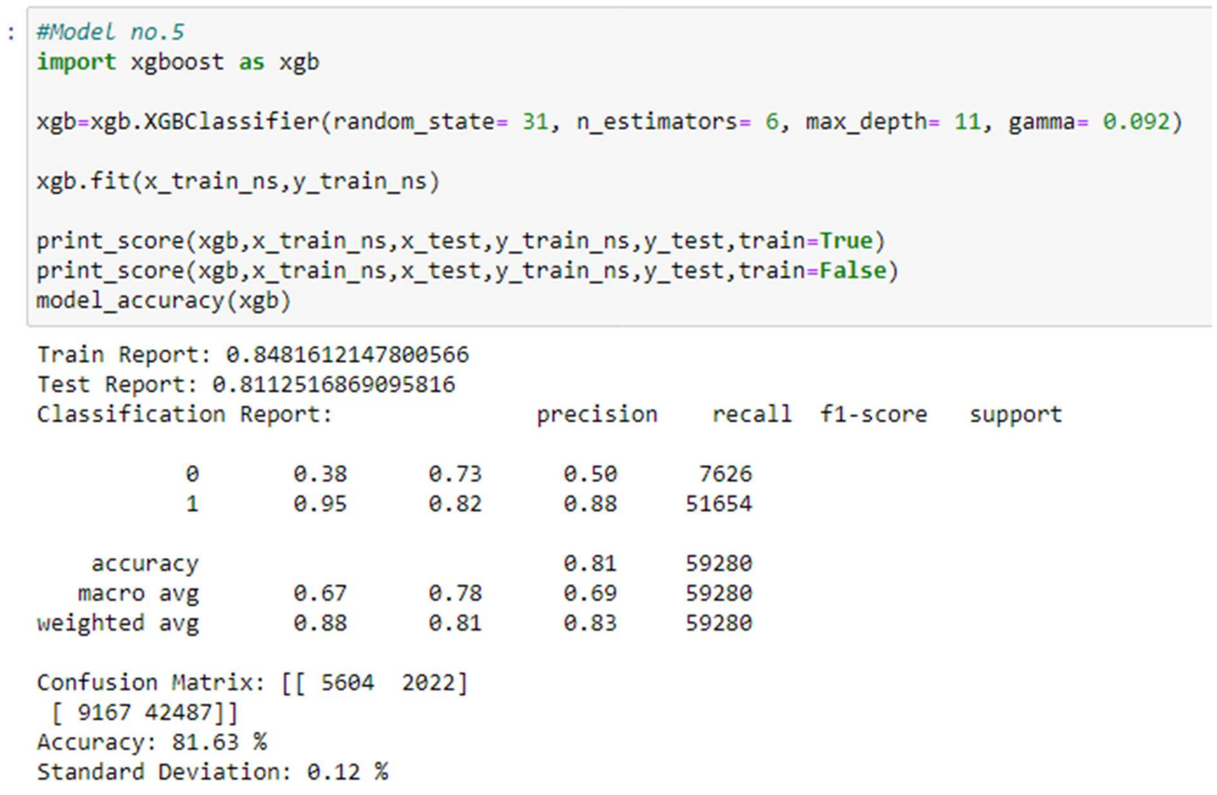
Looking at the accuracy and ROC curve i m selecting XGBClassifier further to get better accuracy I have used **RandomizedSearchCV** hyper tuning parameter.



HyperParameter Tuning

```
# Hyper tuning by using RandomizedSearchCV With XGB
from sklearn.model_selection import RandomizedSearchCV

para={'n_estimators':[7,10,8],'gamma':[0.25,0.1,0.3,0.092],'max_depth':[2,4,11,10,8,9],'random_state':[20,10,30]}
rand=RandomizedSearchCV(estimator=xgb, cv=5,param_distributions=para)
rand.fit(x_train,y_train)

rand.best_params_
```

```
{'random_state': 30, 'n_estimators': 7, 'max_depth': 9, 'gamma': 0.1}
```



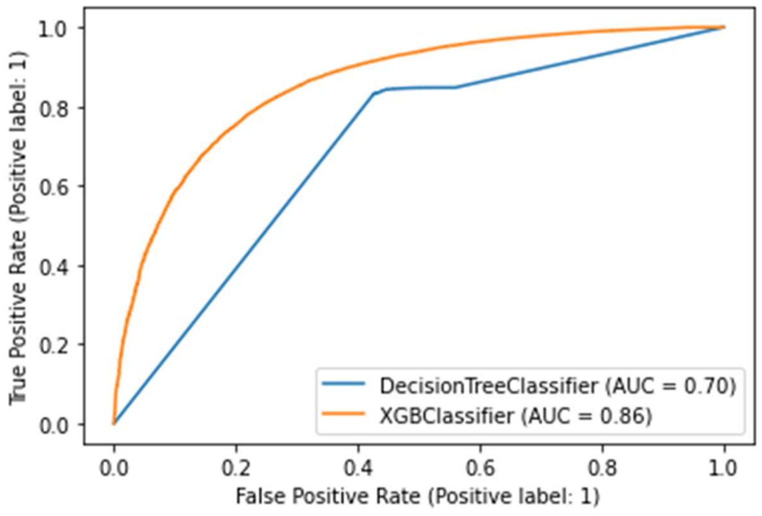
```
#Model no.5
import xgboost as xgb

xgb=xgb.XGBClassifier(random_state= 31, n_estimators= 6, max_depth= 11, gamma= 0.092)

xgb.fit(x_train_ns,y_train_ns)

print_score(xgb,x_train_ns,x_test,y_train_ns,y_test,train=True)
print_score(xgb,x_train_ns,x_test,y_train_ns,y_test,train=False)
model_accuracy(xgb)
```

```
Train Report: 0.8481612147800566
Test Report: 0.8112516869095816
Classification Report:               precision    recall  f1-score   support

           0       0.38      0.73      0.50      7626
           1       0.95      0.82      0.88     51654

    accuracy                           0.81     59280
   macro avg       0.67      0.78      0.69     59280
weighted avg       0.88      0.81      0.83     59280

Confusion Matrix: [[ 5604  2022]
 [ 9167 42487]]
Accuracy: 81.63 %
Standard Deviation: 0.12 %
```

To finalize or to select the best model I have plotted ROC_AUC curve based on that curve I have selected XGBClassifier as shown below.

```
disp=plot_roc_curve(dt,x_test,y_test)
plot_roc_curve(xgb,x_test,y_test,ax=disp.ax_)
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x189209c4d90>
```



Saving the model:

```
#save model
import pickle
Filename='Finalized_model_Micro_Credit_Card.pickle'
pickle.dump(xgb,open(Filename,'wb'))
```

CONCLUSION

- Key Findings and Conclusions of the Study

  The key findings, inferences, observations from the whole problem are that I found out how to handle the dataset which is class imbalanced and use for metrics like f1 score. I observed that data with least correlation with target variable do not impact the result.

- Learning Outcomes of the Study in respect of Data Science

  Visualization is very useful for detection of outliers in particular columns, distribution of data, AUC ROC curves, Heat map for correlation and null values

I choose XGBClassifier algorithm as my final model since it was

having least difference between its cross-validation score and high auc curve.

I faced the challenges in visualizing all the columns at once when there are some object datatypes were in the columns. Then i dropped those columns which were not useful.

- ## Limitations of this work and Scope for Future Work

Limitations of this solutions is that I could have used hyperparameter tuning for various algorithms, since it was consuming more time.

Future scope of this project is that Micro Credit Services can be also implemented in DTH, Supermarkets etc.