NAME OF THE PROJECT

Micro Credit Defaulter Project

Submitted by:

SHANTANU

ACKNOWLEDGMENT

This dataset of micro credit analysis has been provided to us from a client that is in telecom industry. They are a fixed wireless telecommunications network provider. They have launched various products and have developed its business and organization based on the budget operator model, offering better products at Lower Prices to all value conscious customers through a strategy of disruptive innovation that focuses on the subscriber.

INTRODUCTION

➢ Business Problem Framing:

It is a project related to Telecom Industry. They have collaborated with Microfinance Institution (MFI) that offers financial services to low income populations. Micro Finance Service (MFS) become very useful when we are targeting unbanked poor families living in remote areas with not much sources of income.
The client is in telecom industry and they are a fixed wireless telecommunications network provider and they understand the importance of communication and how it affects a person's life, thus focusing on providing their services and products to low income families and customers that can help them in the need of hour. 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 .We have to build a prediction model which will tell us whether the person will become defaulter or not thus helping company in giving credit . This would help client company in further investment and improvement in selection of customers.

➢ Review of Literature:

In this model we will study different variables and how this independent variables are related with dependent variables and how this will help us to predict whether the customer will become defaulter or not using different machine learning model and thus selecting the final model that giving us best score.

➢ Motivation for the Problem Undertaken

In today's modern world scenario communication has become the backbone of every individual. The initiative of helping low income families by proving then micro credit loans for communication has been proved very beneficial to themand building a prediction model for the company which will help them to predict whether loan provided to customer will become defaulter or not , this will help company in future weather and in which condition he should provide the customers micro credit loan.
.

Analytical Problem Framing

➢ Mathematical/ Analytical Modelling of the Problem:

Lets view some basic statistics about the data like the percentile, mean , maximum, minimum etc.

```
df.describe()
```

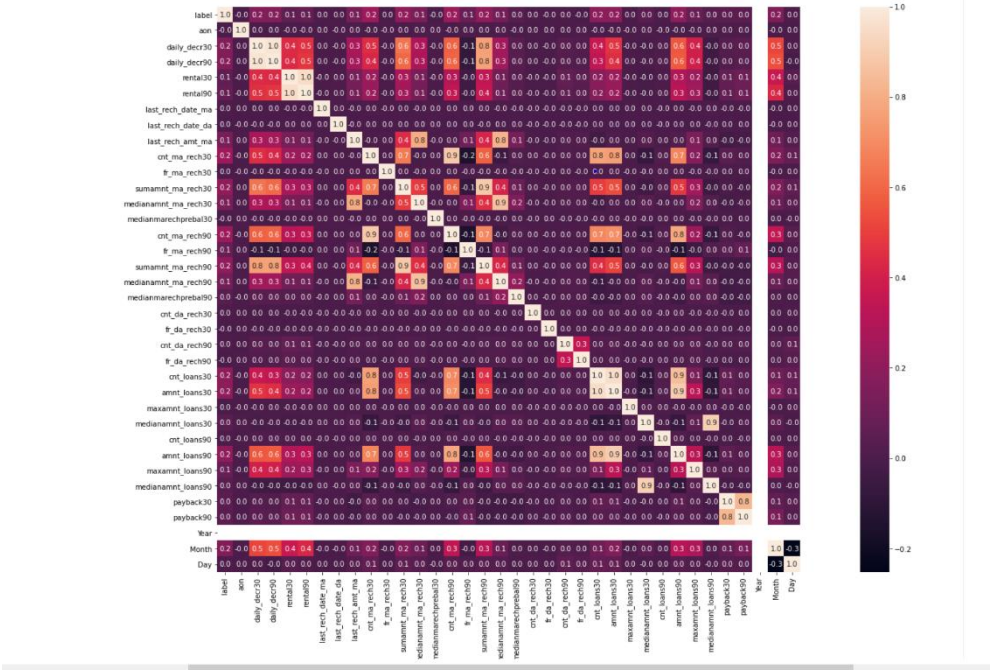
	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_n
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000
mean	0.875177	8112.343445	5381.402289	6082.515068	2692.581910	3483.406534	3755.847800	3712.202921	2064.452797
std	0.330519	75696.082531	9220.623400	10918.812767	4308.586781	5770.461279	53905.892230	53374.833430	2370.786034
min	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000	-29.000000	0.000000
25%	1.000000	246.000000	42.440000	42.692000	280.420000	300.260000	1.000000	0.000000	770.000000
50%	1.000000	527.000000	1469.175667	1500.000000	1083.570000	1334.000000	3.000000	0.000000	1539.000000
75%	1.000000	982.000000	7244.000000	7802.790000	3356.940000	4201.790000	7.000000	0.000000	2309.000000
max	1.000000	999860.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733	999171.809410	55000.000000

cnt_ma_rech30	fr_ma_rech30	sumamnt_ma_rech30	medianamnt_ma_rech30	medianmarechprebal30	cnt_ma_rech90	fr_ma_rech90	sumamnt_ma_rech9
209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.00000	209593.000000	209593.000000
3.978057	3737.355121	7704.501157	1812.817952	3851.927942	6.31543	7.716780	12396.218352
4.256090	53643.625172	10139.621714	2070.864620	54006.374433	7.19347	12.590251	16857.793882
0.000000	0.000000	0.000000	0.000000	-200.000000	0.00000	0.000000	0.000000
1.000000	0.000000	1540.000000	770.000000	11.000000	2.00000	0.000000	2317.000000
3.000000	2.000000	4628.000000	1539.000000	33.900000	4.00000	2.000000	7226.000000
5.000000	6.000000	10010.000000	1924.000000	83.000000	8.00000	8.000000	16000.000000
203.000000	999606.368132	810096.000000	55000.000000	999479.419319	336.00000	88.000000	953036.000000

- There are 209593 distinct micro-credit customers.
- The average value for Number of days till last recharge of main account is 3755.84. The standard deviation is unusually large, max value being 998650.37.
- The average value for Number of days till last recharge of data account is 3712.20. The standard deviation is unusually large, max value being 999171.80.
- The average value for Number of times main account got recharge in last 30 days is 3.97 and the max value of recharge is 203.
- The average value for number of times data account got recharge in last 30 days is 262.57. The standard deviation is high ,a max value being 99914.44
- The average value for number of loans taken by user in last 30 days is 2.75 and std is 2.55 , max value is 50.

Lets See co-relation between the Columns

```
In [49]: #Observing coreLation between the columns through heatmap
         plt.figure(figsize=(30,15))
         sns.heatmap(df.corr(),annot=True,square=True,fmt=".1f")
         plt.show

Out[49]: <function matplotlib.pyplot.show(*args, **kw)>
```



- From the above observation we can see that aon , medianmarechprebal30 , fr_da_rech30 , fr_da_rech90 are negatively co-related and rest are positively co-related with label**.**

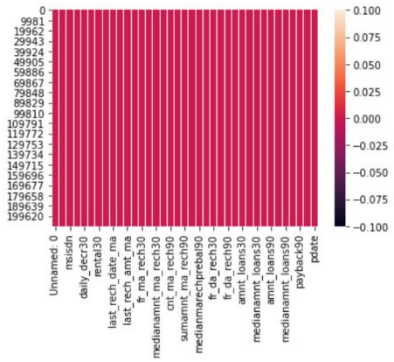
Data Sources and their formats:

We have two excel data file one has the details of all user and their different recharges and loan taken and whether they had paid back loan or not and other file contain details of the data .

Data Pre-processing Done:

- Let's check the shape and see count of the number of empty values in each column.

```
In [32]: # Checking the shape of data set
         df.shape

Out[32]: (209593, 37)

In [33]: #checking null values using heat map
         sns.heatmap(df.isnull())

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x1eb8fff7d30>
```




```
In [34]: df.isnull().sum()

Out[34]: Unnamed: 0              0
         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
```

- As we can see from above Dataset contains 209593 rows and 37 columns in whichlabel is the dependent target column and rest are independent columns .
- And we can see dataset contains no null values.

```
In [35]: df["Year"] = pd.to_datetime(df.pdate, format="%Y-%m-%d").dt.year
         df["Month"] = pd.to_datetime(df.pdate, format="%Y-%m-%d").dt.month
         df["Day"] = pd.to_datetime(df.pdate, format="%Y-%m-%d").dt.day

In [36]: df['Year'].value_counts()

Out[36]: 2016    209593
         Name: Year, dtype: int64

In [37]: df.drop(['pdate','Unnamed: 0'],axis=1,inplace=True)

In [38]: df.head()
```

Out[38]:

	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
0	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	2	21.0
1	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	5787	1	0.0
2	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539	1	0.0
3	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	947	0	0.0
4	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309	7	2.0

```
In [39]: # Checking Unique values of Attributes
         for col in df:
             print(col)
             print(df[col].unique())
             print('')

label
[0 1]

msisdn
['21408I70789' '76462I70374' '17943I70372' ... '22758I85348' '59712I82733'
 '65061I85339']

aon
[2.72000000e+02 7.12000000e+02 5.35000000e+02 ... 8.03380622e+05
 5.81435484e+05 8.11881373e+05]

daily_decr30
[ 3055.05       12122.          1398.         ... 11843.11166667
 12488.22833333  4489.362     ]

daily_decr90
[ 3065.15       12124.75        1398.         ...   151.87233333
 12574.37        4534.82      ]

rental30
[ 220.13 3691.26  900.13 ... 5861.83  411.83  483.92]
```

- Data set contains pdate in format of year, month and date. We will split the pdate column for further analysis.
- After checking the unique values of each column we can see that year count is only one so we will drop pdate, year and unnamed as it is of no use.

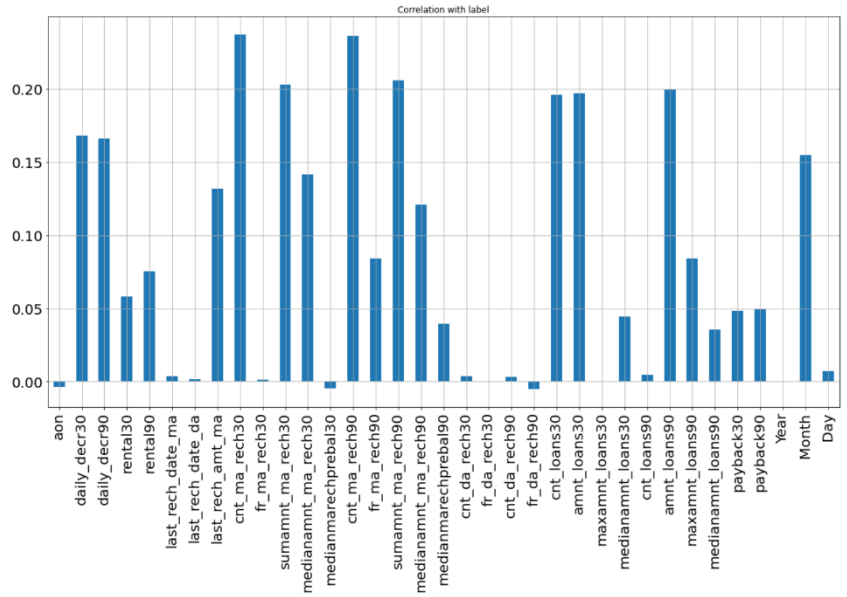
```
]: print("The shape before removing outliers and skewness",df.shape)
   print("skewness before removing outliers")
   print(df.skew())
   from scipy.stats import zscore
   z=np.abs(zscore(df))
   df1=df[(z<3).all(axis=1)]
   print("new shape after removing outliers",df1.shape)
   print("skewness after removing outliers")
   print(df1.skew())

The shape before removing outliers and skewness (209593, 33)
skewness before removing outliers
label                    -2.270254
aon                      10.392949
daily_decr30              3.946230
daily_decr90              4.252565
rental30                  4.521929
rental90                  4.437681
last_rech_date_ma        14.790974
last_rech_date_da        14.814857
last_rech_amt_ma          3.781149
cnt_ma_rech30             3.283842
fr_ma_rech30             14.772833
sumamnt_ma_rech30         6.386787
medianamnt_ma_rech30      3.512324
medianmarechprebal30     14.779875
cnt_ma_rech90             3.425254
fr_ma_rech90              2.285423
sumamnt_ma_rech90         4.897950
medianamnt_ma_rech90      3.752706
medianmarechprebal90     44.880503
cnt_da_rech30            17.818364
cnt_da_rech90            27.267278
fr_da_rech90             28.988083
cnt_loans30               2.713421
amnt_loans30              2.975719
medianamnt_loans30        4.551043
cnt_loans90              16.594408
amnt_loans90              3.150006
maxamnt_loans90           1.678304
medianamnt_loans90        4.895720
payback30                 8.310695
payback90                 6.899951
Month                     0.343242
Day                       0.199845
dtype: float64
new shape after removing outliers (163026, 33)
skewness after removing outliers
label                    -2.090282
aon                       0.958194
daily_decr30              1.963119
daily_decr90              2.077247
rental30                  2.195563
rental90                  2.244957
last_rech_date_ma         3.098605
last_rech_date_da        10.390692
last_rech_amt_ma          2.125025
cnt_ma_rech30             1.174958
```

- Data set contains many outliers so by using zscore we will remove outliers from the data set

➢ Data Inputs- Logic- Output Relationships:

```
X = df.drop(['label'],axis=1)
X.corrwith(df['label']).plot.bar(figsize = (20, 10), title = "Correlation with label", fontsize = 20,rot = 90, grid = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1eba1b73220>
```



It seems from the above graph is that negatively correlated feature is age on cellular network in days, medianmarechprebal30, but we cannot blindly remove this feature because according to me it is very important feature for prediction. msisdn, year, pcircle and Frequency of data account recharged in last 30 days is unimportant and it has no role in prediction so we will remove it later.

➢ Hardware and Software Requirements and Tools Used:

We will use here Jupyter notebook to make Prediction Model.

Model/s Development and Evaluation

➢ Identification of possible problem-solving approaches (methods):

```
In [56]: x=df1.drop('label',axis=1)
         y=df1['label']

In [58]: x
```

Out[58]:

	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	sur
0	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	2	21.0	307
1	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	5787	1	0.0	578
2	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539	1	0.0	153
3	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	947	0	0.0	0.0
4	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309	7	2.0	200
...	...	...	...	...	...	...	...	...	...	...	...
209588	404.0	151.872333	151.872333	1089.19	1089.19	1.0	0.0	4048	3	2.0	104
209589	1075.0	36.936000	36.936000	1728.36	1728.36	4.0	0.0	773	4	1.0	309
209590	1013.0	11843.111667	11904.350000	5861.83	8893.20	3.0	0.0	1539	5	8.0	933
209591	1732.0	12488.228333	12574.370000	411.83	984.58	2.0	38.0	773	5	4.0	121
209592	1581.0	4489.362000	4534.820000	483.92	631.20	13.0	0.0	7526	2	1.0	908

163026 rows × 32 columns

```
In [59]: y
```

Out[59]:

```
0         0
1         1
2         1
3         1
4         1
         ..
209588    1
209589    1
209590    1
209591    1
209592    1
Name: label, Length: 163026, dtype: int64
```

- Now we will split the data set into input and output variable. As you can see above x is your input variable and y (label) is your target out variable.

Let's check feature importance of the Data set.

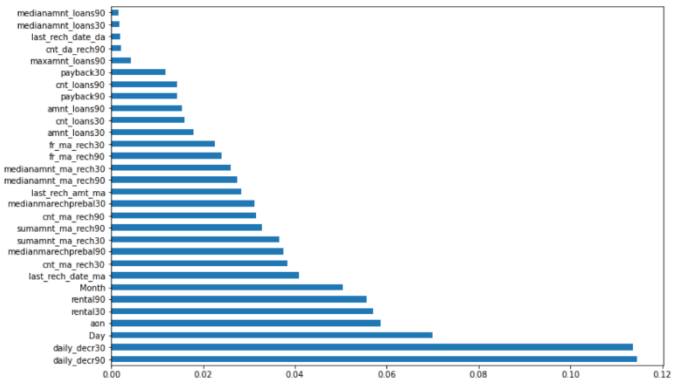
- You can get the feature importance of each feature of your dataset by using the feature importance property of the model.
- Feature importance gives you a score for each feature of your data, the higher the score more important or relevant is the feature towards your output variable.
- Feature importance is an inbuilt class that comes with Tree Based Classifiers, we will be using Extra Tree Classifier for extracting the top 10 features for the dataset

```
In [60]: from sklearn.ensemble import ExtraTreesClassifier
         selection = ExtraTreesClassifier()
         selection.fit(x,y)

Out[60]: ExtraTreesClassifier()

In [61]: print(selection.feature_importances_) #use inbuilt class feature_importances of tree based classifiers
```

```
[0.05865932 0.11359513 0.11445125 0.05708914 0.05557272 0.04090734
 0.00198992 0.02834435 0.03836222 0.02268823 0.03662922 0.02594971
 0.03116649 0.03162234 0.02412374 0.03277798 0.02751454 0.0374299
 0.00118876 0.0022237  0.         0.01597931 0.01791491 0.00177666
 0.01439392 0.01534676 0.00420581 0.00151699 0.01180189 0.01442742
 0.05037766 0.06997265]
```

Checking Feature Importance

```
In [72]: plt.figure(figsize = (12,8))
         feat_importances = pd.Series(selection.feature_importances_, index=x.columns)
         feat_importances.nlargest(30).plot(kind='barh')
         plt.show()
```



- From the above analysis we can see that Daily_decr90, daily_dec30 are the month importantfeature for model valuation and medianamnt_loans90,medianamnt_loans30 are less important .

➢ Testing of Identified Approaches (Algorithms)

Importing Necessary Libraries

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score,f1_score, precision_score, recall_score, confusion_matrix,classification_report,r
from sklearn.model_selection import GridSearchCV
```

1. **Logistic Regression:**

- In Logistic Regression, we wish to model a dependent variable(y) in terms of one or more independent variables(x). It is a method for classification. This algorithm is used for the dependent variable that is Categorical. Y is modelled using a function that gives output between 0 and 1 for all values of X. In Logistic Regression, the Sigmoid (aka Logistic) Function is used

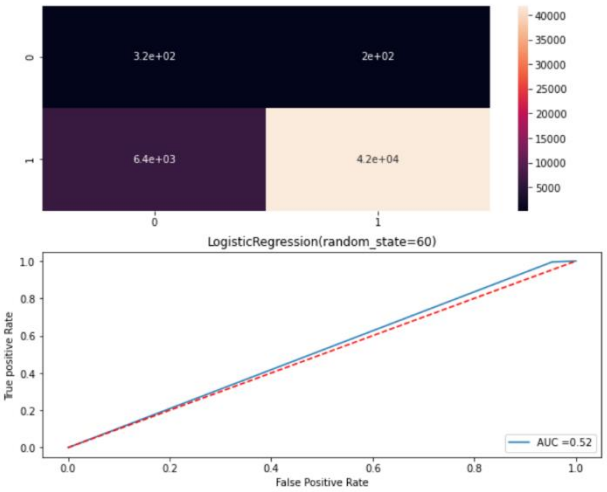
```
max_acc_score=0
for i in range(42,101):
    r_state=i
    train_x,test_x,train_y,test_y=train_test_split(x,y,random_state=r_state ,test_size=0.30)
    lg=LogisticRegression()
    lg.fit(train_x,train_y)
    pred=lg.predict(test_x)
    acc_score=accuracy_score(test_y,pred)
    if acc_score> max_acc_score:
            max_acc_score=acc_score
            final_r_state=r_state
print("The max accuracy score  for LogisticRegression  ", max_acc_score,"is achieved at" ,final_r_state)
print("\n")
```

Further we will use grid search cv to find the best parameter for logistic regression.

```
]: lg=LogisticRegression()
   parameters={"penalty":['l1' ,'l2']  }
   gd=GridSearchCV(lg,parameters)
   gd.fit(train_x,train_y)
   print(gd.best_params_)
   print("\n")
```

Results:

```
The model calculation for LogisticRegression(random_state=60) are:
[1 1 1 ... 1 1 1]
Accuracy Score=  0.8642144434448352
The CV Score is 0.8634819026734579

[[  315   203]
 [ 6438 41952]]

              precision    recall  f1-score   support

           0       0.05      0.61      0.09       518
           1       1.00      0.87      0.93     48390

    accuracy                           0.86     48908
   macro avg       0.52      0.74      0.51     48908
weighted avg       0.99      0.86      0.92     48908


F1 Score=  0.9266552542934452

Precision Score=  0.9951844383821611

Recall Score=  0.8669559826410416

roc_auc_score 0.5209151867610494


AxesSubplot(0.125,0.808774;0.62x0.0712264)
```



2.Decision Tree Classification:

- The idea of a decision tree is to divide the data set into smaller data sets based on the descriptive features until you reach a small enough set that contains data points that fall under one label.
- Decision trees are easy to interpret. To build a decision tree requires little data preparation from the user- there is no need to normalize data.

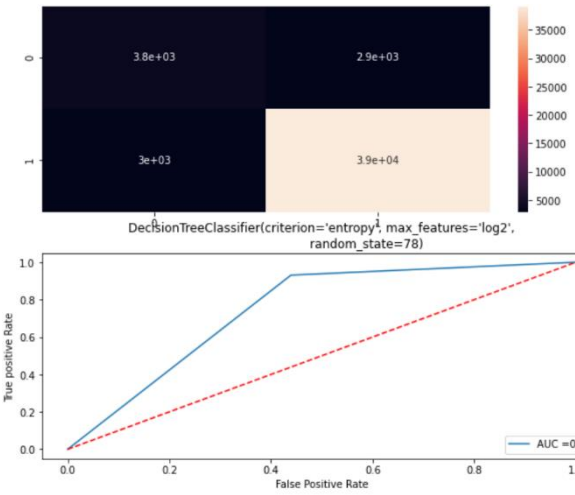
```
max_acc_score=0
for i in range(42,101):
    r_state=i
    train_x,test_x,train_y,test_y=train_test_split(x,y,random_state=r_state ,test_size=0.30)
    dtc=DecisionTreeClassifier()
    dtc.fit(train_x,train_y)
    pred=dtc.predict(test_x)
    acc_score=accuracy_score(test_y,pred)
    if acc_score> max_acc_score:
            max_acc_score=acc_score
            final_r_state=r_state
print("The max accuracy score  for DecisionTreeClassifier ", max_acc_score,"is achieved at" ,final_r_state)
print("\n")
```

Further we will use grid search cv to find the best parameter for Decision Tree Classifier.

```
#Best parameters for DecisionTreeClassifier
dtc=DecisionTreeClassifier()
parameters={"criterion" :("gini" ,"entropy") ,'max_features' : ['auto', 'sqrt', 'log2']}
gd=GridSearchCV(dtc,parameters)
gd.fit(train_x,train_y)
print(gd.best_params_)
print("\n")
```

Results:

```
The model calculation for DecisionTreeClassifier(criterion='entropy', max_features='log2',
                       random_state=78) are:
[0 1 0 ... 1 0 1]
Accuracy Score=  0.8798151631634906
The CV Score is 0.8755106492678056

[[ 3786  2911]
 [ 2967 39244]]

              precision    recall  f1-score   support

           0       0.56      0.57      0.56      6697
           1       0.93      0.93      0.93     42211

    accuracy                           0.88     48908
   macro avg       0.75      0.75      0.75     48908
weighted avg       0.88      0.88      0.88     48908


F1 Score=  0.9303273830690089

Precision Score=  0.930945320839758

Recall Score=  0.9297102650967757

roc_auc_score 0.7457925182608386

AxesSubplot(0.125,0.808774;0.62x0.0712264)
```



3.Random Forest Classification:

Random Forest is a supervised learning algorithm, it creates a forest and makes it somehow random. The "forest“ it builds, is an ensemble of Decision Trees.

Step-1Pick at random K data points from the training set.

Step-2 Build the Decision tree associated to these K data points

Step-3Choose the Number of trees(n) you want to build and repeat Step1 and Step2

Step-4For a new data points make each one of your 'n' trees predict the category to which the data point belongs and assign the new data point to the category that wins the majority vote.

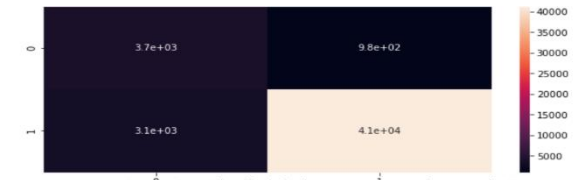
```
max_acc_score=0
for i in range(42,101):
    r_state=i
    train_x,test_x,train_y,test_y=train_test_split(x,y,random_state=r_state ,test_size=0.30)
    rfc=RandomForestClassifier()
    rfc.fit(train_x,train_y)
    pred=rfc.predict(test_x)
    acc_score=accuracy_score(test_y,pred)
    if acc_score> max_acc_score:
            max_acc_score=acc_score
            final_r_state=r_state
print("The max accuracy score  for RandomForestClassifier ", max_acc_score,"is achieved at" ,final_r_state)
print("\n")
```

Further we will use grid search cv to find the best parameter for Random Forest Classifier.

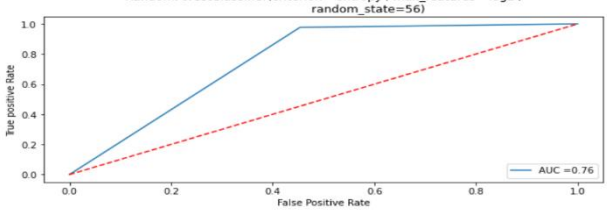
```
#Best parameters for  RandomForestClassifier
rfc=RandomForestClassifier()
parameters={"criterion" :("gini" ,"entropy") , 'max_features' : ['auto', 'sqrt', 'log2']}
gd=GridSearchCV(rfc,parameters)
gd.fit(train_x,train_y)
print(gd.best_params_)
print("\n")
```

Results:

```
The model calculation for RandomForestClassifier(criterion='entropy', max_features='log2',
                       random_state=56) are:
[1 1 0 ... 1 0 1]
Accuracy Score=  0.9173550339412775
The CV Score is 0.9168046779501926

[[ 3690   979]
 [ 3063 41176]]

              precision    recall  f1-score   support

           0       0.55      0.79      0.65      4669
           1       0.98      0.93      0.95     44239

    accuracy                           0.92     48908
   macro avg       0.76      0.86      0.80     48908
weighted avg       0.94      0.92      0.92     48908


F1 Score=  0.9532143435886752

Precision Score=  0.9767761831336733

Recall Score=  0.9307624494224553

roc_auc_score 0.7615999973864723

AxesSubplot(0.125,0.808774;0.62x0.0712264)
```



RandomForestClassifier(criterion='entropy', max_features='log2',
random_state=56)



4.Gradient Boosting:

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

```
max_acc_score=0
for i in range(42,101):
    r_state=i
    train_x,test_x,train_y,test_y=train_test_split(x,y,random_state=r_state ,test_size=0.30)
    gbr=GradientBoostingClassifier()
    gbr.fit(train_x,train_y)
    pred=gbr.predict(test_x)
    acc_score=accuracy_score(test_y,pred)
    if acc_score> max_acc_score:
            max_acc_score=acc_score
            final_r_state=r_state
print("The max accuracy score  for GradientBoostingClassifier", max_acc_score,"is achieved at" ,final_r_state)
print("\n")
```

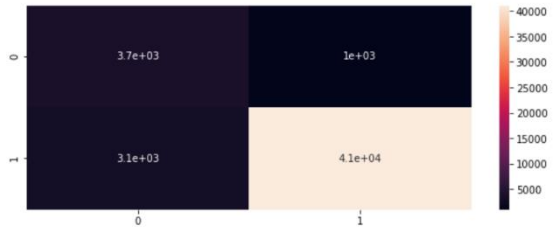
Further we will use grid search cv to find the best parameter for Gradient Boosting Classifier.

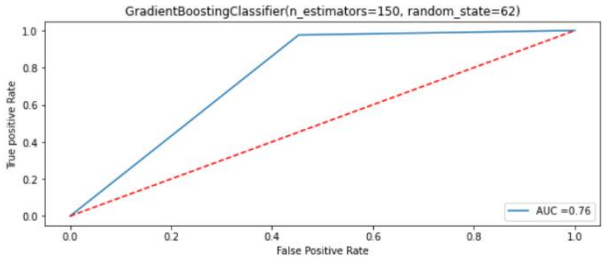
```
gbc=GradientBoostingClassifier()
parameters={"learning_rate":[0.001,0.01,0.1,1],"n_estimators":[10,50,100,120,150]}
gd=GridSearchCV(gbc,parameters)
gd.fit(train_x,train_y)
print(gd.best_params_)
print("\n")
```

```
{'penalty': 'l2'}
```

Results:

```
The model calculation for GradientBoostingClassifier(n_estimators=150, random_state=62) are:
[1 1 0 ... 1 0 1]
Accuracy Score=  0.9166598511490963
The CV Score is 0.9158784387025557

[[ 3695  1018]
 [ 3058 41137]]

              precision    recall  f1-score   support

           0       0.55      0.78      0.64      4713
           1       0.98      0.93      0.95     44195

    accuracy                           0.92     48908
   macro avg       0.76      0.86      0.80     48908
weighted avg       0.93      0.92      0.92     48908


F1 Score=  0.9527967573827446

Precision Score=  0.9758510259755664

Recall Score=  0.9308066523362372

roc_auc_score 0.7615076246418628

AxesSubplot(0.125,0.808774;0.62x0.0712264)
```





5.**K Neighbors Classifier**:

This is a supervised, non-parametric learning algorithm which classifies a given point based on its neighbours. The choice of the ‘k’ becomes very crucial since the data point is assigned to the class of the nearest ‘k’ neighbors. Once we get to know such ‘k’ nearest data points, the test data is assigned a label by taking the majority vote from the class labels of the ‘k’ nearest data points

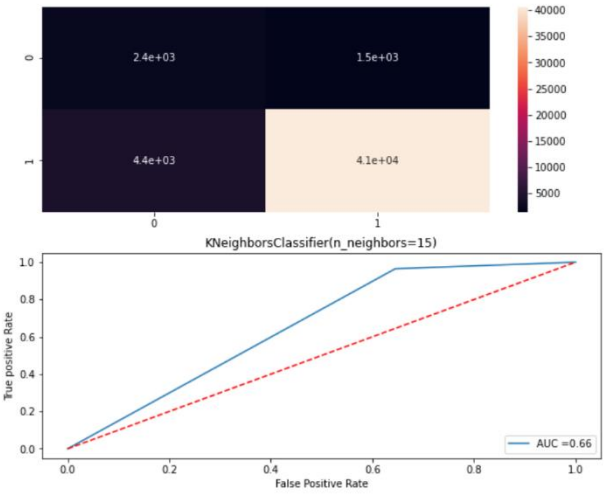
```
max_acc_score=0
for i in range(42,101):
    r_state=i
    train_x,test_x,train_y,test_y=train_test_split(x,y,random_state=r_state ,test_size=0.30)
    knn=KNeighborsClassifier()
    knn.fit(train_x,train_y)
    pred=knn.predict(test_x)
    acc_score=accuracy_score(test_y,pred)
    if acc_score> max_acc_score:
            max_acc_score=acc_score
            final_r_state=r_state
print("The max accuracy score  for KNeighborsClassifier", max_acc_score,"is achieved at" ,final_r_state)
print("\n")
```

Further we will use grid search cv to find the best parameter for K NeighborsClassifier.

```
#Best parameters for KNeighborsClassifier
knn=KNeighborsClassifier()
parameters={"n_neighbors" :(5,10,15) , 'algorithm' :['auto', 'ball_tree', 'kd_tree', 'brute']}
gd=GridSearchCV(knn,parameters)
gd.fit(train_x,train_y)
print(gd.best_params_)
print("\n")
```

Results:

```
The model calculation for KNeighborsClassifier(n_neighbors=15) are:
[1 1 1 ... 1 0 1]
Accuracy Score=  0.8805716856138055
The CV Score is 0.8804853123493848

[[ 2397  1485]
 [ 4356 40670]]

              precision    recall  f1-score   support

           0       0.35      0.62      0.45      3882
           1       0.96      0.90      0.93     45026

    accuracy                           0.88     48908
   macro avg       0.66      0.76      0.69     48908
weighted avg       0.92      0.88      0.89     48908


F1 Score=  0.9330014567394271

Precision Score=  0.9647728620566955

Recall Score=  0.9032558965930796

roc_auc_score 0.6598631080607777

AxesSubplot(0.125,0.808774;0.62x0.0712264)
```



➢ Key Metrics for success in solving problem under consideration

Accuracy Score is the number of correct predictions made as a ratio of all predictions made. It is the most common evaluation metric for classification problems.

Cross-validation is to call the cross_val_score helper function on the estimator and the dataset.

To estimate the accuracy of a linear kernel support vector machine on the dataset by splitting the data, fitting a model and computing the score (n=5 or any number provided by you)consecutive times (with different splits each time):

The **Area Under the Curve (AUC)** is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes
Receiver Operating Characteristic(ROC) summarizes the model's performance by evaluating the trade offs between true positive rate (sensitivity) and false positive rate(1- specificity). For plotting ROC, it is advisable to assume p > 0.5 since we are more concerned about success rate.

ROC summarizes the predictive power for all possible values of p > 0.5. The area under curve (AUC), referred to as index of accuracy(A) or concordance index, is a perfect performance metric for ROC curve. Higher the area under curve, better the prediction power of the model.

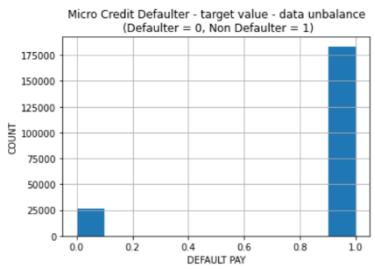
F1-score is a measure of a test's accuracy. It is calculated from the precision and recall of the test, where the precision is the number of correctly identified positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of correctly identified positive results divided by the number of all samples that should have been identified as positive.

The F1 score is the harmonic mean of the precision and recall.

➢ Visualizations

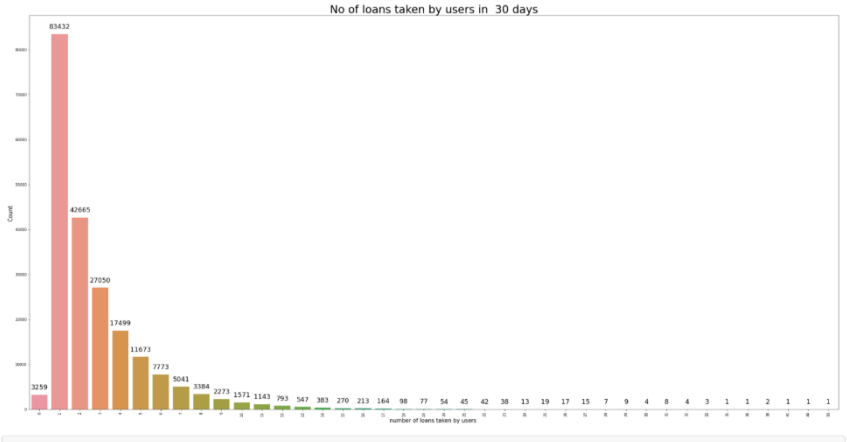
```
df['label'].hist()
plt.xlabel('DEFAULT PAY')
plt.ylabel('COUNT')
plt.title('Micro Credit Defaulter - target value - data unbalance\n (Defaulter = 0, Non Defaulter = 1)')
```

```
Text(0.5, 1.0, 'Micro Credit Defaulter - target value - data unbalance\n (Defaulter = 0, Non Defaulter = 1)')
```



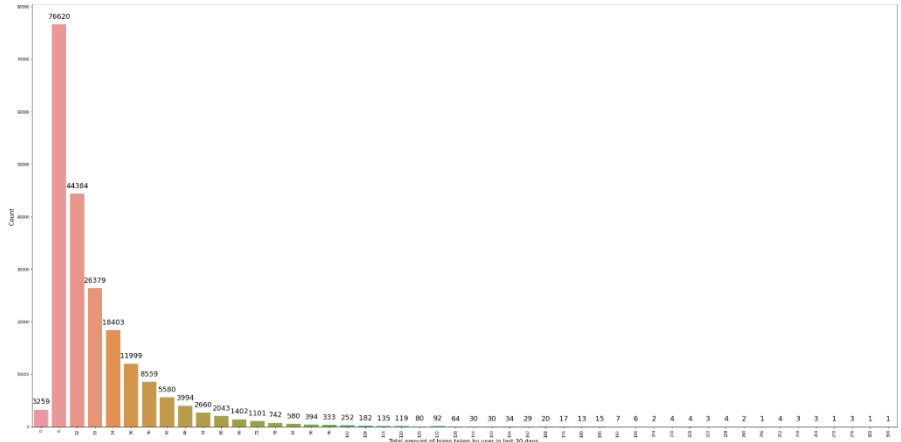
As we can see in above observation the data set in unbalance. The Defaulters are very less as compared to non-Defaulters.

```
#  Number of Loans taken by user in Last 30 days .
plt.figure(figsize=(40,20))
pd =sns.countplot(x = "cnt_loans30" ,  data=df)
for p in pd.patches:
    pd.annotate(format(p.get_height(), '.0f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                size=18,
                xytext = (0, 20),
                textcoords = 'offset points')
plt.xticks(rotation= 90)
plt.xlabel("number of loans taken by users ", size=15)
plt.ylabel("Count " ,size=15)
plt.title ("  No of loans taken by users in  30 days  " , size=30)
plt.show()
```



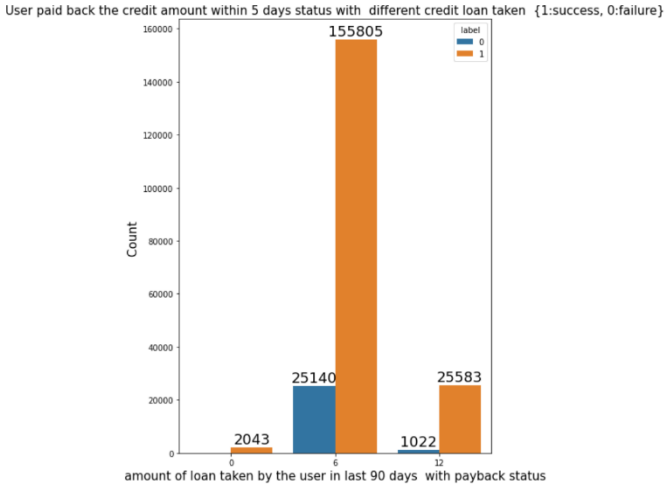
In the above observation we can see that most of the users have taken loans 1 loan in 30 days.

```
:
#  Total amount of Loans taken by user in Last 30 days .
plt.figure(figsize=(40,20))
pd =sns.countplot(x = "amnt_loans30" ,  data=df)
for p in pd.patches:
    pd.annotate(format(p.get_height(), '.0f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                size=18,
                xytext = (0, 20),
                textcoords = 'offset points')
plt.xticks(rotation= 90)
plt.xlabel("Total amount of loans taken by user in last 30 days ", size=15)
plt.ylabel("Count " ,size=15)
plt.title ("  " , size=15)
plt.show()
```



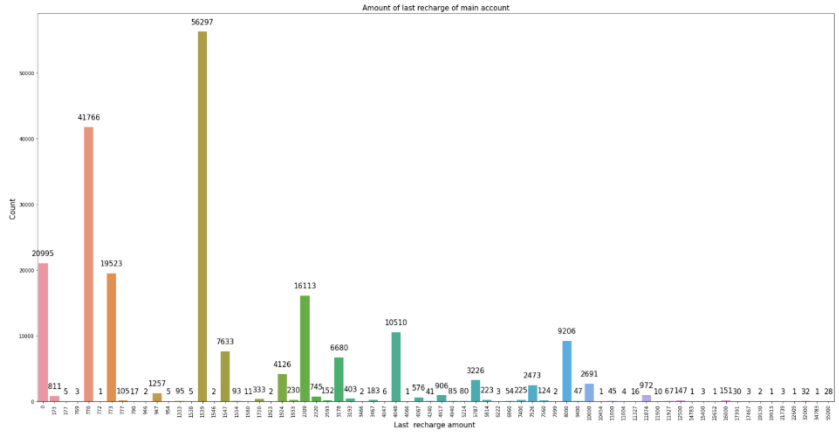
In the above observation we can see that most of the users have taken 6 (Indonesian rupiah) to 30 (Indonesian rupiah) loans. Only Few users have taken loan of more than 30 (Indonesian rupiah)

```
# User paid back the credit amount  within 5 days status with  different credit loan taken.
plt.figure(figsize=(7,10))
pd =sns.countplot(x = "maxamnt_loans90" ,  data=df  , hue= 'label')
for p in pd.patches:
    pd.annotate(format(p.get_height(), '.0f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                size=18,
                xytext = (0, 10),
                textcoords = 'offset points')
plt.xlabel("amount of loan taken by the user in last 90 days  with payback status", size=15)
plt.ylabel("Count " ,size=15)
plt.title (" User paid back the credit amount within 5 days status with  different credit loan taken  {1:success, 0:failure} " , size=15)
plt.show()
```



As we can see above majority of users have paid back their loan of 6 (Indonesian Rupiah) and 12(Indonesian rupiah) within 5 days of time.

```
#  Amount of last recharge of main account .
plt.figure(figsize=(30,15))
pd =sns.countplot(x = "last_rech_amt_ma" ,  data=df)
for p in pd.patches:
    pd.annotate(format(p.get_height(), '.0f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                size=15,
                xytext = (0, 20),
                textcoords = 'offset points')
plt.xticks(rotation= 90)
plt.xlabel("Last  recharge amount ", size=15)
plt.ylabel("Count " ,size=15)
plt.title (" Amount of last recharge of main account " , size=15)
plt.show()
```



As we can see above majority of people have recharge their main account with 770 and 1539 , very few users have recharged their main account with more than 10,000 .

➢ Interpretation of the Results

	Model	Accuracy_Score	Cross_val_score	Roc_auc_curve	F1_Score	Precison_Score	Recall_Score
0	LogisticRegression	0.864214	0.863482	0.520915	0.926655	0.995184	0.866956
1	KNeighborsClassifier	0.880572	0.880485	0.659863	0.933001	0.964773	0.903256
2	DecisionTreeClassifier	0.879815	0.875511	0.745793	0.930327	0.930945	0.929710
3	RandomForestClassifier	0.917355	0.916805	0.761600	0.953214	0.976776	0.930762
4	GaussianNB	0.649158	0.648381	0.730318	0.752320	0.618195	0.960773
5	AdaBoostClassifier	0.907888	0.907407	0.725205	0.948174	0.977583	0.920482
6	GradientBoostingClassifier	0.916660	0.915878	0.761508	0.952797	0.975851	0.930807

From the above tabel we can see that Random Forest is the best performing model .So we will select Random forest classifier as our final model.

```
rfc=RandomForestClassifier(criterion ="entropy" , max_features = 'log2' ,random_state= 56 )
rfc.fit(train_x,train_y)
predict=rfc.predict(test_x)
AS=accuracy_score(predict,test_y)
print("Accuracy Score =",AS)
cv_score=cross_val_score(k,x,y,cv=5,scoring="accuracy").mean()
print("The CV Score =",cv_score)
false_positive_rate,true_positive_rate,thresholds=roc_curve(test_y,predict)
roc_auc=auc(false_positive_rate,true_positive_rate)
print('roc_auc_score',roc_auc)
F1=f1_score(predict,test_y)
print("F1 Score= " ,F1)
precision=precision_score(predict,test_y)
print("Precision Score= " ,precision)
rec=recall_score(predict,test_y)
print("Recall Score= " ,rec)
```

```
Accuracy Score = 0.9173550339412775
The CV Score = 0.9158784387025557
roc_auc_score 0.7615999973864723
F1 Score=  0.9532143435886752
Precision Score=  0.9767761831336733
Recall Score=  0.9307624494224553
```

For Model Evaluation we are refereing Confusion Matrix

After we train a Randon forest classifier model on some training data, we will evaluate the performance of the model on some test data. For this, we use the Confusion Matrix

the accuracy of the model : - (TP + TN) / Total

Here, TP stands for True Positive which are the cases in which we predicted yes and the actual value was true. TN stands for True Negative which are the cases in which we predicted no and the actual value was false. FP stands for False Positive which are the cases which we predicted yes and the actual value was False. FN stands for False Negative which are the cases which we predicted No and the actual value was true

CONCLUSION

➢ Key Findings and Conclusions of the Study

Today, microfinance is widely accepted as a poverty-reduction tool, representing $70 billion in outstanding loans and a global outreach of 200 million clients.

The aim was to determine an appropriate quantative model for using financial information pertaining to the loan and customer behaviour on the mobile network to predict the outcome of the loan. Classification models are appropriate for dealing with the two distinct outcomes for customer behaviour of repayment nddeafaulter.

We have used different models for the prediction.

1) Using a Logistic Regression classifier, we can predict with 86.4% accuracy,

2) Using Gradient Boosting classifier, we can predict with 91.66% accuracy

3) Using a Random Forest classifier, we can predict with 91.73% accuracy,

4) Using a K-Nearest Neighbour classifier, we can predict with 88.05% accuracy,.

5) Using a Decision Tree classifier, we can predict with 87.98% accuracy,