NAME OF THE PROJECT

Micro-Credit Defaulter Model

Submitted by: ***Prathamesh Nayak***

Batch No: ***29***

Introduction

1. 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 much 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 hightouch model used since long for the purpose of delivering microfinance services. Though the MFI industry is primarily focusing on low-income families and are very useful in such areas, the implementation of MFS has been uneven with both significant challenges and successes. Today, microfinance is widely accepted as a poverty-reduction tool, representing $70 billion in outstanding loans and a global outreach of 200 million clients. We are working with one such client that is in the 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.

2. 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 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. 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 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 selection of customers. We have to build a model which can be used 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. 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.

3.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 Implicit guarantee of ready access to future loans if present loans are repaid fully and promptly Microfinance

is seen as a catalyst for poverty alleviation, delivered in innovative and sustainable ways to assist the underserved poor, especially in developing countries (Dixon, Ritchie, & Siwale, 2007; Spiegel, 2012). Economic development may be achieved by helping the underserved poor to engage in income-generating/poverty reduction activities through entrepreneurship (Milana 2012). On December 18, 1997, the United Nations (UN) passed a microcredit resolution, also known as the Grameen Dialogue of 1998 at its General Assembly. The resolution was adopted because of the importance of microcredit programs in poverty reduction (Elahi & Demopoulos 2004). The UN later declared the year 2005 as the International Year of Micro Credit. Globally, Microfinance has become an important sector. It is estimated that more than 3,500 institutions are meeting the demands of 205 million clients with a volume that is still uncertain but substantial (Maes and Reed 2012).

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 deployment of loans to stimulate the poverty reduction goal if repayment rates are high and consistent (Wongnaa 2013). Machine Learning Techniques for microfinance & finance Pollio and Obuobie [ ] applied logistic regression on four factors and concluded that the probability of default increases with the number of dependents, whether the proceeds are used to acquire fixed assets, the frequency of monitoring, decreases with the availability of non-business income, years in business, the number of guarantors, whether the proceeds were used for working capital purposes and whether the client is a first-time borrower. In Addo et al. (2018) the authors examined credit risk scoring by employing various machine and deep learning techniques. The authors used binary classifiers in modeling loan default probability (DP) estimations by incorporating ten key features to test the classifiers' stability by evaluating performance on separate data. Their results indicated that the models such as the logistic regression, random forest, and gradient boosting modeling generated more accurate results than the models based on the neural network approach incorporating various technicalities. 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. 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) resulted in significant advancements in classification accuracy.

4. Motivation for the Problem Undertaken

The project was the first provided to me by Flip Robo Technologies as a part of the internship programme. The exposure to real world data and the opportunity to deploy my skills in solving a real time problem has been the primary motivation. 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 ideas can make a difference, there is so much in the financial market to explore and analyze 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 the background a lot of mathematical framework work. In this project we have done a lot of data pre-processing & ML model building. In this section we dive into the mathematical background of some of these algorithms.

1. Logistic Regression

The response variable, label, is a binary variable (whether the loan was repaid or not). Therefore, logistic regression is a suitable technique to use because it is developed to predict a binary dependent variable as a function of the predictor variables. The logit, in this model, is the likelihood ratio that the dependent

variable, non-defaulter, is one (1) as opposed to zero (0), defaulter. The probability, P, of credit default is given by;

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

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 $\beta_0, \beta_1, \beta_2, \ldots, \beta_k$ are the regression (model) coefficients

2. Decision T

Decision Tre
supervised le
create a mo
decision rules

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

The application to decision trees arises from the fact that at each node, when considering a split on a given attribute, we have a probability distribution P with a component pj for each class j of the target variable Y. Hence, we see that a split on an attribute it most impure if P is uniform, and is pure if some pj=1, meaning all records that past this split are definitely of class j. Once we have an impurity function, we can define an impurity measure of a dataset D node n as so. If there are k possible values $y1, y2, \ldots, yk$ of the target variable Y, and σ is the selection operator from relational algebra then the probability distribution of S over the attribute Y is

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

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

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 X_i that has m possible values $v_1,..v_m$ 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.

Information gain is a splitting criterion that comes from information theory. It uses information entropy as the impurity function Given a probability distribution P= (p1, p2...pn), where pi is the probability that a point is in the subset Di of a dataset D, we define the entropy H:

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

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

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

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

3. Random Forest Classifier

The random forest classifier is an ensemble method algorithm of decision trees wherein each tree depends on randomly selected samples trained independently, with a similar distribution for all the trees in the forest. Hence, a random forest is a classifier incorporating a collection of tree-structured classifiers that decrease overfitting, resulting in an increase in the overall accuracy (Geurts et al. 2006). As such, random forest's accuracy differs based on the strength of each tree classifier and their dependencies

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

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. Hence, it differs from other tree-based ensemble approaches because it divides nodes by determining cut-points entirely at random, and it practices on the entire training sample to grow the trees. The practice of using the entire initial training samples instead of bootstrap replicas is to decrease bias. At each test node, each extra tree's algorithm is provided by the number of decision trees in the ensemble (denoted by M), the number of features randomly selected at each node (K), and the minimum number of instances needed to split a node (nmin). Hence, each decision tree must choose the best feature to split the data based on some criteria, leading to the final prediction by forming multiple decision trees.

Data Sources and their formats

The data set comes from my internship company – Fliprobo technologies in excel format.

```
# 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])
df.head()
```

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

There are 37 columns and 209593 rows in this dataset. The different features in dataset are as below:

➢ label : Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan {1:success, 0:failure}

➢ msisdn : mobile number of user

➢ aon : age on cellular network in days

➢ daily_decr30 : Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)

➢ daily_decr90 : Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)

➢ rental30 : Average main account balance over last 30 days

➢ rental90 : Average main account balance over last 90 days

➢ last_rech_date_ma : Number of days till last recharge of main account

➢ last_rech_date_da: Number of days till last recharge of data account

➢ last_rech_amt_ma : Amount of last recharge of main account (in Indonesian Rupiah)

➢ cnt_ma_rech30 : Number of times main account got recharged in last 30 days

➢ fr_ma_rech30 : Frequency of main account recharged in last 30 days

➢ sumamnt_ma_rech30 : Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)

➢ medianamnt_ma_rech30 : Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)

➢ medianmarechprebal30 : Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)

➢ cnt_ma_rech90 : Number of times main account got recharged in last 90 days

➢ fr_ma_rech90 : Frequency of main account recharged in last 90 days

➢ sumamnt_ma_rech90: Total amount of recharge in main account over last 90 days (in Indonesian Rupiah)

➢ medianamnt_ma_rech90: Median of amount of recharges done in main account over last 90 days at user level (in Indonesian Rupiah)

➢ medianmarechprebal90: Median of main account balance just before recharge in last 90 days at user level (in Indonesian Rupiah)

➢ cnt_da_rech30 : Number of times data account got recharged in last 30 days
➢ fr_da_rech30: Frequency of data account recharged in last 30 days

➢ cnt_da_rech90 : Number of times data account got recharged in last 90 days
➢ fr_da_rech90 : Frequency of data account recharged in last 90 days

➢ cnt_loans30 : Number of loans taken by user in last 30 days

➢ amnt_loans30 : Total amount of loans taken by user in last 30 days

➢ maxamnt_loans30 : maximum amount of loan taken by the user in last 30 days

➢ medianamnt_loans30 : Median of amounts of loan taken by the user in last 30 days

➢ cnt_loans90 : Number of loans taken by user in last 90 days

➢ amnt_loans90 : Total amount of loans taken by user in last 90 days

➢ maxamnt_loans90 : maximum amount of loan taken by the user in last 90 days

➢ medianamnt_loans90 : Median of amounts of loan taken by the user in last 90 days

➢ payback30: Average payback time in days over last 30 days

➢ payback90: Average payback time in days over last 90 days

➢ pcircle: telecom circle

➢pdate:date

```
# As we have 37 Columns Lets sort Columns by their datatype
df.columns.to_series().groupby(df.dtypes).groups
```

```
{int64: ['Unnamed: 0', 'label', 'last_rech_amt_ma', 'cnt_ma_rech30', 'cnt_ma_rech90', 'fr_ma_rech90', 'sumamnt_ma_rech90', 'cnt
_da_rech90', 'fr_da_rech90', 'cnt_loans30', 'amnt_loans30', 'amnt_loans90', 'maxamnt_loans90'], float64: ['aon', 'daily_decr3
0', 'daily_decr90', 'rental30', 'rental90', 'last_rech_date_ma', 'last_rech_date_da', 'fr_ma_rech30', 'sumamnt_ma_rech30', 'med
ianamnt_ma_rech30', 'medianmarechprebal30', 'medianamnt_ma_rech90', 'medianmarechprebal90', 'cnt_da_rech30', 'fr_da_rech30', 'm
axamnt_loans30', 'medianamnt_loans30', 'cnt_loans90', 'medianamnt_loans90', 'payback30', 'payback90'], object: ['msisdn', 'pcir
cle', 'pdate']}
```

The different datatypes of these features are as shown in above figure. Out of all features only three features with object datatypes and rest are int64. We can note here ‘pdate’ has data type of object instead of datetime datatype.

Data Pre-processing

The dataset is large and it may contain some data errors. In order to reach clean, error free data some preprocessing is done on data. At first integrity check is performed on data for presence of missing values, whitespaces. After that statistical matrix is plotted using df.describe() command to gain more insight about data.

• 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.shape
```

```
(209592, 37)
```

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

```
False
```

Statistical_Matrix –

From df.describe () command we got some key observations about data. One of it was that some features contain negative values and another observation few features contain extreme maximum value indicating possible outliers or invalid data.

Strategy to handle data error in min and max column

Assumption- All negative values are typing errors that happen accidentally by type - in front of original value (except feature depicting median). Corrective approach - Negative values are converted into absolute value to correct negative typing errors whenever applicable except features depicting median.

```
#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'])
```

Upper limit of these features is handled by outlier removal.

• Data error and correction in maxamnt_loans30 column

(maxamnt_loans30: maximum amount of loan taken by the user in last 30 days)

```
# maxamnt_loans30 : maximum amount of loan taken by the user in last 30 days
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
```

The maximum value in maxamnt_loans30 is not reliable. We already know the maximum loan amount taken by customers can be 0,5,10 and which can be repaid with an amount of 0,6,12.

Assumption - The maximum value in maxamnt_loans30 is 12.

We replaced values greater than 12 into the category of zero.

```
df.loc[(df['maxamnt_loans30'] != 6.0) & (df['maxamnt_loans30'] != 12.0)
       & (df['maxamnt_loans30']!=0.0), 'maxamnt_loans30']=0.0

# marking values greater than 12 and assign value zero to them.
```

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

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

• Feature Engineering on 'pdate' column

Simple feature engineering operation performed on ‘pdate’ to extract day, month and year columns. At last Unnamed :0, PCircle , msisdn columns are drop as they are unnecessary for further investigation

```
# Converting Date datatypes and spliting date into date, month and year.
df['pdate']=pd.to_datetime(df['pdate'])
df['Day']=df['pdate'].apply(lambda x:x.day)
df['Month']=df['pdate'].apply(lambda x:x.month)
df['Year']=df['pdate'].apply(lambda x:x.year)
df.head()
```

• Outliers Detection and removal

Outliers detected in boxplot. In order to remove outliers the Z-score method is employed but it results in huge data loss of 23.42 %, which we cannot afford. We got an observation from boxplot that outliers do not exist in the lower bound but outliers exist in the upper bound of features. Based on this observation we decided to employ quantile-based flooring- capping methods. Flooring is performed at 0th percentile for lower bound and capping performed at 99th percentile for upper bound.

```
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
```

```
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("\033[1m"+'Percentage Data Loss :'+"\033[0m",((209592-198174)/209592)*100,'%')
```

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

• Skewness in features & it's transformation

Considerable amount of skewness found in most features by skew () function. Power transformer from sklearn.preprocessing library used to transform skewness in features.

```
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']
```

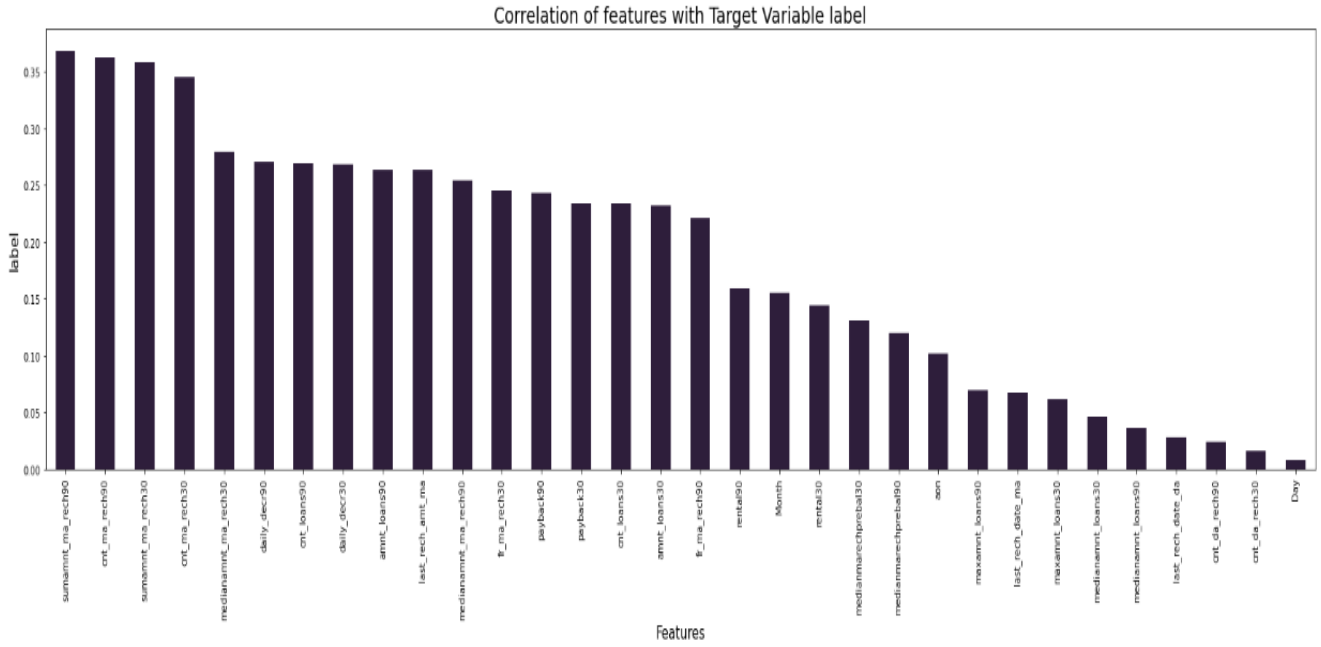
```
from sklearn.preprocessing import PowerTransformer
scaler = PowerTransformer(method='yeo-johnson')
```

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

For most of feature's skewness is reduced within the permissible limit except few ones.

Data Inputs- Logic- Output Relationships

To gain more insight about the relationship between input & output heatmap of correlation and bar plot of correlation of labels with independent features is plotted.



We can see that most independent features are poorly or moderately correlated with target variable labels. After that data is split into X and Y and data is scaled using standard scalar. The target variable label is imbalanced in nature, in order to resolve it SMOTE is applied to an oversample minority label class.

```
data.label.value_counts()
```

```
1    173461
0     24713
Name: label, dtype: int64
```

```
# Balancing using SMOTE
from imblearn.over_sampling import SMOTE
```

```
# 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 and now all the categories have the same data ensuring that the ML model does not get biased towards one category. The multicollinearity between features is checked using variance inflation factor. Few findings are as below:

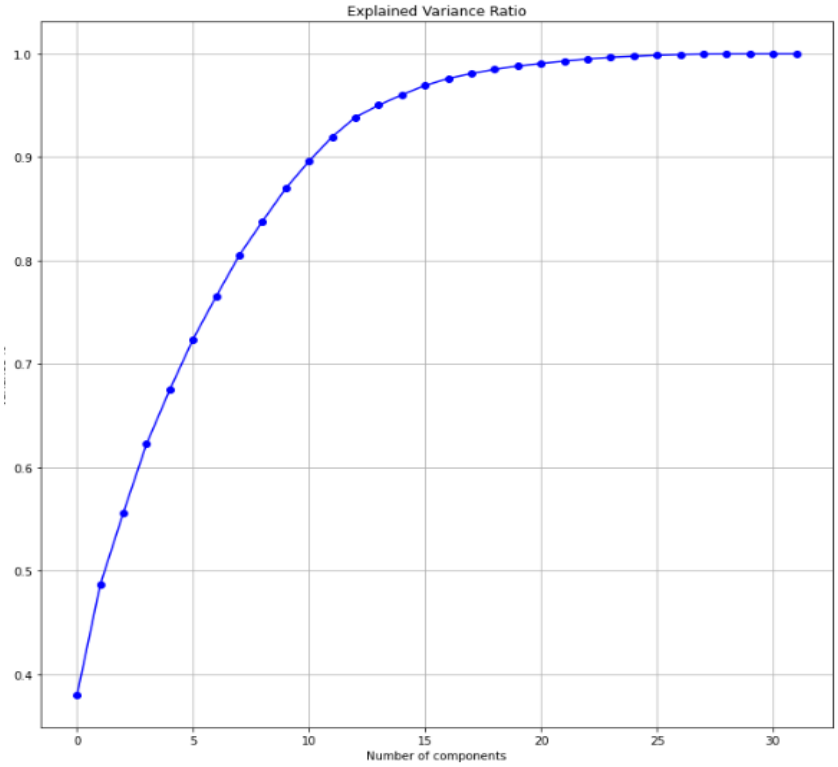
➢ daily_decr30 and daily_decr90 are highly correlated with each other.

➢ cnt_loans90 and amnt_loans90 are highly correlated with each other.

➢ cnt_loans30 and amnt_loans30 are highly correlated with each other.

➢ cnt_ma_rech30 and sumamnt_ma_rech30 are highly correlated with each other.

For most Independent features VIF exceeds the permissible limit of 20. PCA is applied to remove multicollinearity among features.

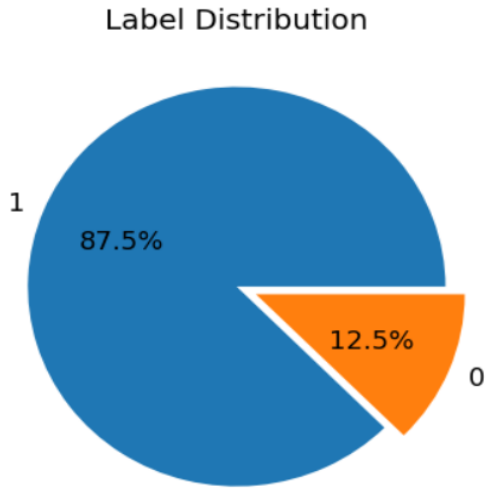


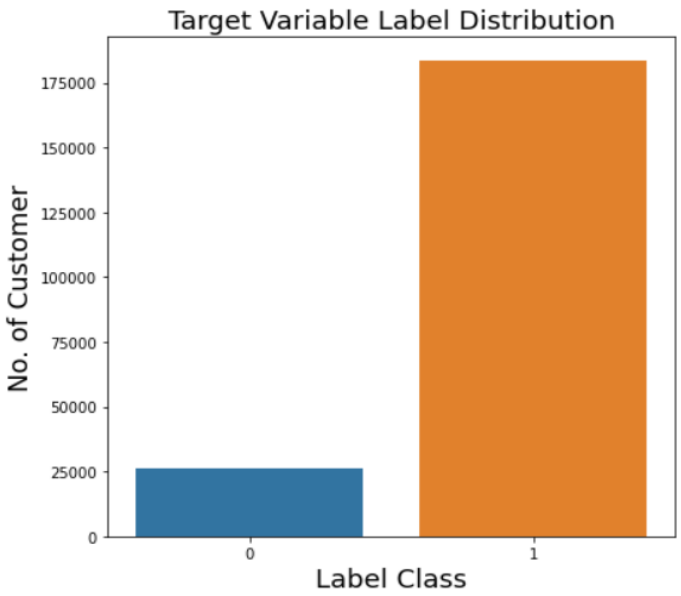

```
pca_new = PCA(n_components=20)
x_new = pca_new.fit_transform(X_scale)
```

```
principle_x=pd.DataFrame(x_new,columns=np.arange(20))
```

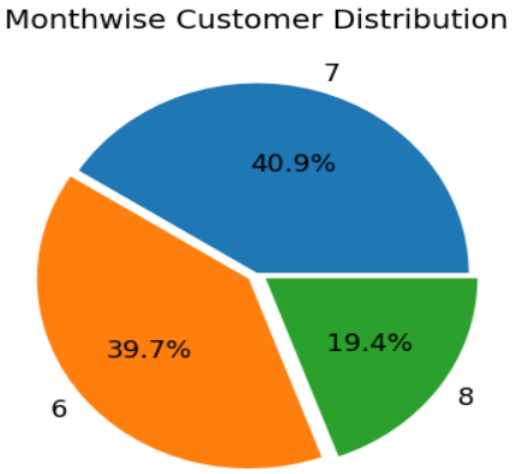
Exploratory Data Analysis

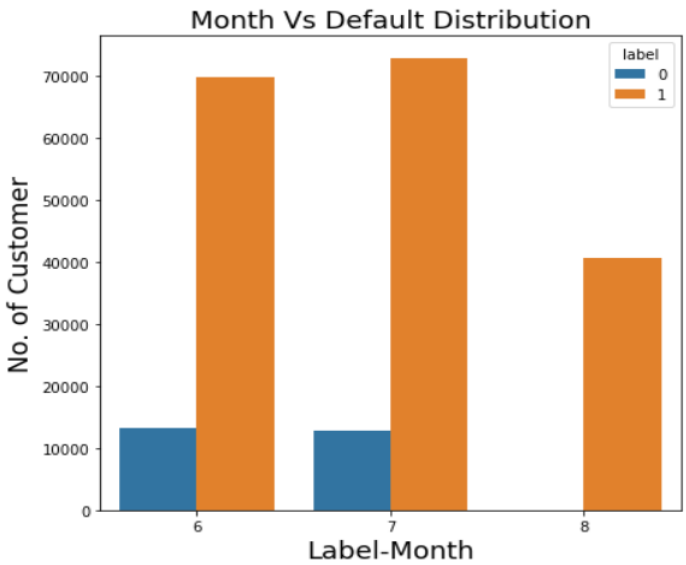
Lets see target variable distribution before balancing data



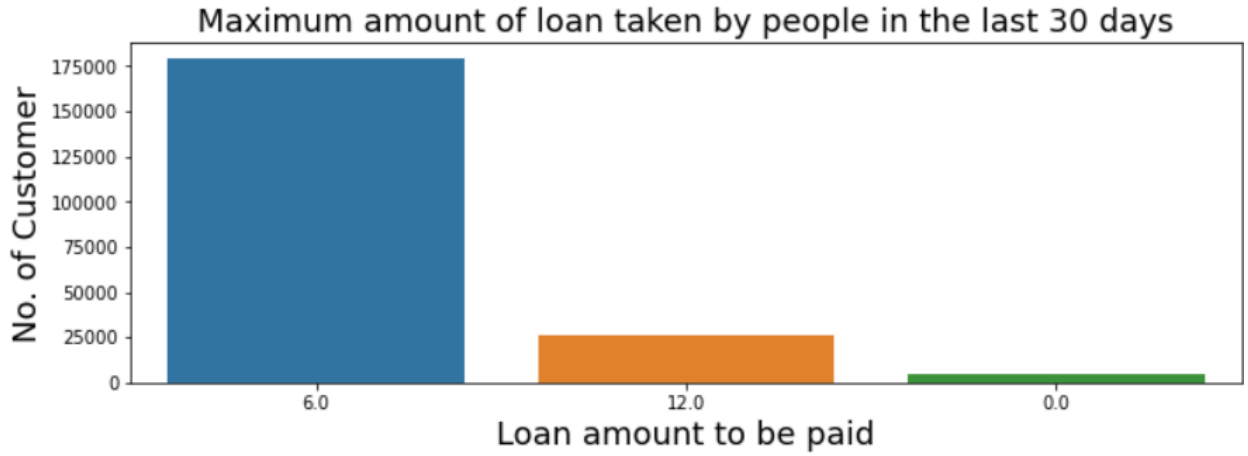


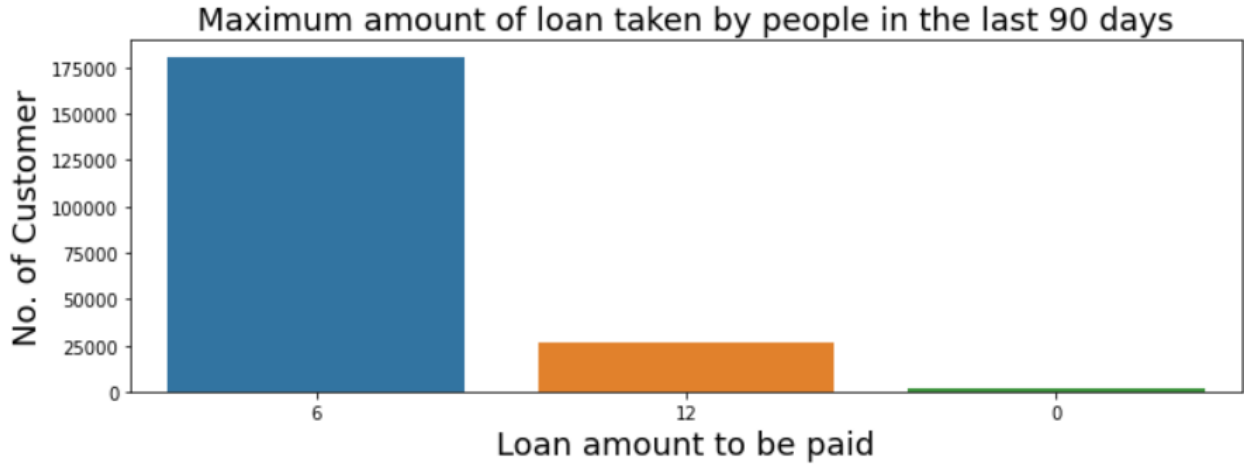
Here target variable Label class 1 represent non-defaulter while Label class 0 represent defaulter i.e., Loan not paid. We can see Most of the customers are non-defaulter while very few are defaulters. From ML model building point of view the target variable is imbalanced which needs to be balanced using balancing techniques.





Most of the data belong to month 6 and 7, followed by month 8. We can see very few defaulters in month 8.



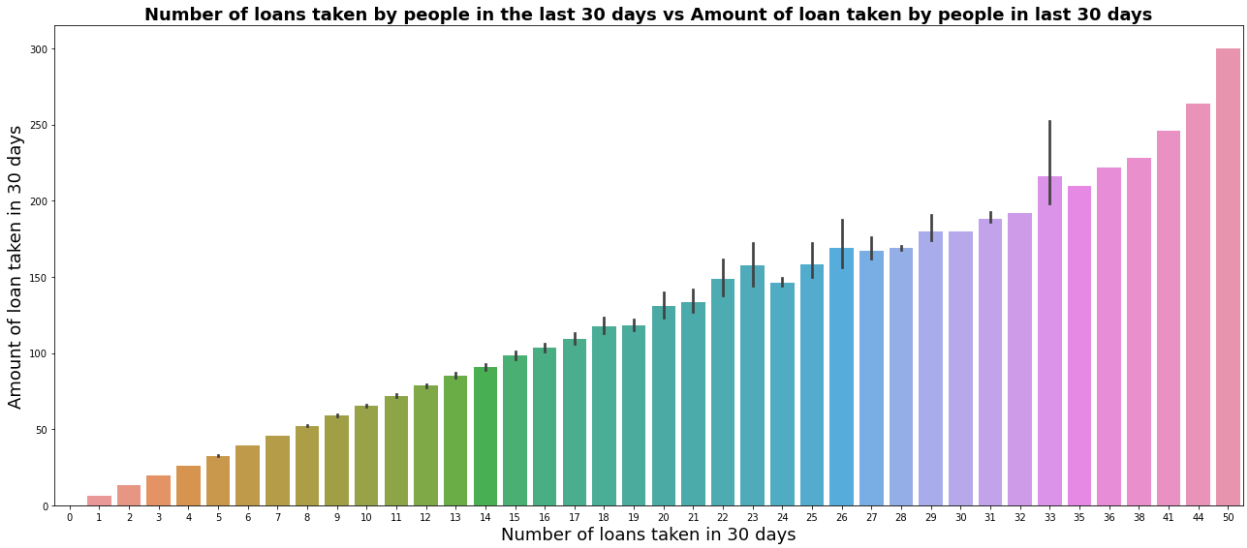


Observations:

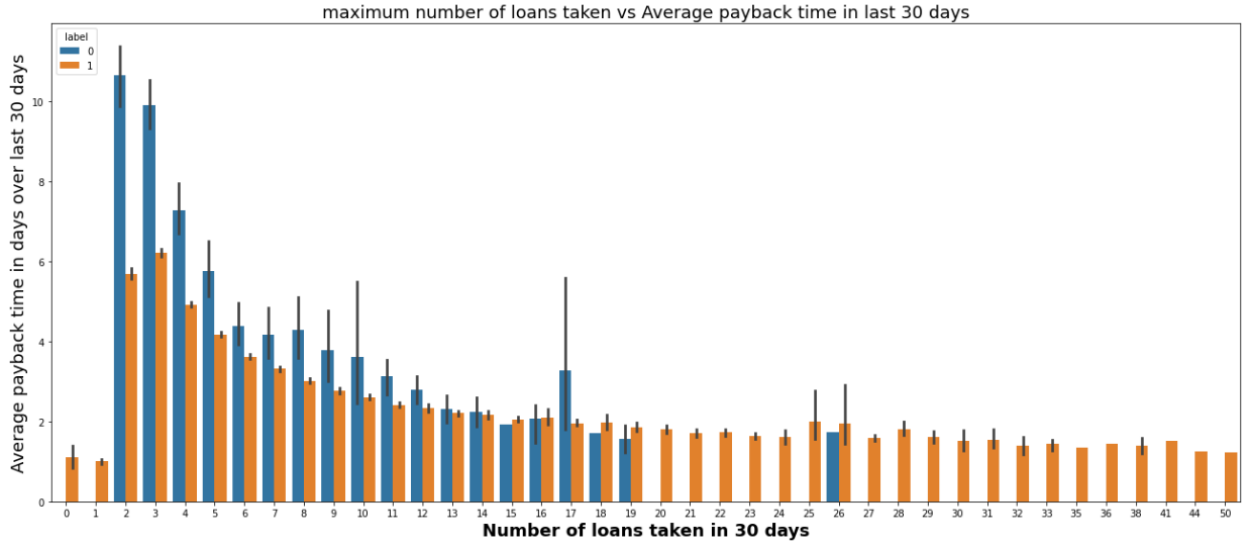
1. In 30 days, the maximum number of people had taken 6Rs as the loan amount and the number of people was 179192 whereas the number of people had not taken loan and their number is 4291.

2. In 90 days, the maximum number of people had taken 6Rs as the loan amount and the number of people was 180944 whereas the number of people who had not taken loan was 2043.

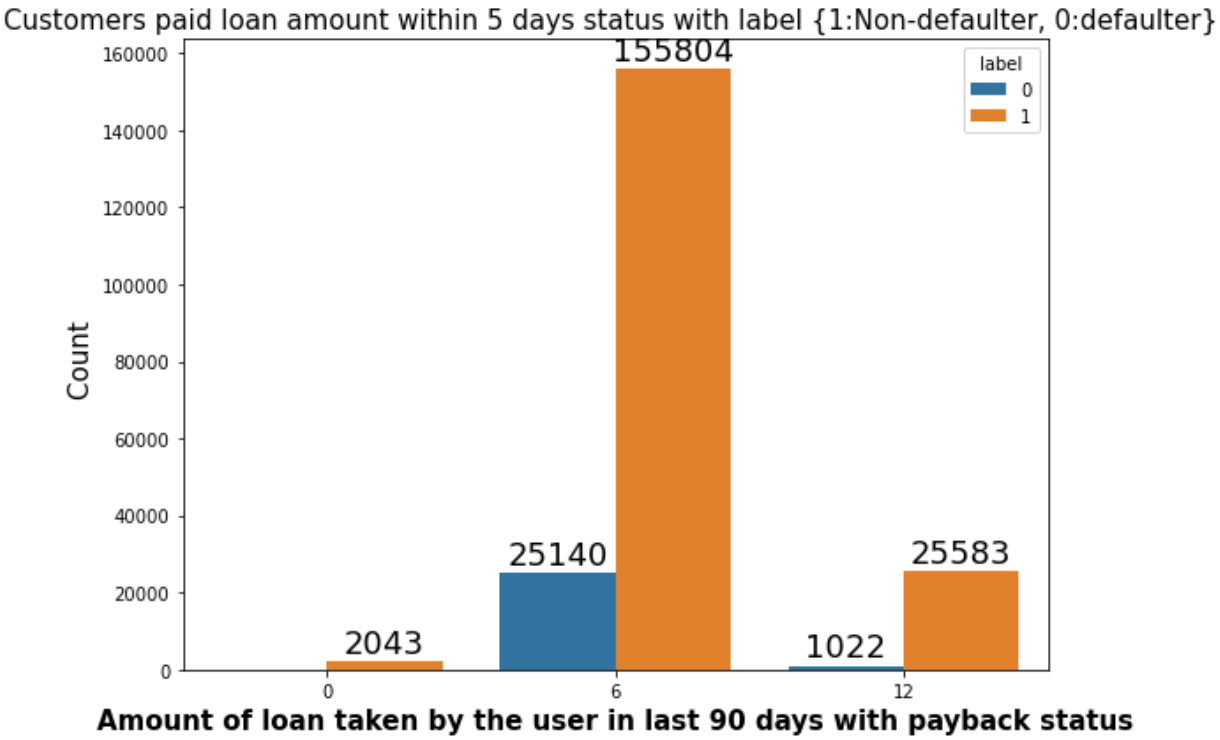
3. Maximum number of people had taken 12Rs as the loan amount within 90 days and their number is 26605 whereas for 30 days the number of people who had taken 12Rs is 26109 respectively



Maximum number of loans taken by the people is 50 and the Average loan amount is equivalent to 300. Minimum number of loans taken by the people is 0.



We can observe that the Average payback time over last 30 days is higher for people who had taken out loans 2 times.



Very few defaulters in the case of customers who have taken loans in amounts of 12.

Models Building & Evaluation

Identification Of Possible Problem Solving Approaches (Methods)

The target variable label has two classes i.e., label ’1’ indicates non default & 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

Key Metrics For Success In Solving Problem Under Consideration

▪ Precision can be seen as a measure of quality; higher precision means that an algorithm returns more relevant results than irrelevant ones.

▪ Recall is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.

▪ Accuracy score is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar.

▪ F1-score is used when the False Negatives and False Positives are crucial. While F1-score is a better metric when there are imbalanced classes.

▪ Cross validation Score: To run cross-validation on multiple metrics and also to return train scores, fit times and score times. Get predictions from each split of cross-validation for diagnostic purposes. Make a scorer from a performance metric or loss function.

▪ AUC_ROC _score: ROC curve. It is a plot of the false positive rate (xaxis)

versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0

▪ We have used Accuracy Score and Cross validation score as key parameter for model evaluation in this project since balancing of data is perform.

RUN AND EVALUATE SELECTED MODELS

LOGISTICS REGRESSION

```
X_train, X_test, Y_train, Y_test = train_test_split(principle_x, Y, random_state=156, test_size=.02)
log_reg=LogisticRegression()
log_reg.fit(X_train,Y_train)
y_pred=log_reg.predict(X_test)
print('\033[1m'+'Logistics Regression Evaluation'+'\033[0m')
print('\n')
print('\033[1m'+'Accuracy Score of Logistics Regression :'+'\033[0m', accuracy_score(Y_test, y_pred))
print('\n')
print('\033[1m'+'Confusion matrix of Logistics Regression :'+'\033[0m \n',confusion_matrix(Y_test, y_pred))
print('\n')
print('\033[1m'+'classification Report of Logistics Regression'+'\033[0m \n',classification_report(Y_test, y_pred))
```

```
Logistics Regression Evaluation


Accuracy Score of Logistics Regression : 0.7691309987029832


Confusion matrix of Logistics Regression :
 [[2711  751]
 [ 851 2626]]


classification Report of Logistics Regression
              precision    recall  f1-score   support

           0       0.76      0.78      0.77      3462
           1       0.78      0.76      0.77      3477

    accuracy                           0.77      6939
   macro avg       0.77      0.77      0.77      6939
weighted avg       0.77      0.77      0.77      6939
```

```
Cross Validation Score LogisticRegression() :

CVScore : [0.77100238 0.77375513 0.77268535 0.77434279 0.77490488]
Mean CV Score : 0.7733381045845947
Std deviation : 0.0013798400023616211
```

DECISION TREE CLASSIFIER

Decision Tree Classifier Evaluation

Accuracy Score of Decision Tree Classifier : 0.8845283562729697

Confusion matrix of Decision Tree Classifier :

```
 [[31217  3354]
 [ 4658 30156]]
```

classification Report of Decision Tree Classifier

	precision	recall	f1-score	support
0	0.87	0.90	0.89	34571
1	0.90	0.87	0.88	34814
accuracy			0.88	69385
macro avg	0.89	0.88	0.88	69385
weighted avg	0.89	0.88	0.88	69385

Cross Validation Score DecisionTreeClassifier() :

```
CVScore : [0.88338978 0.8850472  0.88458434 0.88720743 0.88497348]
Mean CV Score : 0.885040446105782
Std deviation : 0.0012355608337378315
```

RANDOM FOREST CLASSIFIER

Random Forest Classifier Evaluation

Accuracy Score of Random Forest Classifier : 0.9435036391150825

Confusion matrix of Random Forest Classifier :

```
 [[33095  1476]
 [ 2444 32370]]
```

classification Report of Random Forest Classifier

	precision	recall	f1-score	support
0	0.93	0.96	0.94	34571
1	0.96	0.93	0.94	34814
accuracy			0.94	69385
macro avg	0.94	0.94	0.94	69385
weighted avg	0.94	0.94	0.94	69385

Cross Validation Score RandomForestClassifier() :

CVScore : [0.94118325 0.94433955 0.94495849 0.94369019 0.94425228]
Mean CV Score : 0.9436847529032413
Std deviation : 0.0013138334817329321

EXTRA TREE CLASSIFIER

Extra Trees Classifier Evaluation

Accuracy Score of Extra Trees Classifier : 0.957007998847013

Confusion matrix of Extra Trees Classifier :

```
 [[33553  1018]
 [ 1965 32849]]
```

classification Report of Extra Trees Classifier

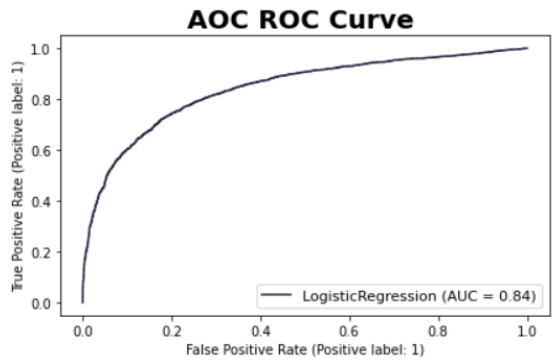
	precision	recall	f1-score	support
0	0.94	0.97	0.96	34571
1	0.97	0.94	0.96	34814
accuracy			0.96	69385
macro avg	0.96	0.96	0.96	69385
weighted avg	0.96	0.96	0.96	69385

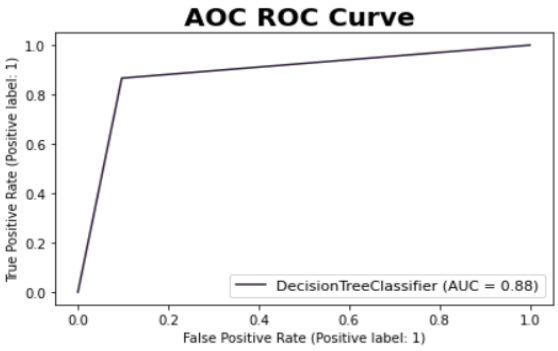
```
from sklearn.model_selection import cross_val_score
CVscore = cross_val_score(etc, principle_x, Y, cv =5)
print('\033[1m'+'Cross Validation Score', etc, ':'+'\033[0m\n')
print("CVScore :" ,CVscore)
print("Mean CV Score :",CVscore.mean())
print("Std deviation :",CVscore.std())
```

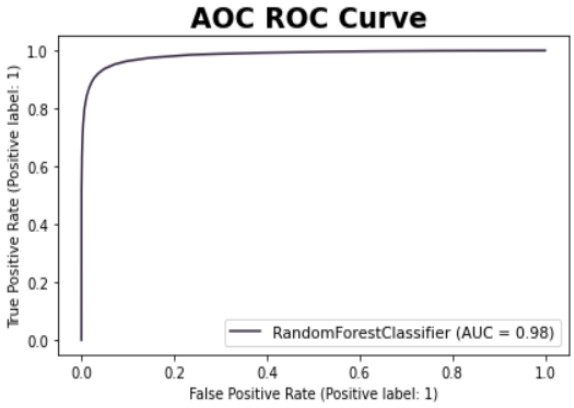
Cross Validation Score ExtraTreesClassifier() :

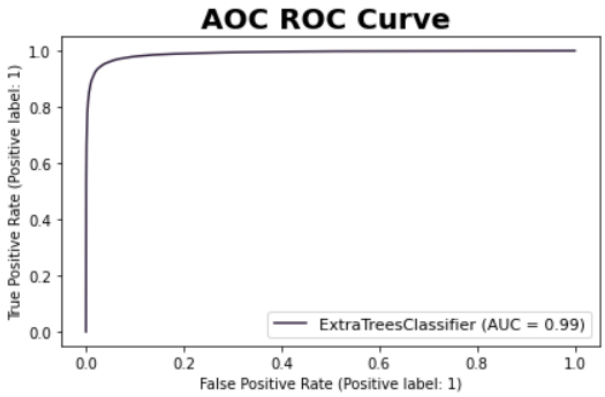
```
CVScore : [0.95640268 0.95821864 0.95811715 0.95689208 0.95826127]
Mean CV Score : 0.9575783620969796
Std deviation : 0.0007771502104052282
```

AOC -ROC CURVE OF DIFFERENT ML MODEL









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

Hyper Parameter Tuning

```
from sklearn.model_selection import GridSearchCV
```

```
parameter= {'criterion' : ['gini', 'entropy'],
            'max_features':['auto','sqrt','log2'] }
```

```
GCV = GridSearchCV(ExtraTreesClassifier(),parameter,verbose=10,n_jobs = -1)
GCV.fit(X_train,Y_train)
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits

GridSearchCV(estimator=ExtraTreesClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_features': ['auto', 'sqrt', 'log2']},
             verbose=10)
```

```
GCV.best_params_
```

```
{'criterion': 'entropy', 'max_features': 'auto'}
```

Extra Trees Classifier Evaluation

Accuracy Score of Extra Trees Classifier : 0.9570224111839735

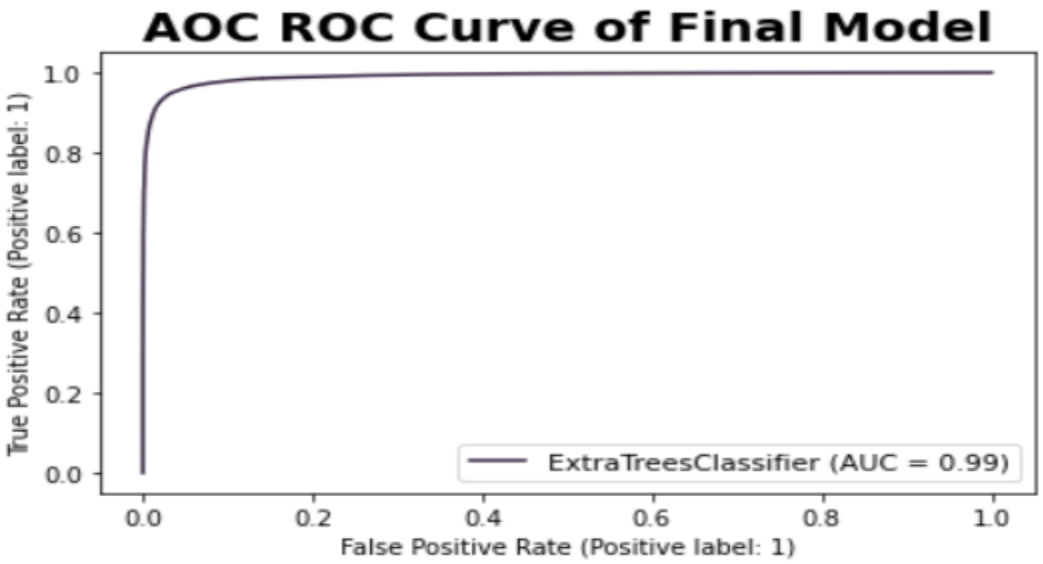
Confusion matrix of Extra Trees Classifier :

```
 [[33591   980]
 [ 2002 32812]]
```

classification Report of Extra Trees Classifier

	precision	recall	f1-score	support
0	0.94	0.97	0.96	34571
1	0.97	0.94	0.96	34814
accuracy			0.96	69385
macro avg	0.96	0.96	0.96	69385
weighted avg	0.96	0.96	0.96	69385

Hyper parameter tuning leads to sight increase in accuracy from 0.9570 to 0.9570. This will be over the final model.



Final model is saved using the joblib library.

```
import joblib
joblib.dump(etc,'Micro_Credit_Defaulter.pkl')
```

```
['Micro_Credit_Defaulter.pkl']
```

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 sometimes 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 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

1. Extra Tree Classifier Hyper parameter tuned gives maximum accuracy score of 0.9570 with cross validation score of 0.9555. It also gives us a maximum AUC score.

2. 12.5 % customers are defaulters out of the whole dataset.

3. Tendency to pay loan within 5 days is high among customers who take loans many times within a month compared to those who take loans 1-2 times.

4. It is recommended to extend the loan repayment time frame from 5 days to 7 days.

Learning Outcomes of the Study in respect of Data Science

1. First time I handle such a huge dataset.

2. First time any project I worked on ever needed such data clean operation. I paid attention to realistic & unrealistic data, considering corrective measures taken as per need. This was beyond normal missing value imputation for me.

3. As data was huge and required high computational capacity, it took a huge amount of time for the Hyper parameter tuning process

4. I run Hyper parameter tuning 2-3 times with several parameters. It was taking a lot of time so at the end I reduced the Hyperparameter search parameter and still it took 12 - 15 hr to find the best parameter.

Limitations of this work and Scope for Future Work

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