Micro-Credit Defaulter Model

Submitted by:

Pratyush Raj

ACKNOWLEDGMENT

I would like to express my special gratitude to the “Flip Robo” team, who has given me this opportunity to deal with a dataset and it has helped me to improve my analysis skills. And I want to express my huge gratitude to Mr. Kashif (SME Flip Robo), she is the person who has helped me to get out of all the difficulties I faced while doing the project.A huge thanks to “Data trained” who are the reason behind my Internship at Fliprobo

References used in this project:

1. SCIKIT Learn Library Documentation
2. Data Science Projects with Python Second Edition by Packt
3. Hands-on Machine learning with scikit learn and tensor flow by Aurelien Geron
4. Stackoverflow.com to resolve some project-related queries.
5. Predicting Microfinance Credit Default: A Study of Nsoatreman Rural Bank, Ghana Ernest Yeboah Boateng
6. A Machine Learning Approach for Micro-Credit Scoring Apostolos Ampountolas

INTRODUCTION

Business Problem Framing:

A Microfinance Institution (MFI) is an organization that offers financial services to low-income populations. MFS becomes very useful when targeting especially the unbanked poor families living in remote areas with not many sources of income. The Microfinance services (MFS) provided by MFI are Group Loans, Agricultural Loans, Individual Business Loans, and so on.

Many microfinance institutions (MFI), experts, and donors are supporting the idea of using mobile financial services (MFS) which they feel are more convenient and efficient, and cost-saving, than the traditional high- touch model used for long for the purpose of delivering microfinance services. Though the MFI industry is primarily focusing on low-income families and is 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 global outreach of 200 million clients.

We are working with one such client that is in Telecom Industry. They are a fixed wireless telecommunications network provider. They have launched various products and have developed their 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.

Conceptual Background of the Domain Problem

Telecom Industries 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 poor customers that can help those in the need of the 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

a defaulter if he deviates from the path of paying back the loaned amount within the time duration of 5 days. For a loan amount of 5 (in Indonesian Rupiah), the payback amount should be 6 (in Indonesian Rupiah), while, for a loan amount of 10 (in Indonesian Rupiah), the payback amount should be 12 (in Indonesian Rupiah).

The sample data is provided to us from our client database. It is hereby given to you for this exercise. 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 the selection of customers.

We have to build a model which can be used to predict in terms of probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance of the loan. In this case, Label '1' indicates that the loan has been paid i.e., non-defaulter, while, Label '0' indicates that the loan has not been paid i.e., defaulter.

Review of Literature

What is Microfinance?

"Microfinance" is often seen as financial services for poor and low-income clients (Ayayi, 2012; Mensah, 2013; Tang, 2002). In practice, the term is often used more narrowly to refer to loans and other services from providers that identify themselves as "microfinance institutions" (MFIs) [Consultative Group to Assist the Poor (CGAP) 2010]. Microfinance can also be described as a setup of a number of different operators focusing on the financially under-served people with the aim of satisfying their need for poverty alleviation, social promotion, emancipation, and inclusion. Microfinance institutions reach and serve their target market in very innovative ways (Milana 2012).

The CGAP (2010) identifies some unique features of microfinance as follows:

➢ Delivery of very small loans to unsalaried workers

➢ Little or no collateral requirements

➢ Group lending and liability

➢ Pre-loan savings requirement

➢ Gradually increasing loan sizes

Default in Microfinance

Default in microfinance is the failure of a client to repay a loan. The default could be in terms of the amount to be paid or the timing of the payment. MFIs can sustain and increase the deployment of loans to stimulate the poverty reduction goal if repayment rates are high and consistent (Wongnaa 2013).

Machine Learning Techniques for microfinance & Finance

Machine learning-based systems are growing in popularity in research applications in most disciplines. Considerable decision-making knowledge from data has been acquired in the broad area of machine learning, in which decision-making tree-based ensemble techniques are recognized for supervised classification problems. A classification is an essential form of data analysis in data mining that formulates models while describing significant data classes (Rastogi and Shim 2000). Accordingly, such models estimate categorical class labels, which can provide users with an enhanced understanding of the data at large Han et al. (2012) resulting in significant advancements in classification accuracy.

Motivation for the Problem Undertaken

This project includes the real time problem for Microfinance Institution (MFI), and it is related to financial sectors, as I believe that with growing technologies and Idea can make a difference, there are so much in the financial market to explore and analyse and with Data Science the financial world becomes more interesting. The objective of the project is to prepare a model based on the sample dataset that classifies all loan defaulters and help our client in further investment and improvement in selection of customers. The model will be a good way for the management to understand whether the customer will be paying back the loaned amount within 5 days of insurance of loan.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem:

Whenever we employ any ML algorithm, statistical models or feature pre-processing in background lot of mathematical framework work. In this project we have done lot of data pre-processing & ML model building. In this section we dive into mathematical background of some of these algorithms.

1. Logistic Regression

$$\ln\left[\frac{P(Y)}{1-P(Y)}\right] = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k$$

Where;

$ln\left[\frac{P(Y)}{1-P(Y)}\right]$ is the log (odds) of credit default

Y is the dichotomous outcome which represents credit default (whether the loan was repaid or not) $X_1, X_2, \ldots, X_K$ are the predictor variables which are as educational level, number of dependents, type of loan, adequacy of the loan facility, duration for repayment of loan, number of years in business, cost of capital and period within the year the loan was advanced to the client $\beta_o, \beta_1, \beta_2, \ldots, \beta_k$ are the regression (model) coefficients

2. Decision Tree Classifier

Algorithm: Train Tree
Input: D, a dataset of training records of the form (**X**, Y).
Output: Root node R of a trained decision tree
1) Create a root node R
2) If a stopping criterion has been reached then label R with the most common value of Y in D and output R
3) For each input variable Xi in **X**
   a. Find the test Ti whose partition $D1$, $D2 \ldots Dn$ performs best according to the chosen splitting metric.
   b. Record this test and the value of the splitting metric
4) Let Ti be the best test according to the splitting metric, let V be the value of the splitting metric, and let $D1$, $D2 \ldots Dn$ be the partition.
5) If $V<threshold$
   a. Label R with the most common value of Y in D and output R
6) Label R with Ti and make a child node Ci of R for each outcome Oi of Ti.
7) For each outcome Oi of Ti
   a. Create a new child node Ci of R, and label the edge Oi
   b. Set Ci = Train Tree(Di)
8) Output R

$$P_Y(D) = \left(\frac{|\sigma_{Y=y_1}(D)|}{|D|}, \frac{|\sigma_{Y=y_2}(D)|}{|D|}, \ldots, \frac{|\sigma_{Y=y_k}(D)|}{|D|}\right)$$

$$\sigma_\varphi(D) = set\ of\ all\ X \in D\ s.t.\ the\ expression\ \varphi\ holds\ true\ for\ X$$

And the impurity measure of a dataset D is denoted as,

$$impurity_Y(D) = \phi(P_Y(D))$$

Lastly, we define the goodness-of-split (or change in purity) with respect to an input variable Xi that has m possible values $v1,..vm$ and a dataset D as,

$$\Delta i_Y(X_i, D) = impurity_Y(D) - \sum_{j=1}^{m} \frac{|\sigma_{X_i=v_j}(D)|}{|D|} impurity_Y(\sigma_{X_i=v_j}(D))$$

Impurity based splitting criteria use an impurity function ϕ plugged into the general goodness-of-split equation defined above.

$$Entropy(P) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Plugging in $Entropy$ as our function ϕ gives us $InformationGainY\ (Xi, D)$:

$$InformationGain_Y(X_i, D) = Entropy(P_Y(D)) - \sum_{j=1}^{m} \frac{|\sigma_{X_i=v_j}(D)|}{|D|} Entropy\left(P_Y(\sigma_{X_i=v_j}(D))\right)$$

$$InformationGain_Y(X_i, D) = EntropyBeforeSplit - EntropyAfterSplit$$

3. Random Forest Classifier

$$r_N(X, \beta) = \frac{\sum_{i=1}^{N} y_i^1 x_j \in A_N(X, \beta)}{\sum_{i=1}^{N} 1_{x_j} \in A_N(X, \beta)} 1_{L_N}$$

where $L_N = \sum_{i=1}^{N} 1_{x_j} \in A_N(x, \beta) \neq 0$. We can achieve the estimate of r_N with respect to the parameter β by taking the expectation of r_N (Addo et al. 2018).

4. Extra Trees Classifier

The extremely randomized trees classifier (extra trees classifier) establishes an ensemble of decision trees following an original top-down approach. Thus, it is similar to a random forest classifier differing only in the decision trees' mode of construction. Each decision tree is formed from the initial training data set sample. It entails random both element and cut-point choice while dividing a node of a tree.

Data Sources and their formats

The data set comes from my internship company – Fliprobo technologies in excel format.There are 37 columns and 209593 rows in this dataset. The different features in dataset are as below:

```
# Importing dataset CSV file using pandas
df= pd.read_csv('Data_file.csv')
```

```
print('No. of Rows :',df.shape[0])
print('No. of Columns :',df.shape[1])
pd.set_option('display.max_columns',None) # # This will enable us to see truncated columns
df.head(10)
```

```
No. of Rows : 209593
No. of Columns : 37
```

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	cnt_ma_recl
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	5787	
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539	
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	947	
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309	
5	6	1	35819I70783	568.0	2257.362667	2261.460000	368.13	380.13	2.0	0.0	1539	
6	7	1	96759I84459	545.0	2876.641667	2883.970000	335.75	402.90	13.0	0.0	5787	
7	8	1	09832I90846	768.0	12905.000000	17804.150000	900.35	2549.11	4.0	55.0	3178	
8	9	1	59772I84450	1191.0	90.695000	90.695000	2287.50	2287.50	1.0	0.0	1539	
9	10	1	56331I70783	536.0	29.357333	29.357333	612.96	612.96	11.0	0.0	773	

Data Preprocessing Done

Missing value check – Data contain no missing value

Data integrity check –

```
df.duplicated().sum() # This will check the duplicate data for all columns.
```

```
1
```

```
df.duplicated('msisdn').sum()  # This will check the duplicate data for all columns.
```

```
23350
```

```
# Droping duplicate entries
df.drop_duplicates(keep='last',inplace=True)
```

```
df.isin(['NA','N/A','-',' ','?',' ?']).sum().any()
```

```
False
```

```
df.shape
```

```
(209592, 37)
```

```
plt.figure(figsize=(10,8))
sns.heatmap(df.isnull())
plt.show()
```

Statistical Matrix –

```
df['aon'].describe()
```

```
count    209592.000000
mean       8112.380399
std       75696.261220
min         -48.000000
25%         246.000000
50%         527.000000
75%         982.000000
max      999860.755168
Name: aon, dtype: float64
```

```
(999860/365)
```

```
2739.3424657534247
```

```
df[df['aon']<0].value_counts().sum()
```

```
1539
```

```
df[df['aon'] > 3650].value_counts().sum()  # Assume people stay one network for 10 years.
```

```
2089
```

Now we will convert all negative values into absolute values.

```
#Converting all negative values to positive values in above columns
df['aon']=abs(df['aon'])
df['daily_decr30']=abs(df['daily_decr30'])
df['daily_decr90']=abs(df['daily_decr90'])
df['rental30']=abs(df['rental30'])
df['rental90']=abs(df['rental90'])
df['last_rech_date_ma']=abs(df['last_rech_date_ma'])
df['last_rech_date_da']=abs(df['last_rech_date_da'])
```

Data error and correction in maxamnt_loans30 column

```
df['maxamnt_loans30'].describe()
```

```
count    209592.000000
mean        274.660029
std        4245.274734
min           0.000000
25%           6.000000
50%           6.000000
75%           6.000000
max       99864.560864
Name: maxamnt_loans30, dtype: float64
```

Feature Engineering

```
df['aon'].describe()
```

```
count    209592.000000
mean       8112.380399
std       75696.261220
min         -48.000000
25%         246.000000
50%         527.000000
75%         982.000000
max      999860.755168
Name: aon, dtype: float64
```

```
(999860/365)
```

```
2739.3424657534247
```

```
df[df['aon']<0].value_counts().sum()
```

```
1539
```

```
df[df['aon'] > 3650].value_counts().sum()  # Assume people stay one network for 10 years.
```

```
2089
```

Now we will convert all negative values into absolute values.

```
#Converting all negative values to positive values in above columns
df['aon']=abs(df['aon'])
df['daily_decr30']=abs(df['daily_decr30'])
df['daily_decr90']=abs(df['daily_decr90'])
df['rental30']=abs(df['rental30'])
df['rental90']=abs(df['rental90'])
df['last_rech_date_ma']=abs(df['last_rech_date_ma'])
df['last_rech_date_da']=abs(df['last_rech_date_da'])
```

Outliers Detection and removal –

```
df1=df.copy()
```

```
from scipy.stats import zscore
z = np.abs(zscore(df))
threshold = 3
df2 = df1[(z<3).all(axis = 1)]

print ("Shape of the dataframe before removing outliers: ", df1.shape)
print ("Shape of the dataframe after removing outliers: ", df2.shape)
print ("Percentage of data loss post outlier removal: ", (df1.shape[0]-df2.shape[0])/df1.shape[0]*100)

df1=df2.copy() # reassigning the changed dataframe name to our original dataframe name
```

```
Shape of the dataframe before removing outliers:  (209592, 35)
Shape of the dataframe after removing outliers:  (160499, 35)
Percentage of data loss post outlier removal:  23.42312683690217
```

Huge amount of data loss in Z-score , which is not acceptable

```
df.shape
```

```
(209592, 35)
```

```
df1=df.copy()
Q1 = df1.quantile(0)
Q3= df1.quantile(0.99)
IQR = Q3 - Q1
print(IQR)
```

```
data = df1[~((df1 < (Q1 - 1.5 * IQR)) |(df1 > (Q3 + 1.5 * IQR))).any(axis=1)]
print(data.shape)
```

```
(198174, 35)
```

```
print("Percentage Data Loss :",((209592-198174)/209592)*100,'%')
```

```
Percentage Data Loss : 5.447727012481392 %
```

Skewness in features & it's transformation

click to expand output; double click to hide output

```
data.drop(['fr_da_rech30'],axis=1,inplace=True)
data.drop(['fr_da_rech90'],axis=1,inplace=True)
```

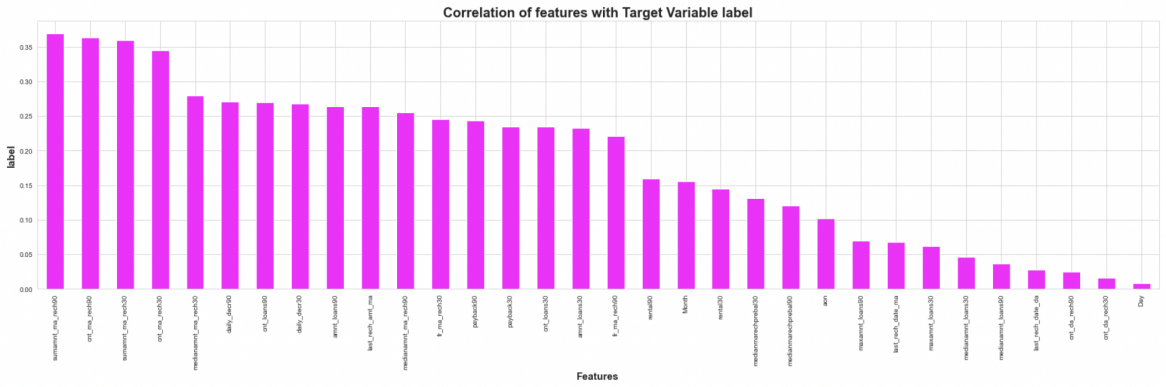
```
skew_fea=['aon','daily_decr30', 'daily_decr90', 'rental30','rental90','last_rech_date_ma', 'last_rech_date_da',
          'last_rech_amt_ma','cnt_ma_rech30', 'fr_ma_rech30', 'sumamnt_ma_rech30','medianamnt_ma_rech30',
          'medianmarechprebal30', 'cnt_ma_rech90','fr_ma_rech90', 'sumamnt_ma_rech90', 'medianamnt_ma_rech90',
          'medianmarechprebal90', 'cnt_da_rech30','cnt_da_rech90', 'cnt_loans30', 'amnt_loans30',
          'maxamnt_loans30', 'medianamnt_loans30', 'cnt_loans90', 'amnt_loans90',
          'maxamnt_loans90','medianamnt_loans90', 'payback30', 'payback90']
```

```
from sklearn.preprocessing import PowerTransformer
scaler = PowerTransformer(method='yeo-johnson')
```

```
data[skew_fea] = scaler.fit_transform(data[skew_fea].values)
```

- Data Inputs- Logic- Output Relationships

```
plt.figure(figsize = (24,8))
data.corr()['label'].drop(['label']).sort_values(ascending=False).plot(kind='bar',cmap = 'spring')
plt.xlabel('Features',fontsize=15,fontweight='bold')
plt.ylabel('label',fontsize=15,fontweight='bold')
plt.title('Correlation of features with Target Variable label',fontsize = 20,fontweight='bold')
plt.show()
```




```
from imblearn.over_sampling import SMOTE
```

```
data.shape
```

```
(198174, 33)
```

```
# Oversampleing using SMOTE Techniques
oversample = SMOTE()
X_scale, Y = oversample.fit_resample(X_scale, Y)
```

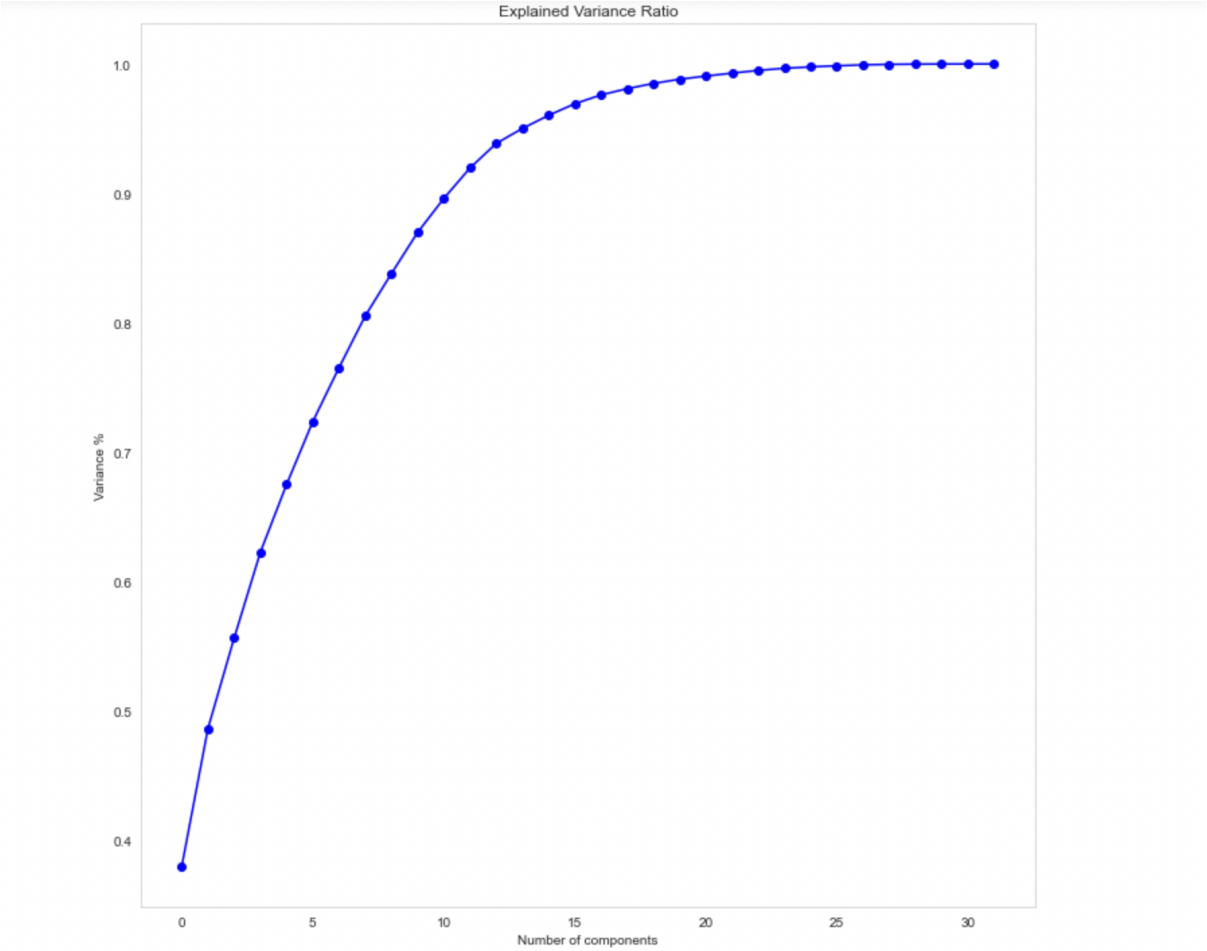
```
Y.value_counts()
```

```
0    173461
1    173461
Name: label, dtype: int64
```

We have successfully resolved the class imbalanced problem

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

.



- Hardware and Software Requirements and Tools Used

Hardware Used -

1. Processor — ios processor with 2.4GHZ

2. RAM — 8 GB

3. GPU — 2GB AMD Radeon Graphics card

Software utilised -

1.Anaconda – Jupyter Notebook

Libraries Used –

Different libraries are used while building ML model and Visualisation of data.

Machine Learning Model Building

```
]: from sklearn.model_selection import train_test_split
   from sklearn.linear_model import LogisticRegression
   from sklearn.naive_bayes import GaussianNB
   from sklearn.svm import SVC
   from sklearn.tree import DecisionTreeClassifier
   from sklearn.neighbors import KNeighborsClassifier
   from sklearn.ensemble import RandomForestClassifier
   from sklearn.ensemble import ExtraTreesClassifier
   from sklearn.neighbors import KNeighborsClassifier
   from sklearn.metrics import accuracy_score, confusion_matrix,classification_report,f1_score
```

```
]: X_train, X_test, Y_train, Y_test = train_test_split(principle_x, Y, random_state=101, test_size=.28)
   print('Training feature matrix size:',X_train.shape)
   print('Training target vector size:',Y_train.shape)
   print('Test feature matrix size:',X_test.shape)
   print('Test target vector size:',Y_test.shape)
```

```
Training feature matrix size: (249783, 11)
Training target vector size: (249783,)
Test feature matrix size: (97139, 11)
Test target vector size: (97139,)
```

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

The target variable label has two classes i.e., label ’1’ indicates non- defaulter & label ‘0’ indicates defaulter. Our objective is to predict whether customer is defaulter or not. This becomes binary classification problem which can be solved using various classification algorithms. In order to gain high accuracy of model we will train model with different classification model and select final model among them. To enhance performance of best model will employ hyper parameter tuning over it. At end we will save our final model using joblib.

Testing of Identified Approaches (Algorithms)

The different classification algorithm used in this project to build ML model are as below:

- ❖ Logistics Regression

❖ Decision Tree Classifier

❖ Random Forest Classifier

❖ Extra Tree Classifier

Run and Evaluate selected models

1. LOGISTICS REGRESSION

Logistic Regression

```
X_train, X_test, Y_train, Y_test = train_test_split(principle_x, Y, random_state=62, test_size=.28)
log_reg=LogisticRegression()
log_reg.fit(X_train,Y_train)
y_pred=log_reg.predict(X_test)
print('Logistics Regression Evaluation')
print('\n')
print('Accuracy Score of Logistics Regression :', accuracy_score(Y_test, y_pred))
print('\n')
print('Confusion matrix of Logistics Regression :','\n',confusion_matrix(Y_test, y_pred))
print('\n')
print('classification Report of Logistics Regression','\n',classification_report(Y_test, y_pred))
```

```
Logistics Regression Evaluation


Accuracy Score of Logistics Regression : 0.761774364570358


Confusion matrix of Logistics Regression :
 [[36873 11485]
 [11656 37125]]


classification Report of Logistics Regression
              precision    recall  f1-score   support

           0       0.76      0.76      0.76     48358
           1       0.76      0.76      0.76     48781

    accuracy                           0.76     97139
   macro avg       0.76      0.76      0.76     97139
weighted avg       0.76      0.76      0.76     97139
```

```
from sklearn.model_selection import cross_val_score
CVscore = cross_val_score(log_reg, principle_x, Y, cv =5)
print('Cross Validation Score', log_reg, ':','\n')
print("CVScore :" ,CVscore)
print("Mean CV Score :",CVscore.mean())
print("Std deviation :",CVscore.std())
```

```
Cross Validation Score LogisticRegression() :

CVScore : [0.76079844 0.76108669 0.76455667 0.76494581 0.76441254]
Mean CV Score : 0.7631600314621998
Std deviation : 0.001821221103890908
```

2. DECISION TREE CLASSIFIER

```
X_train, X_test, Y_train, Y_test = train_test_split(principle_x, Y, random_state=62, test_size=.28)
dtc=DecisionTreeClassifier()
dtc.fit(X_train,Y_train)
y_pred=dtc.predict(X_test)
print('Decision Tree Classifier Evaluation')
print('\n')
print('Accuracy Score of Decision Tree Classifier :', accuracy_score(Y_test, y_pred))
print('\n')
print('Confusion matrix of Decision Tree Classifier :','\n',confusion_matrix(Y_test, y_pred))
print('\n')
print('classification Report of Decision Tree Classifier','\n',classification_report(Y_test, y_pred))
```

```
Decision Tree Classifier Evaluation


Accuracy Score of Decision Tree Classifier : 0.8565663636644396


Confusion matrix of Decision Tree Classifier :
 [[42411  5947]
 [ 7986 40795]]


classification Report of Decision Tree Classifier
              precision    recall  f1-score   support

           0       0.84      0.88      0.86     48358
           1       0.87      0.84      0.85     48781

    accuracy                           0.86     97139
   macro avg       0.86      0.86      0.86     97139
weighted avg       0.86      0.86      0.86     97139
```

to expand output; double click to hide output

```
on import cross_val_score
CVscore = cross_val_score(dtc, principle_x, Y, cv =5)
print('Cross Validation Score', dtc, ':')
print("CVScore :" ,CVscore)
print("Mean CV Score :",CVscore.mean())
print("Std deviation :",CVscore.std())
```

```
Cross Validation Score DecisionTreeClassifier() :
CVScore : [0.8636737  0.86092095 0.8643203  0.86574715 0.86214401]
Mean CV Score : 0.8633612227757034
Std deviation : 0.0016820618052549791
```

3. RANDOM FOREST CLASSIFIER

```
X_train, X_test, Y_train, Y_test = train_test_split(principle_x, Y, random_state=62, test_size=.28)
rfc=RandomForestClassifier()
rfc.fit(X_train,Y_train)
y_pred=rfc.predict(X_test)
print('Random Forest Classifier Evaluation')
print('\n')
print('Accuracy Score of Random Forest Classifier :', accuracy_score(Y_test, y_pred))
print('\n')
print('Confusion matrix of Random Forest Classifier :','\n',confusion_matrix(Y_test, y_pred))
print('\n')
print('classification Report of Random Forest Classifier','\n',classification_report(Y_test, y_pred))
```

```
Random Forest Classifier Evaluation


Accuracy Score of Random Forest Classifier : 0.9200732970279702


Confusion matrix of Random Forest Classifier :
 [[45184  3174]
 [ 4590 44191]]


classification Report of Random Forest Classifier
              precision    recall  f1-score   support

           0       0.91      0.93      0.92     48358
           1       0.93      0.91      0.92     48781

    accuracy                           0.92     97139
   macro avg       0.92      0.92      0.92     97139
weighted avg       0.92      0.92      0.92     97139
```

```
from sklearn.model_selection import cross_val_score
CVscore = cross_val_score(rfc, principle_x, Y, cv =5)
print('Cross Validation Score', rfc, ':','\n')
print("CVScore :" ,CVscore)
print("Mean CV Score :",CVscore.mean())
print("Std deviation :",CVscore.std())
```

```
Cross Validation Score RandomForestClassifier() :

CVScore : [0.92587735 0.92610795 0.92570333 0.92823994 0.92684192]
Mean CV Score : 0.9265540989700813
Std deviation : 0.000927981160949982
```

4. EXTRA TREE CLASSIFIER

```
X_train, X_test, Y_train, Y_test = train_test_split(principle_x, Y, random_state=62, test_size=.28)
etc=ExtraTreesClassifier()
etc.fit(X_train,Y_train)
y_pred=etc.predict(X_test)
print('Extra Trees Classifier Evaluation')
print('\n')
print('Accuracy Score of Extra Trees Classifier :', accuracy_score(Y_test, y_pred))
print('\n')
print('Confusion matrix of Extra Trees Classifier :','\n',confusion_matrix(Y_test, y_pred))
print('\n')
print('classification Report of Extra Trees Classifier','\n',classification_report(Y_test, y_pred))
```

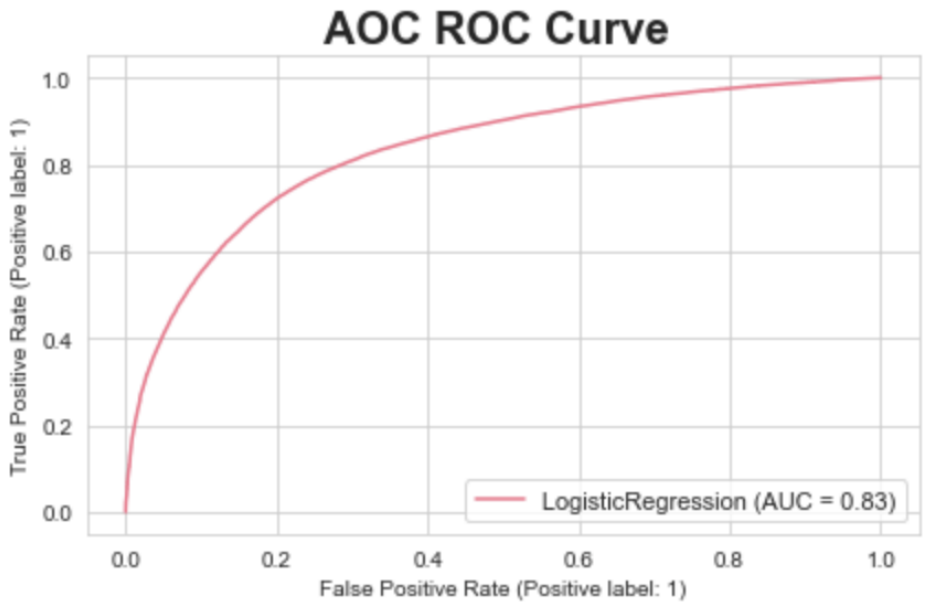
```
Extra Trees Classifier Evaluation


Accuracy Score of Extra Trees Classifier : 0.9351136000988275


Confusion matrix of Extra Trees Classifier :
 [[45886  2472]
 [ 3831 44950]]


classification Report of Extra Trees Classifier
              precision    recall  f1-score   support

           0       0.92      0.95      0.94     48358
           1       0.95      0.92      0.93     48781

    accuracy                           0.94     97139
   macro avg       0.94      0.94      0.94     97139
weighted avg       0.94      0.94      0.94     97139
```

```
: from sklearn.model_selection import cross_val_score
  CVscore = cross_val_score(etc, principle_x, Y, cv =5)
  print('Cross Validation Score', etc, ':','\n')
  print("CVScore :" ,CVscore)
  print("Mean CV Score :",CVscore.mean())
  print("Std deviation :",CVscore.std())
```

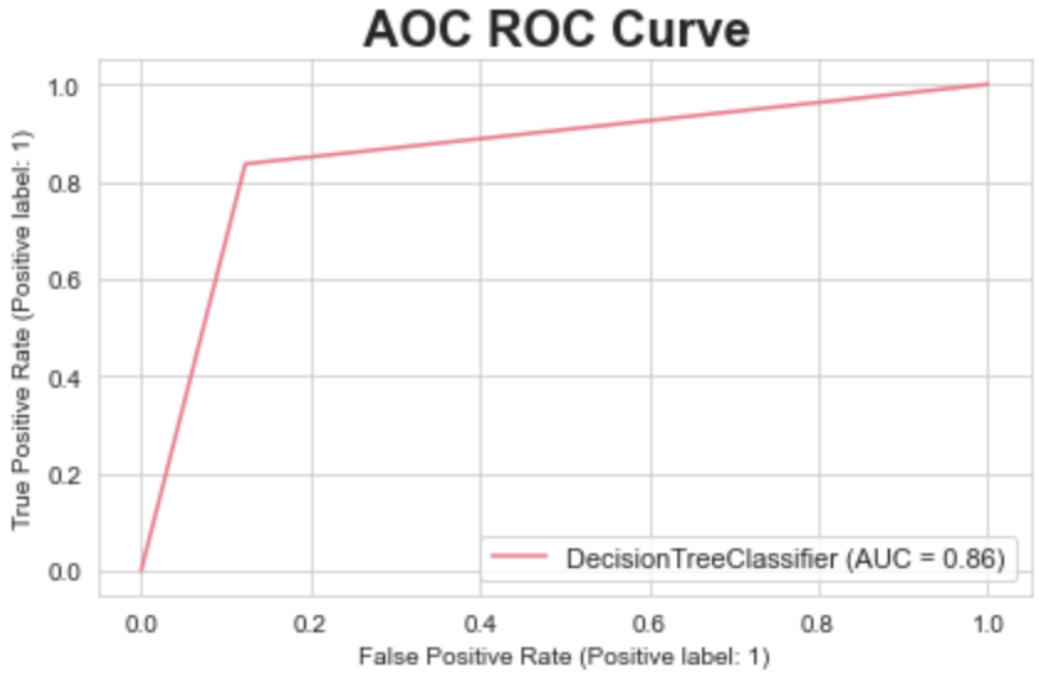
```
Cross Validation Score ExtraTreesClassifier() :

CVScore : [0.94376306 0.94098148 0.9397988  0.94256601 0.93966909]
Mean CV Score : 0.9413556879264922
Std deviation : 0.001591783423299921
```

Extra Tree classifier gives maximum accuracy and cross validation score.

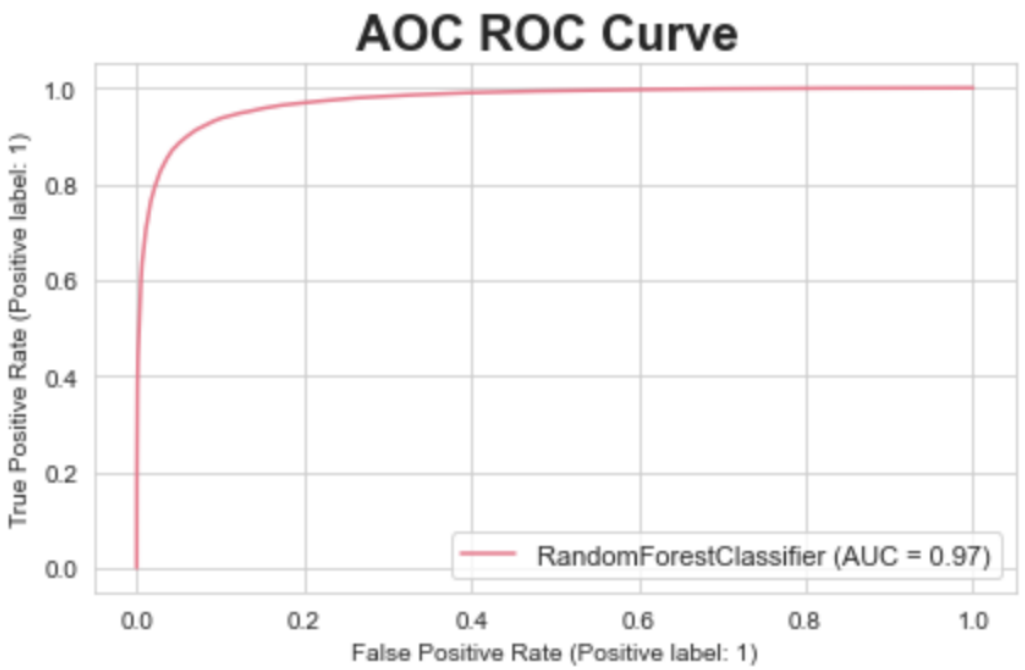
AOC -ROC CURVE OF DIFFERENT ML MODELS



Auc Score :
0.7617775129525993



Auc Score :
0.8566550505798042



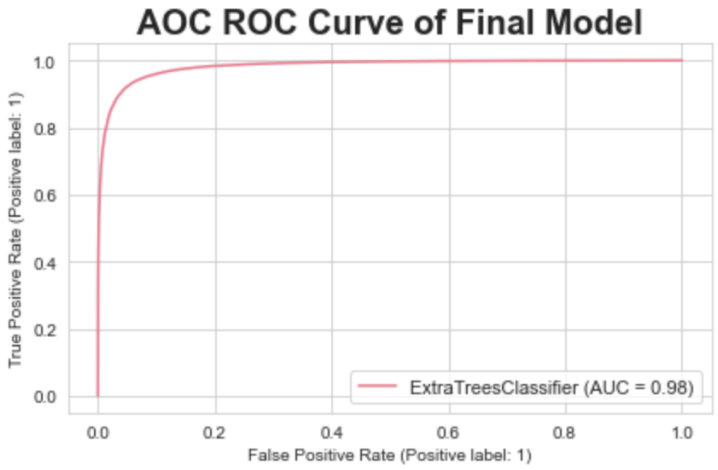

```
Auc Score :
 0.9201352595959453
```

We can see Extra Tree Classifier gives maximum AUC. It also gives us highest accuracy score and cross validation score. Hyper parameter tuning perform on this model to enhance accuracy of model.

Final model is saved using joblib library.

Final Model AOC ROC

```
]4]: from sklearn.metrics import roc_auc_score
     from sklearn.metrics import roc_curve
     from sklearn.metrics import plot_roc_curve
     disp = plot_roc_curve(etc,X_test,Y_test)
     plt.legend(prop={'size':11}, loc='lower right')
     plt.title('AOC ROC Curve of Final Model',fontsize=20,fontweight='bold')
     plt.show()
     auc_score = roc_auc_score(Y_test, etc.predict(X_test))
     print('Auc Score :' ,'\n', auc_score)
```

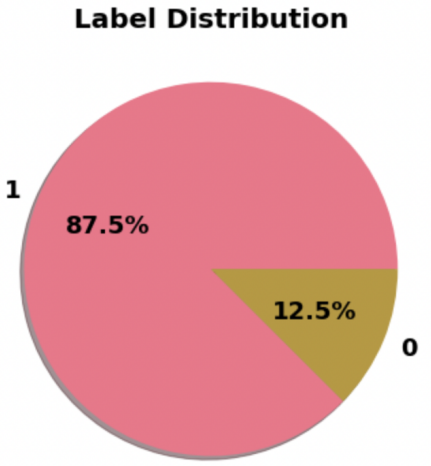


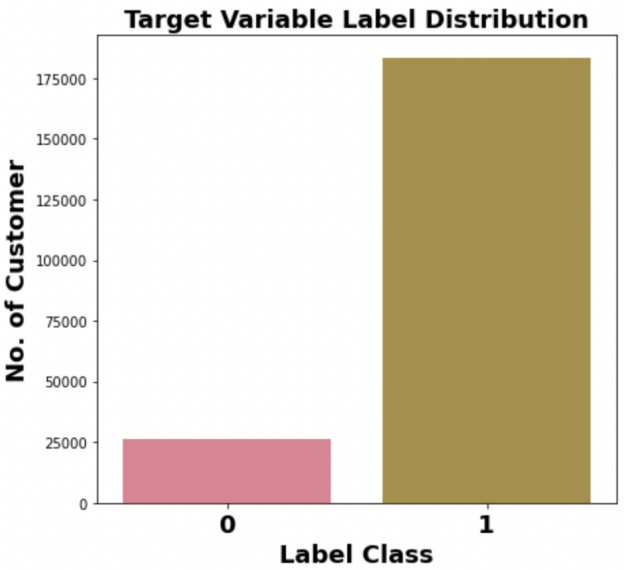

```
Auc Score :
 0.9360190736308613
```

Saving Final Model

```
]5]: import joblib
     joblib.dump(etc,'Micro_Credit_Defaulter.pkl')
]5]: ['Micro_Credit_Defaulter.pkl']
```

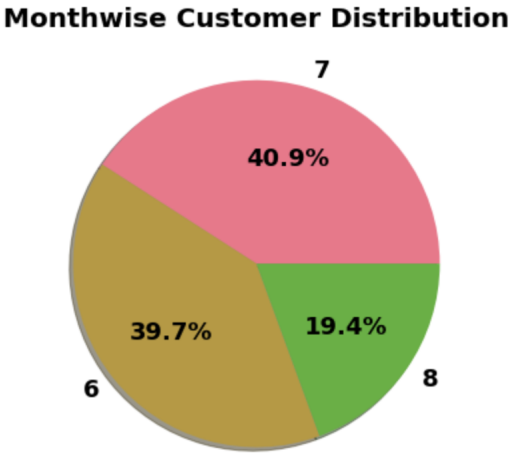
- Visualizations

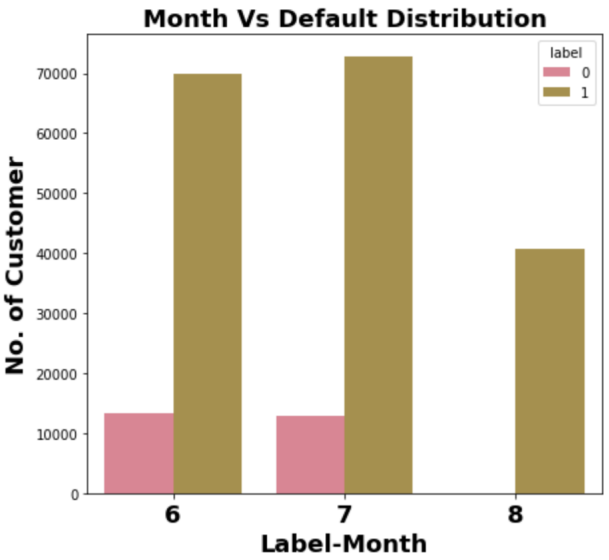




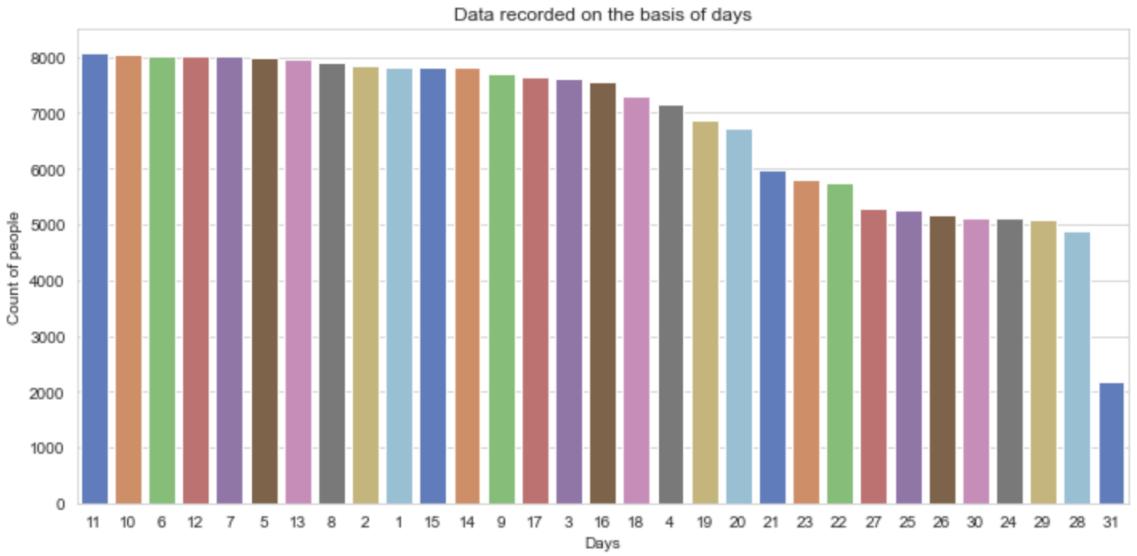
Label class 1 represent Non-defaulter while Label class 0 represent Defaulter i.e. Loan not paid

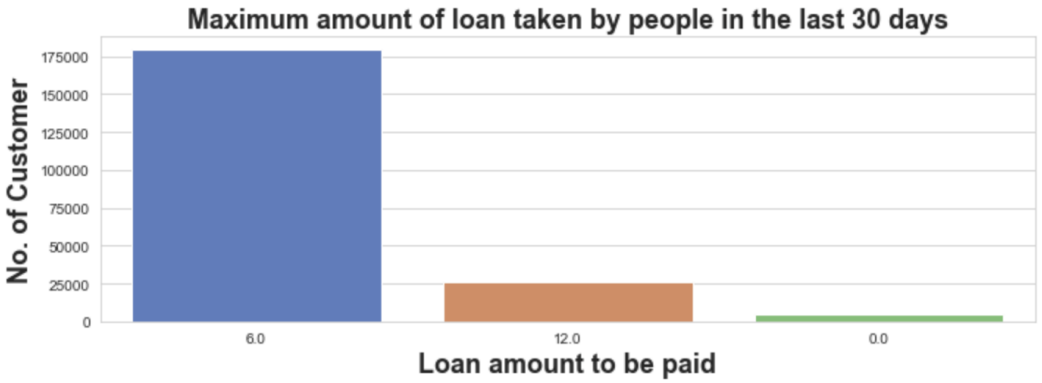
We can see Most of customers are Non-defaulter while very few are defaulter.

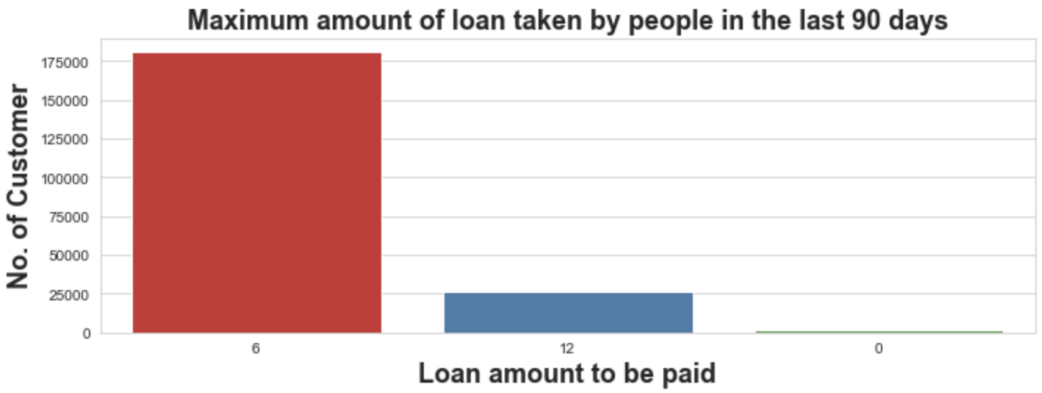




Comment : Most of data belong to month 6 and 7, followed my month 8. We can see very few defaulter in month 8.



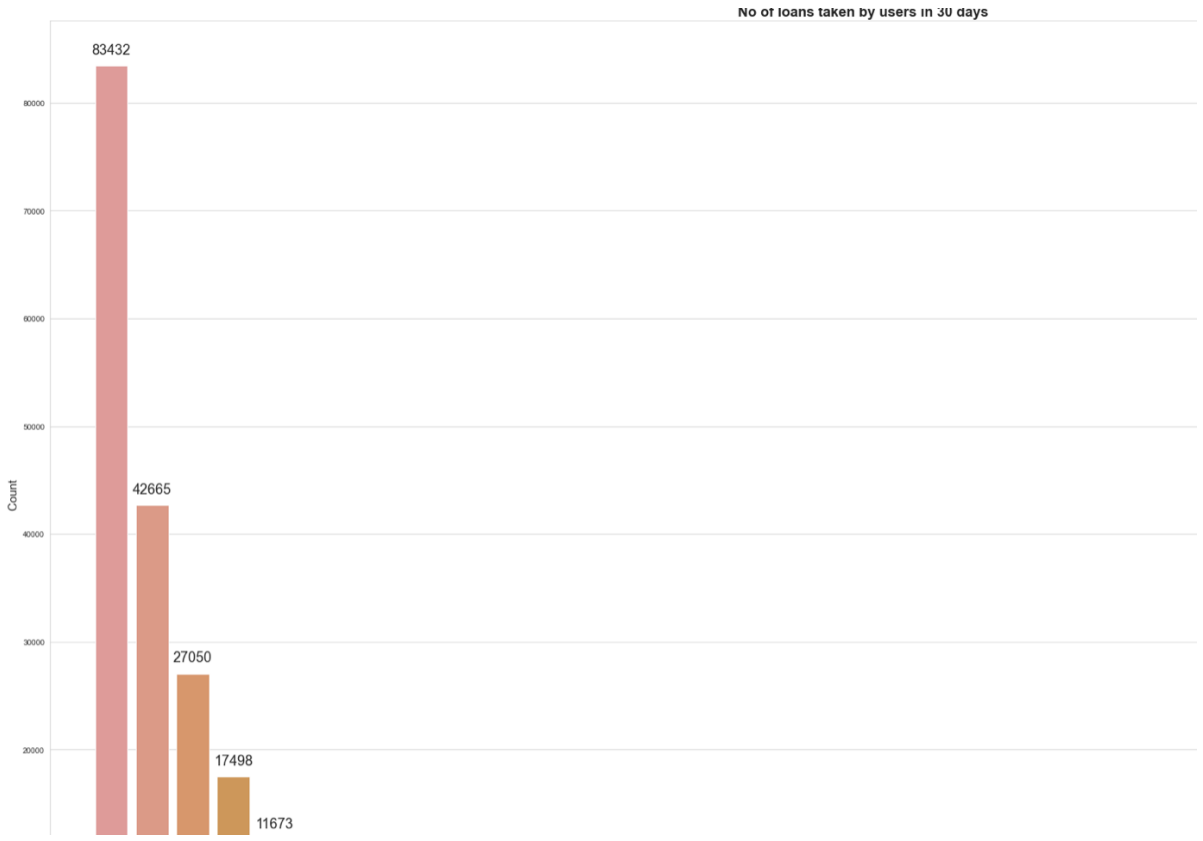


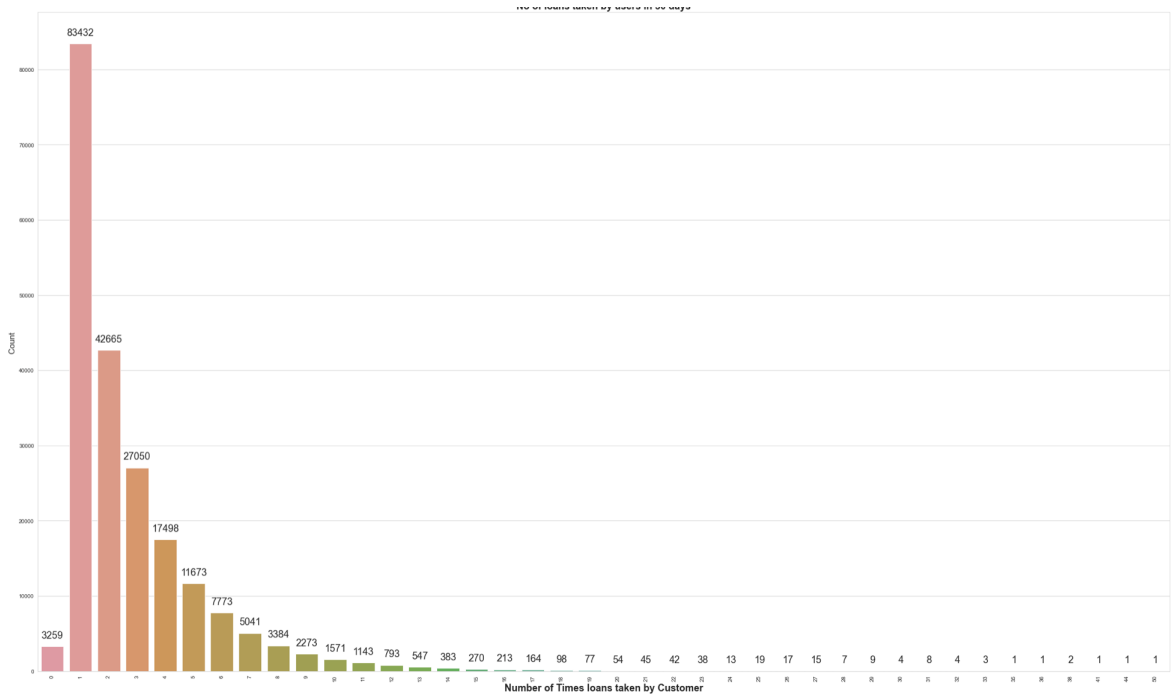


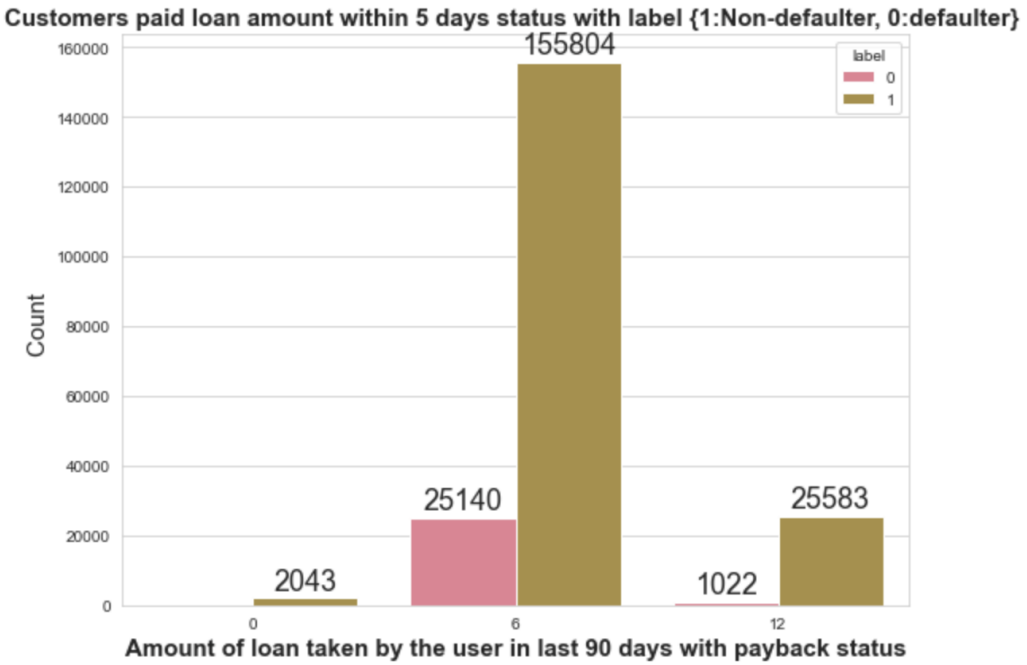

```
6.0     179192
12.0     26109
0.0       4291
Name: maxamnt_loans30, dtype: int64
```

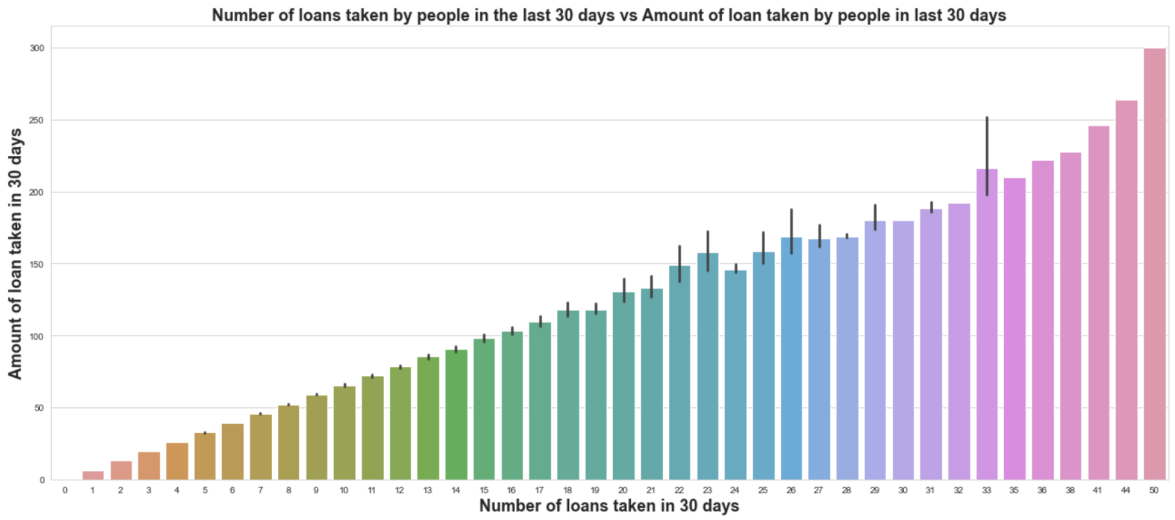
```
6     180944
12     26605
0       2043
Name: maxamnt_loans90, dtype: int64
```

Observations: In 30 days, maximum number of people had taken 6Rs as the loan amount and the number of people is 179192 whereas the number of people had not taken loan and their number is 4291. In 90 days, maximum number of people had taken 6Rs as the loan amount and the number of people is 180944 whereas the number of people had not taken loan and their number is 2043. Maximum number of people had taken 12Rs as the loan amount within 90 days and their number is 26605 whereas for 30 days the number of people who had taken 12Rs is 26109 respectively.


No of loans taken by users in 30 days
Count
83432
42665
27050
17498
11673


Count
83432
42665
27050
17498
11673
7773
5041
3384
2273
1571
1143
793
547
383
270
213
164
98
77
54
45
42
38
13
19
17
15
7
9
4
8
4
3
1
1
2
1
1
1
3259
Number of Times loans taken by Customer


Customers paid loan amount within 5 days status with label {1:Non-defaulter, 0:defaulter}
label
0
1
Count
160000
140000
120000
100000
80000
60000
40000
20000
0
155804
25140
25583
2043
1022
0
6
12
Amount of loan taken by the user in last 90 days with payback status


Number of loans taken by people in the last 30 days vs Amount of loan taken by people in last 30 days
Amount of loan taken in 30 days
300
250
200
150
100
50
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 35 36 38 41 44 50
Number of loans taken in 30 days

- Interpretation of the Results

As this dataset belongs from the year 2016, the data are recorded in the month of June, July and August. From the visualization, we can say that the most loan amount taken is rupiah 6 and most of the users are paying the loan within the time frame of 5 days, but many early users failed to do so. They usually take almost 7 to 8 days to pay the loan amount and even the valuable customers some time fails to pay the amount within the time frame.

➢One more thing I noticed that, the smaller number of loans taken by the people are more defaulters and the frequently loan taking customers are less defaulters.

➢Most importantly, the people are paying the amount early or lately and sometimes they might fail to pay within the time frame, but I observed that almost 80% of users are paying the amount within 7-8 days. It is recommended that to extent loan repayment time frame from 5 days to 7 days.

➢The collected data is only for one Telecom circle area as per Dataset Documentation so that we had dropped that column.

➢Customer who takes a greater number of loans are non-defaulters (i.e., 98% of the category) as they repay the loan within the given time i.e., 5 days

CONCLUSION

- Key Findings and Conclusions of the Study

Table 1

Algorithm	Accuracy score	CV score
Logistic Regression	0.761774364	0.763160031
Decision Tree Classifier	0.856566363	0.863361222
Random Forest Classifier	0.920073297	0.926554098
Extra Tree Classifier	0.935113600	0.001591783

- Limitations of this work and Scope for Future Work

Limited computational resources put limitation on optimization through hyper parameter tuning. Accuracy of model can increase with hyperparameter tuning with several different parameter. Here we use only two parameters for tuning.

Data is imbalanced, we utilised SMOTE for it but if get label data which at least in ratio of 70:30, It can give us much more realistic model.