NAME OF THE PROJECT

Micro Credit Defaulter Project

Submitted by:

Arshi Maitra

Acknowledgement:

This project has been completed with the help of training documents and live classes recordings from Data Trained Education. Few helps on coding have also been taken from few data science websites like Toward Data Science, Geek for Geeks, Stack Overflow.

INTRODUCTION

Microfinance is a form of financial service which provides small loans and other financial services to poor and low-income households. In good times, microfinance helps families and small businesses to prosper, and at times of crisis it can help them cope and rebuild. The objective of microfinance is similar to that of microcredit; its goal is to provide financial services to help encourage entrepreneurs in impoverished nations to act on their ideas and obtain the financial tools available to do so and to eventually become self-sustainable.

Like a bank, a microfinance institution is a provider of credit. However, the size of the loans are smaller than those granted by traditional banks. These small loans are known as microcredit. The clients of an MFI are often microentrepreneurs in need of economic support to launch their business. This type of client is considered too risky by traditional banks because they cannot provide real collateral and because they tend to work in the informal sector of the economy.

Before granting the loan, MFIs analyse the clients' willingness and ability to pay. MFIs usually carry out a field survey to gather as much information as possible, not only from the future entrepreneur, but also from people who know them.

Below is the dataset available one such client that is in Telecom Industry. They are a fixed wireless telecommunications network provider. They are collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed to be defaulter if he deviates from the path of paying back the loaned amount within the time duration of 5 days. For the loan amount of 5 (in Indonesian Rupiah), payback amount should be 6 (in Indonesian Rupiah), while, for the loan amount of 10 (in Indonesian Rupiah), the payback amount should be 12 (in Indonesian Rupiah).

The main goal of the project will be to predict in terms of a probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance of loan.

Analytical Problem Framing

Importing of necessary libraries:

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest, f_classif, f_regression
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import accuracy_score,confusion_matrix,roc_curve, roc_auc_score
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import GridSearchCV
from prettytable import PrettyTable
import warnings
warnings.filterwarnings('ignore') #Importing the necessary libraries
```

Data preprocessing/Data Cleaning:

In [2]:

```
Data=pd.read_csv("Data file.csv") # Loadng the data
```

In [3]:

```
Data.head()
```

Out[3]:

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30	medianami
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	...	6.0	
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	...	12.0	
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	...	6.0	
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	...	6.0	
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	...	6.0	

5 rows × 37 columns

Going through the columns of the data:

```
In [6]:  1 Data.columns #checking the total number of columns

Out[6]: Index(['label', 'msisdn', 'aon', 'daily_decr30', 'daily_decr90', 'rental30',
           'rental90', 'last_rech_date_ma', 'last_rech_date_da',
           'last_rech_amt_ma', 'cnt_ma_rech30', 'fr_ma_rech30',
           'sumamnt_ma_rech30', 'medianamnt_ma_rech30', 'medianmarechprebal30',
           'cnt_ma_rech90', 'fr_ma_rech90', 'sumamnt_ma_rech90',
           'medianamnt_ma_rech90', 'medianmarechprebal90', 'cnt_da_rech30',
           'fr_da_rech30', 'cnt_da_rech90', 'fr_da_rech90', 'cnt_loans30',
           'amnt_loans30', 'maxamnt_loans30', 'medianamnt_loans30', 'cnt_loans90',
           'amnt_loans90', 'maxamnt_loans90', 'medianamnt_loans90', 'payback30',
           'payback90', 'pcircle', 'pdate'],
          dtype='object')
```

Checking the null values from data:

```
In [9]:  1 Data.isnull().sum() #Checking for null values

Out[9]: label                   0
        msisdn                  0
        aon                     0
        daily_decr30            0
        daily_decr90            0
        rental30                0
        rental90                0
        last_rech_date_ma       0
        last_rech_date_da       0
        last_rech_amt_ma        0
        cnt_ma_rech30           0
        fr_ma_rech30            0
        sumamnt_ma_rech30       0
        medianamnt_ma_rech30    0
        medianmarechprebal30    0
        cnt_ma_rech90           0
        fr_ma_rech90            0
        sumamnt_ma_rech90       0
        medianamnt_ma_rech90    0
        medianmarechprebal90    0
        cnt_da_rech30           0
        fr_da_rech30            0
        cnt_da_rech90           0
        fr_da_rech90            0
        cnt_loans30             0
        amnt_loans30            0
        maxamnt_loans30         0
        medianamnt_loans30      0
        cnt_loans90             0
        amnt_loans90            0
```

There is no null values.

Encoding the object data types:

There were 3 object data type which was the customer's telephone number, telecom circle as well as the data. As those columns were not needed, hence they were dropped off.

Selecting K Best feature selection method:

As we had 33 columns hence the K Best feature was used to get the 30 best column.

In [56]:

```
Features= SelectKBest(score_func=f_regression, k=30)
fit=Features.fit(X,Y)
scores=pd.DataFrame(fit.scores_)
columns=pd.DataFrame(X.columns)
```

In [57]:

```
Total_Score=pd.concat([columns,scores],axis=1)
Total_Score.columns=['Column','Score']
print(Total_Score.nlargest(30,'Score'))
```

Checking the scores:

```
13          cnt_ma_rech90  147884.585884
15       sumamnt_ma_rech90  130101.001211
8           cnt_ma_rech30  129722.510857
10       sumamnt_ma_rech30  118609.542232
26            cnt_loans90   79206.001627
27           amnt_loans90   77047.217154
31              payback90   69989.605154
30              payback30   66477.109482
9            fr_ma_rech30   63350.051024
11    medianamnt_ma_rech30   62551.767587
2            daily_decr90   59435.090618
22            cnt_loans30   58216.565274
23           amnt_loans30   58126.471552
1            daily_decr30   57837.363830
7         last_rech_amt_ma   53591.410834
14           fr_ma_rech90   49733.098654
16    medianamnt_ma_rech90   48812.026785
4                rental90    5758.215406
17    medianmarechprebal90    5628.594906
28         maxamnt_loans90    5164.489765
3                rental30    3293.097415
25      medianamnt_loans30    2276.668400
29      medianamnt_loans90    1372.769876
0                     aon     681.475008
5        last_rech_date_ma     614.686606
12    medianmarechprebal30     569.819220
20          cnt_da_rech90     362.280821
18          cnt_da_rech30      82.790457
21           fr_da_rech90      44.502147
6        last_rech_date_da      29.282351
```

Therefore now received the best 30 features for the target variable.

The target variable is imbalanced in nature with more data classified as a success than failure, this might make the model biased. Therefore to remove any such biasedness resampling is needed.

Importing Resample:

In [41]:
```
from sklearn.utils import resample
```

In [42]:
```
failure=Data[Data.label==0]
success=Data[Data.label==1]
```

Upsampling the failure data:

In [43]:
```
failure_upsamples=resample(failure,replace=True,n_samples=len(success),random_state=27)
```

In [44]:
```
upsampled=pd.concat([success,failure_upsamples])
```

In [45]:
```
Data=upsampled
```

Balanced Target Data:

In [46]:
```
Data['label'].value_counts()
```

Out[46]:
```
1    183431
0    183431
Name: label, dtype: int64
```

In [47]:
```
# Target data has now been balanced
```

Skewness Score:

```
Out[50]: medianmarechprebal90    45.439607
         cnt_da_rech90           27.093817
         fr_da_rech90            25.812222
         cnt_da_rech30           18.468319
         maxamnt_loans30         17.954840
         cnt_loans90             16.811494
         last_rech_date_ma       15.335222
         last_rech_date_da       15.060253
         fr_ma_rech30            14.974802
         fr_da_rech30            14.807260
         medianmarechprebal30    14.112560
         aon                     10.175736
         payback30                8.213847
         payback90                6.864790
         sumamnt_ma_rech30        6.778993
         sumamnt_ma_rech90        5.562973
         medianamnt_loans90       5.288104
         daily_decr90             5.282585
         medianamnt_loans30       5.005464
         last_rech_amt_ma         4.920042
         medianamnt_ma_rech90     4.861770
         daily_decr30             4.856573
         rental90                 4.570556
         rental30                 4.504373
         medianamnt_ma_rech30     4.392988
         cnt_ma_rech90            3.905038
         amnt_loans90             3.891698
         amnt_loans30             3.614976
         cnt_ma_rech30            3.599915
         cnt_loans30              3.359139
         fr_ma_rech90             2.588747
```

```
In [49]: X=Data.drop('label',axis=1)
         Y=Data['label'] #Seperating the target and classes
```

```
In [50]: X.skew().sort_values(ascending=False) #Checking the skewness
```

Importing Power Transformation.

As the data is skewed it will affect the prediction score. Therefore data has to be transformed before taking any further action:

```
In [51]: from sklearn.preprocessing import power_transform
```

```
In [52]: New_X=power_transform(X)
```

```
In [53]: pd.DataFrame(New_X,columns=X.columns).skew().sort_values(ascending=False) # transforming the data to reduce skewness
```

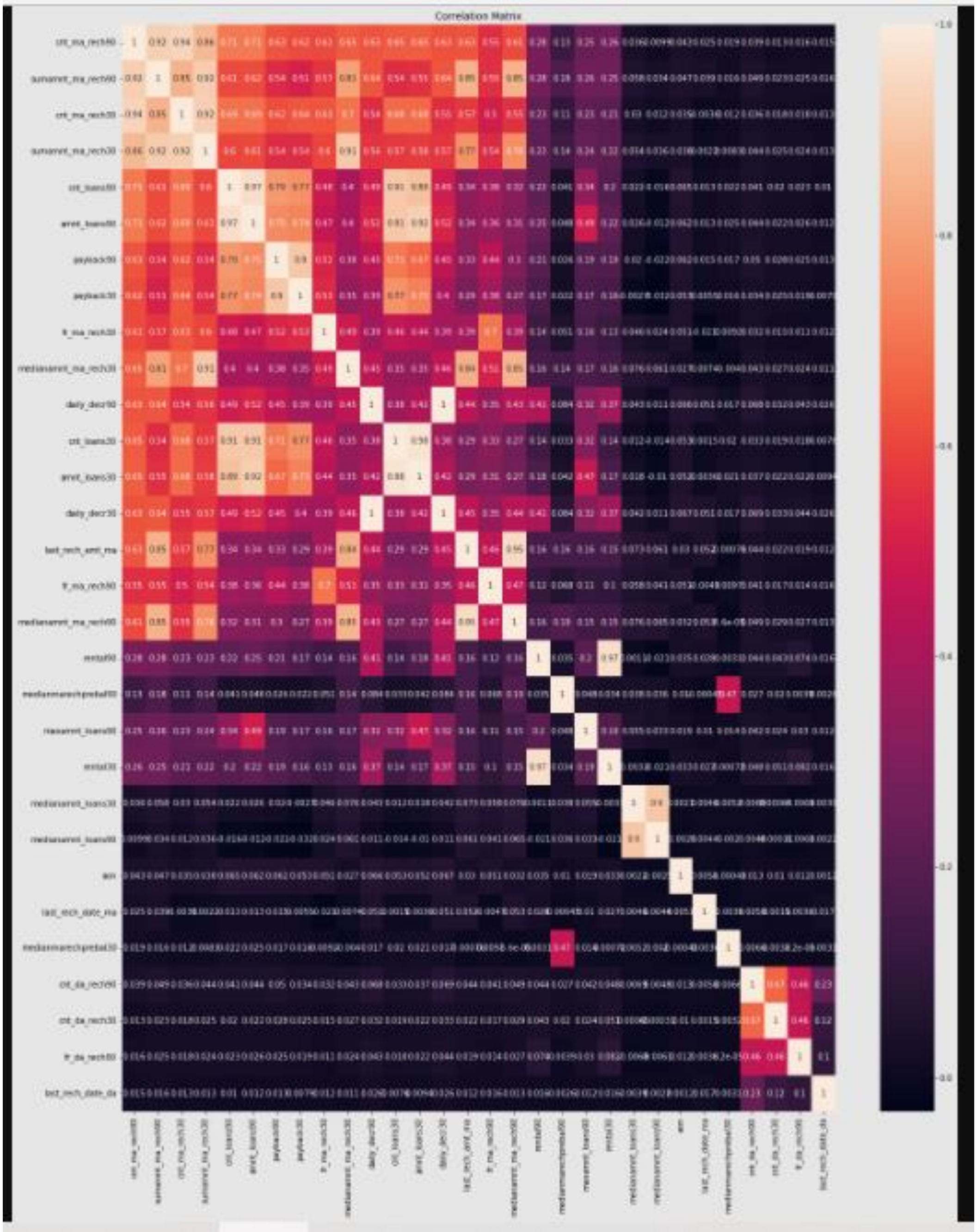
Skewness has been removed.

Checking Multicollinearity:

i) Heat Map:

In [60]:

```
corr_mat=X_New.corr()
plt.figure(figsize=[20,25])
sns.heatmap(corr_mat,annot=True)
plt.title("Correlation Matrix")
plt.show() #Checking correlation
```



From the heat map it can be observed that there is an issue of huge multicollinearity in the data. Therefore checking the issue of multicollinearity via VIF.

ii) VIF:

In [62]:
```
vif_data = pd.DataFrame()
vif_data["feature"] = X_New.columns
vif_data["VIF"] = [variance_inflation_factor(X_New.values, i)
                          for i in range(len(X_New.columns))]

vif_data
```

Out[62]:

	feature	VIF
0	cnt_ma_rech90	95.756029
1	sumamnt_ma_rech90	178.591863
2	cnt_ma_rech30	102.184974
3	sumamnt_ma_rech30	241.438482
4	cnt_loans90	27.196408
5	amnt_loans90	36.677369
6	payback90	7.377498
7	payback30	7.561827
8	fr_ma_rech30	2.731155
9	medianamnt_ma_rech30	58.836042
10	daily_decr90	411.334203
11	cnt_loans30	78.929183
12	amnt_loans30	79.954822
13	daily_decr30	401.635291
14	last_rech_amt_ma	12.628795
15	fr_ma_rech90	2.500932
16	medianamnt_ma_rech90	43.476041
17	rental90	18.915574
18	medianmarechprebal90	1.374190
19	maxamnt_loans90	3.957123
20	rental30	18.011782
21	medianamnt_loans30	5.440365
22	medianamnt_loans90	5.428783
23	aon	1.009602
24	last_rech_date_ma	1.013811
25	medianmarechprebal30	1.298231
26	cnt_da_rech90	1.991707
27	cnt_da_rech30	1.925440
28	fr_da_rech90	1.348058
29	last_rech_date_da	1.061289

Therefore as concluded there is a huge issue of multicollinearity. Therefore few columns have been dropped off based on the following assumptions:

1. Total amount of recharge in the main account for 30 days is included in total amount of recharge for 90 days. Both these features have high correlation hence dropping one .
2. Daily recharge from main account for 30 days is included in Daily recharge from main account for 90 days. Both these features have high correlation hence dropping one .
3. Number of times main account recharge for 30 days is included in Number of times main account recharge for 90 days. Both these features have high correlation hence dropping one .
4. cnt_loans30 and amnt_loans30 are already included in cnt_loans90 and amnt_loans30. Therefore dropping one.

Checking the final VIF Score:

In [68]:
```
vif_data = pd.DataFrame()
vif_data["feature"] = X_New.columns
vif_data["VIF"] = [variance_inflation_factor(X_New.values, i)
                          for i in range(len(X_New.columns))]

vif_data
```

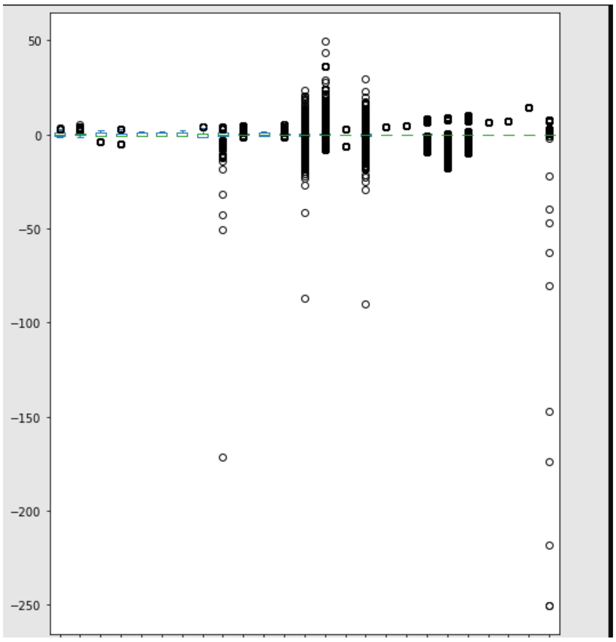
Out[68]:

	feature	VIF
0	cnt_ma_rech90	23.390770
1	sumamnt_ma_rech90	54.154101
2	cnt_loans90	24.048747
3	amnt_loans90	28.428751
4	payback90	6.557669
5	payback30	6.163103
6	fr_ma_rech30	2.562045
7	medianamnt_ma_rech30	4.368223
8	daily_decr90	2.102082
9	last_rech_amt_ma	12.253729
10	fr_ma_rech90	2.299741
11	medianamnt_ma_rech90	21.423719
12	rental90	16.778663
13	medianmarechprebal90	1.373250
14	maxamnt_loans90	2.193553
15	rental30	16.140835
16	medianamnt_loans30	5.419353
17	medianamnt_loans90	5.408884
18	aon	1.009357
19	last_rech_date_ma	1.012901
20	medianmarechprebal30	1.298223
21	cnt_da_rech90	1.991461
22	cnt_da_rech30	1.925207
23	fr_da_rech90	1.347754
24	last_rech_date_da	1.061280

The issue of multicollinearity still remains however due to the issue of data loss, we are going ahead with the present VIF scores.

Detecting Outliers:
Zscore method has been applied to remove the outliers in the data.

```
In [70]:  X_New.plot(kind='box',figsize=(8,10),layout=(4,3))
          plt.show() #checking for oultiers
```



Data has huge outliers and hence this needs to be removed.

Importing Z Score:

```
In [72]:  from scipy.stats import zscore
```

Checking those values which has got zscore less than 3:

```
Out[73]: cnt_ma_rech90             False
         sumamnt_ma_rech90         False
         cnt_loans90                True
         amnt_loans90              False
         payback90                  True
         payback30                  True
         fr_ma_rech30               True
         medianamnt_ma_rech30      False
         daily_decr90              False
         last_rech_amt_ma          False
         fr_ma_rech90               True
         medianamnt_ma_rech90      False
         rental90                  False
         medianmarechprebal90      False
         maxamnt_loans90            True
         rental30                  False
         medianamnt_loans30        False
         medianamnt_loans90        False
         aon                       False
         last_rech_date_ma         False
         medianmarechprebal30      False
         cnt_da_rech90             False
         cnt_da_rech30             False
         fr_da_rech90              False
         last_rech_date_da         False
         dtype: bool
```

Assigning a variable to the values having less than 3 zscore

```
In [75]:
# assigning a variable to the values having less than 3 zscore
X_new = X_New[(np.abs(zscore(X_New))<3).all(axis=1)]
X_new
```

Dropping off the outlier values from Target variables:

```
In [77]:
Y_new=Y.drop(index[0],axis=0)
Y_new  #removing the outliers from target variables

Out[77]: 0         1
         1         1
         2         1
         3         1
         4         1
                  ..
         366857    0
         366858    0
         366859    0
         366860    0
         366861    0
         Name: label, Length: 310691, dtype: int64
```

Scaling the data:

```
In [79]:  1 Scalar=StandardScaler() #scaling the data

In [80]:  1 X_Scaled=Scalar.fit_transform(X_new)
          2 X_Scaled

Out[80]: array([[-0.51182453,  0.4240501 , -0.85741498, ...,  1.        ,
                 -1.        , -0.02843201],
                [-0.51182453, -0.15917618, -0.85741498, ...,  1.        ,
                 -1.        , -0.02843201],
                [-0.51182453, -0.33452936,  0.2382696 , ...,  1.        ,
                 -1.        , -0.02843201],
                ...,
                [-0.0425187 ,  0.12524568, -0.85741498, ...,  1.        ,
                 -1.        , -0.02843201],
                [-0.0425187 ,  0.78292484,  0.2382696 , ...,  1.        ,
                 -1.        , -0.02843201],
                [-0.51182453, -0.15917618, -0.85741498, ...,  1.        ,
                 -1.        , -0.02843201]])
```

Model/s Development and Evaluation

(Linear Regression, Decision Tree Regressor, Random Forest Regressor and Gradient Boosting Regressor)

Logistic Regression: 77.62%

In [81]:
```
LR=LogisticRegression()
```

In [82]:
```
X_train,X_test,y_train,y_test=train_test_split(X_Scaled,Y_new,test_size=0.20,random_state=58)
LR.fit(X_train,y_train)
pred_test=LR.predict(X_test) # Testing the prediction of test data
```

In [83]:
```
LR_accuracy=accuracy_score(y_test,pred_test)*100
LR_accuracy
```

Out[83]:
```
77.6211397029241
```

In [84]:
```
Conf_Mat=confusion_matrix(y_test,pred_test)
Conf_Mat
```

Out[84]:
```
array([[25523,  6697],
       [ 7209, 22710]], dtype=int64)
```

Decision Tree Regressor: 95.56%

Decision Tree Classifier

In [85]:
```
DT=DecisionTreeClassifier()
```

In [86]:
```
X_train,X_test,y_train,y_test=train_test_split(X_Scaled,Y_new,test_size=0.20,random_state=58)
DT.fit(X_train,y_train)
pred_test=DT.predict(X_test) # Testing the prediction of test data
```

In [87]:
```
DT_accuracy=accuracy_score(y_test,pred_test)*100
DT_accuracy
```

Out[87]:
```
95.56800077246173
```

In [88]:
```
Conf_Mat=confusion_matrix(y_test,pred_test)
Conf_Mat
```

Out[88]:
```
array([[32187,    33],
       [ 2721, 27198]], dtype=int64)
```

Random Forest Regressor: 97.85%

In [89]:
```
RF=RandomForestClassifier()
```

In [90]:
```
X_train,X_test,y_train,y_test=train_test_split(X_Scaled,Y_new,test_size=0.20,random_state=58)
RF.fit(X_train,y_train)
pred_test=RF.predict(X_test) # Testing the prediction of test data
```

In [91]:
```
RF_accuracy=accuracy_score(y_test,pred_test)*100
RF_accuracy
```

Out[91]:
```
97.85320008368336
```

In [92]:
```
Conf_Mat=confusion_matrix(y_test,pred_test)
Conf_Mat
```

Out[92]:
```
array([[32185,    35],
       [ 1299, 28620]], dtype=int64)
```

Gradient Boosting Regressor:80.65%

In [93]:
```
GB=GradientBoostingClassifier()
```

In [94]:
```
X_train,X_test,y_train,y_test=train_test_split(X_Scaled,Y_new,test_size=0.20,random_state=58)
GB.fit(X_train,y_train)
pred_test=GB.predict(X_test) # Testing the prediction of test data
```

In [95]:
```
GB_accuracy=accuracy_score(y_test,pred_test)*100
GB_accuracy
```

Out[95]:
```
80.65144273322711
```

In [96]:
```
Conf_Mat=confusion_matrix(y_test,pred_test)
Conf_Mat
```

Out[96]:
```
array([[26631,  5589],
       [ 6434, 23485]], dtype=int64)
```

After getting the r accuracy, hence cross validation technique has been used.

Cross Validation for LR:

Cross Validation for LR

In [97]:
```
LR_Val=cross_val_score(LR,X_Scaled,Y_new,cv=5)
print("The cross validation score is",RF_Val.mean())
```

```
The cross validation score for 2 is 0.7755165084891841
The cross validation score for 3 is 0.7753748913094468
The cross validation score for 4 is 0.7754489214078257
The cross validation score for 5 is 0.7754778869284764
```

Cross Validation for DT:

Cross Validation for DT

In [98]:
```
DT_Val=cross_val_score(DT,X_Scaled,Y_new,cv=5)
print("The cross validation score is",DT_Val.mean())
```

```
The cross validation score for 2 is 0.9329494568676597
The cross validation score for 3 is 0.9478452832003352
The cross validation score for 4 is 0.9526925483320713
The cross validation score for 5 is 0.9549938646918494
```

Cross Validation for RF:

Cross Validation for RF

In [99]:
```
RF_Val=cross_val_score(RF,X_Scaled,Y_new,cv=3)
print("The cross validation score is",RF_Val.mean())
```

```
The cross validation score is 0.9779716817597965
```

Cross Validation for GB:

Cross Validation for GB

In [100]:
```
GB_Val=cross_val_score(GB,X_Scaled,Y_new,cv=3)
print("The cross validation score is",GB_Val.mean())
```

```
The cross validation score is 0.8072393467557414
```

As none of the models are overfitted, and based on the accuracy score and cross validation scores, Random Forest Classifier model is best for this dataset.

Conclusion

Key Findings:

This is a dataset which is filled with outliers however following points can be concluded:

a) A customer is ready to spent around 2,65,926 amount in a month for their recharge also there are people who have not recharged their account over a month.
b) In a month an account has been recharged over 200 times as well it has not been recharged once in a month.
c) The highest amount recharged in a period of 90 days is 9,00,000.

The top 10 points are more important to check whether a customer can successfully pay back the loan amount:

1. ***Number of times main account got recharged in last 90 days***
2. ***Total amount of recharge in main account over last 90 days***
3. ***Number of times main account got recharged in last 30 days***
4. ***Total amount of recharge in main account over last 30 days***
5. ***Number of loans taken by user in last 90 days***
6. ***Total amount of loans taken by user in last 90 days***
7. ***Average payback time in days over last 90 days***
8. ***Average payback time in days over last 30 days***
9. ***Frequency of main account recharged in last 30 days***
10. ***Median of amount of recharges done in main account over last 30 days at user level***