MICRO CREDIT DEFAULTER PROJECT

Submitted by:

HARSH NEMA

ACKNOWLEDGMENT

I would like to express my profound gratitude to the **Flip Robo team**, who has given me this opportunity to deal with a beautiful dataset and it has helped me to improve my analytical skills. I would like to express my special thanks to our mentor **Mr. Shwetank Mishra Sir** (SME Flip Robo) for their contributions to the completion of my project titled **'Micro Credit Defaulter Project'**.

I am eternally grateful to **"Datatrained"** for giving me the opportunity for an internship at Flip Robo. Last but not least I have to thank my parents and wife for their love and support during my project.

References:

1. SCIKIT Learn Library Documentation
2. Datatrained recorded videos for Data Science Course
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 Credit Default among Micro Borrowers in Ghana Kwame Simpe Ofori, Eli Fianu
6. Predicting Microfinance Credit Default: A Study of Nsoatreman Rural Bank, Ghana Ernest Yeboah Boateng
7. A Machine Learning Approach for Micro-Credit Scoring Apostolos Ampountolas

1. 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 to deliver 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 a 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. 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 a 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

"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 several different operators focusing on the financially under-served people to satisfy their need for poverty alleviation, social promotion, emancipation, and inclusion. Microfinance institutions reach and serve their target market in very innovative ways (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). The global asset size of Microfinance institutions is estimated to be over US$70 billion (Mix Market). The independent microfinance think-tank housed by the World Bank, Consultative Group to Assist the Poor (CGAP) and Swiss manager and adviser Symbiotics estimate the global asset size at US$7 billion (Milana, 2012). Microfinance is the best-known, most developed sector of the so-called 'impact investing' world of credit suppliers (Grene 2012).

- **Motivation for the Problem Undertaken**

The project is provided to me by Flip Robo Technologies as a part of the internship program. The exposure to real-world data and the opportunity to deploy my skillset in solving a real-time problem has been my primary motivation.

This project includes the real-time problem for Microfinance Institution (MFI), and it is related to financial sectors, as I believe that growing technologies and Idea can make a difference, there are 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 helps our client in further investment and improvement in the 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 the loan.

2. Analytical Problem Framing

- **Mathematical/ Analytical Modeling of the Problem**

Whenever we employ any ML algorithm, statistical models, or feature pre-processing in the background 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 binary (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 $\beta_o, \beta_1, \beta_2, \ldots, \beta_k$ are the regression (model) coefficients

2. Decision Tree Classifier

Decision Trees (DTs) are a non-parametric (fixed number of parameters) supervised learning method used for classification and regression. The goal is to create a model that predicts the label of a target variable by learning simple decision rules inferred from the data features.

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 is most impure if P is uniform, and is pure if some pj=1, meaning all records that pass 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 $y1$, $y2$, …, 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 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.

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 $InformationGainY\ (Xi, D)$:

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

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

3. Random Forest Classifier

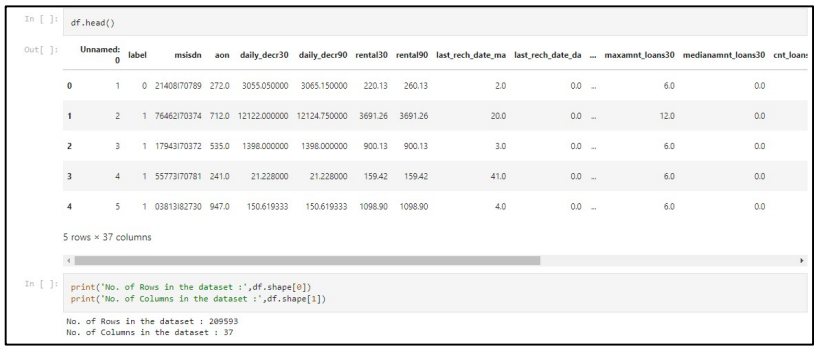
The random forest classifier is an ensemble method algorithm of decision trees wherein each tree depends on randomly selected samples trained independently, with a similar distribution for all the trees in the forest. Hence, a random forest is a classifier incorporating a collection of tree-structured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

- **Data Sources and their formats**

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



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

Out[ ]:

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30	medianamnt_loans30	cnt_loans
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	...	6.0	0.0	
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	...	12.0	0.0	
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	...	6.0	0.0	
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	...	6.0	0.0	
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	...	6.0	0.0	

5 rows × 37 columns

```
In [ ]: print('No. of Rows in the dataset :',df.shape[0])
        print('No. of Columns in the dataset :',df.shape[1])
```

```
No. of Rows in the dataset : 209593
No. of Columns in the dataset : 37
```

There are 37 columns and 209593 rows in this dataset. The different features in the 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 a cellular network in days
- daily_decr30: Daily amount spent from the main account, averaged over the last 30 days (in Indonesian Rupiah)
- daily_decr90: Daily amount spent from the main account, averaged over the last 90 days (in Indonesian Rupiah)
- rental30: Average main account balance over the last 30 days
- rental90: Average main account balance over last 90 days
- last_rech_date_ma: Number of days till the last recharge of the main account
- last_rech_date_da: Number of days till the last recharge of data account
- last_rech_amt_ma: Amount of the last recharge of the main account (in Indonesian Rupiah)
- cnt_ma_rech30: Number of times main account got recharged in last 30 days
- fr_ma_rech30: Frequency of the main account recharged in the last 30 days
- sumamnt_ma_rech30: Total amount of recharge in the main account over last 30 days (in Indonesian Rupiah)
- medianamnt_ma_rech30: Median amount of recharges done in the main account over the last 30 days at the user level (in Indonesian Rupiah)

- medianmarechprebal30: Median of main account balance just before recharge in the last 30 days at the user level (in Indonesian Rupiah)
- cnt_ma_rech90: Number of times main account got recharged in last 90 days
- fr_ma_rech90: Frequency of the main account recharged in the last 90 days
- sumamnt_ma_rech90: Total amount of recharge in the main account over the last 90 days (in Indonesian Rupiah)
- medianamnt_ma_rech90: Median amount of recharges done in the main account over the last 90 days at the user level (in Indonesian Rupiah)
- medianmarechprebal90: Median of main account balance just before recharge in the last 90 days at the 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 the user in the last 30 days
- amnt_loans30: Total amount of loans taken by the user in the last 30 days
- maxamnt_loans30: maximum amount of loan taken by the user in the last 30 days
- medianamnt_loans30: Median amounts of loans taken by the user in the last 30 days
- cnt_loans90: Number of loans taken by the user in the last 90 days
- amnt_loans90: Total amount of loans taken by the user in the last 90 days
- maxamnt_loans90: maximum amount of loan taken by the user in the last 90 days
- medianamnt_loans90: Median amounts of loans taken by the user in the last 90 days
- payback30: Average payback time in days over the last 30 days

- payback90: Average payback time in days over the last 90 days
- pcircle: telecom circle
- pdate: date

```
In [ ]: # Lets sort columns by their datatype
        df.columns.to_series().groupby(df.dtypes).groups

Out[ ]: {int64: ['label', 'last_rech_amt_ma', 'cnt_ma_rech30', 'cnt_ma_rech90', 'fr_ma_rech90', 'sumamnt_ma_rech90', 'cnt_da_rech90', 'fr_da_rech90', 'cnt_loa
        ns30', 'amnt_loans30', 'amnt_loans90', 'maxamnt_loans90'], float64: ['aon', 'daily_decr30', 'daily_decr90', 'rental30', 'rental90', 'last_rech_date_m
        a', 'last_rech_date_da', 'fr_ma_rech30', 'sumamnt_ma_rech30', 'medianamnt_ma_rech30', 'medianmarechprebal30', 'medianamnt_ma_rech90', 'medianmarechpre
        bal90', 'cnt_da_rech30', 'fr_da_rech30', 'maxamnt_loans30', 'medianamnt_loans30', 'cnt_loans90', 'medianamnt_loans90', 'payback30', 'payback90'], obje
        ct: ['msisdn', 'pcircle', 'pdate']}
```

- **<u>Data Preprocessing Done</u>**

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

a) Missing value check – Data contain no missing value

b) Data integrity check –

```
In [ ]: df.isin([' ','?','-','null','NA']).sum().any()

Out[ ]: False

In [ ]: df.duplicated().sum()

Out[ ]: 0

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

<u>Statistical Matrix</u>

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

• **Strategy to handle data error in min and max columns -**

Assumption 1- All negative values are typing errors that happen accidentally by typing - in front of the original value (except the feature depicting the median).

Corrective approach - Negative values are converted into absolute values to correct negative typing errors whenever applicable except feature depicting median

```
In [ ]: #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'])

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

The upper limit of these features will be 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)

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

In [ ]: df['maxamnt_loans30'].value_counts()

Out[ ]: 6.0     179193
        12.0     26109
        0.0       4291
        Name: maxamnt_loans30, dtype: int64
```

Assumption 2 - The maximum value in maxamnt_loans30 is 12

- **Feature Engineering on 'pdate' column:** Simple feature engineering operation performs on ‘pdate’ to extract a day, month, and year column. At last Unnamed :0, PCircle, msisdn columns are dropped as they are unnecessary for further investigation

```
In [ ]:
df['pdate'] = pd.to_datetime(df['pdate'])
df['pday'] = df['pdate'].dt.day
df['pmonth'] = df['pdate'].dt.month
df['pyear'] = df['pdate'].dt.year
```

- **<u>Outliers Detection and removal –</u>**

  Outliers detected in a boxplot. To remove outliers Z-score method is employed but it results in a huge data loss of 23.42 %, which we cannot afford. We got observation from the 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 the quantile-based flooring-capping method. Flooring is performed at the 0th percentile for the lower bound and capping perform at the 99th percentile for the upper bound.

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

```
label                      1.000000
aon                     2419.080000
daily_decr30           41730.440000
daily_decr90           49967.383600
rental30               19465.962000
rental90               26997.968000
last_rech_date_ma         57.000000
last_rech_date_da         56.000000
last_rech_amt_ma       10000.000000
cnt_ma_rech30             20.000000
fr_ma_rech30              26.000000
sumamnt_ma_rech30      46857.440000
medianamnt_ma_rech30   10000.000000
medianmarechprebal30    1531.540000
cnt_ma_rech90             33.000000
fr_ma_rech90              54.000000
sumamnt_ma_rech90      78717.240000
medianamnt_ma_rech90   10000.000000
medianmarechprebal90    1040.196400
cnt_da_rech30              1.000000
fr_da_rech30               0.000000
cnt_da_rech90              1.000000
fr_da_rech90               0.000000
cnt_loans30               12.000000
amnt_loans30              84.000000
maxamnt_loans30           12.000000
medianamnt_loans30         1.000000
cnt_loans90               23.000000
amnt_loans90             132.000000
maxamnt_loans90           12.000000
medianamnt_loans90         1.000000
payback30                 38.666667
payback90                 49.770000
pday                      30.000000
pmonth                     2.000000
dtype: float64
```

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

```
(198175, 35)
```

```
In [ ]:
Data_loss = (df.shape[0] - data.shape[0])/df.shape[0]*100
Data_loss
```

```
Out[ ]: 5.44770102054935
```

Skewness Reduction

- **Skewness in features & its transformation**

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

Skewness Reduction

In [ ]:
```
# Considering skewness reduction through PowerTransformer

from sklearn.preprocessing import PowerTransformer
pt = PowerTransformer()
```

In [ ]:
```
# Seperating features and target variable for classification task

X = data.drop('label',axis=1)

y = data['label']
```

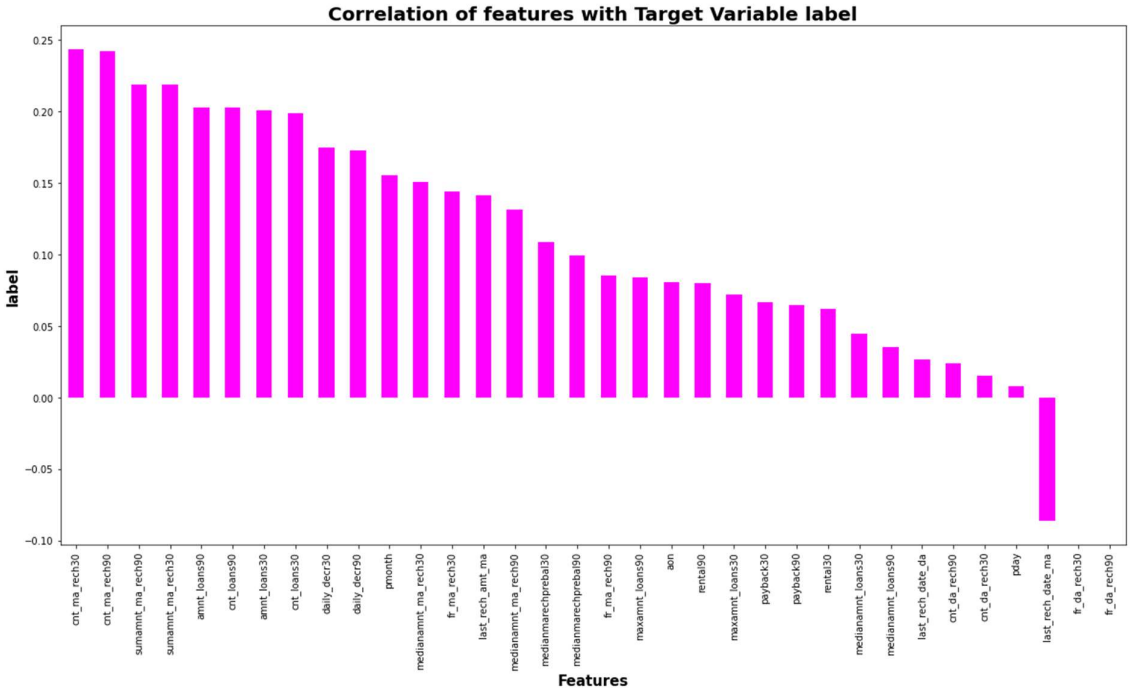
In [ ]:
```
# Applying Power transformer

X_new_pt = pt.fit_transform(X)

# Converting numpy array(X_new_pt) into Dataframe and reassigning the values

X = pd.DataFrame(X_new_pt,columns= X.columns)

# Checking Skewness

X.skew().sort_values(ascending=False)
```

Out[ ]:
```
cnt_da_rech30           9.681414
cnt_da_rech90           6.688841
last_rech_date_da       6.421032
medianamnt_loans90      3.782971
medianamnt_loans30      3.452258
medianmarechprebal90    1.464526
medianmarechprebal30    1.286926
maxamnt_loans30         0.427389
maxamnt_loans90         0.367233
payback30               0.296428
payback90               0.208659
fr_ma_rech90            0.143565
fr_ma_rech30            0.136361
cnt_loans90             0.097593
pmonth                  0.079280
```

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

- **Data Inputs- Logic- Output Relationships**

To gain more insight into the relationship between input & output, a heatmap of correlation and a bar plot of the correlation of label with independent features is plotted.



We can see that most of the independent features are poorly or moderately correlated with the target variable 'label'. After that data is split into X and Y and data is scaled using a standard scalar.

The target variable label is **imbalanced**, to resolve it **SMOTE** is applied to oversample the minority label class:

BALANCING TARGET VARIABLE USING SMOTE

```
[ ]  1 from imblearn.over_sampling import SMOTE
     2 over_smp = SMOTE(0.8)
```

```
[ ]  1 print("The number of target classes before fit{}".format(Counter(y)))
```

```
The number of target classes before fitCounter({1: 173462, 0: 24713})
```

Clearly visible the data is imbalanced

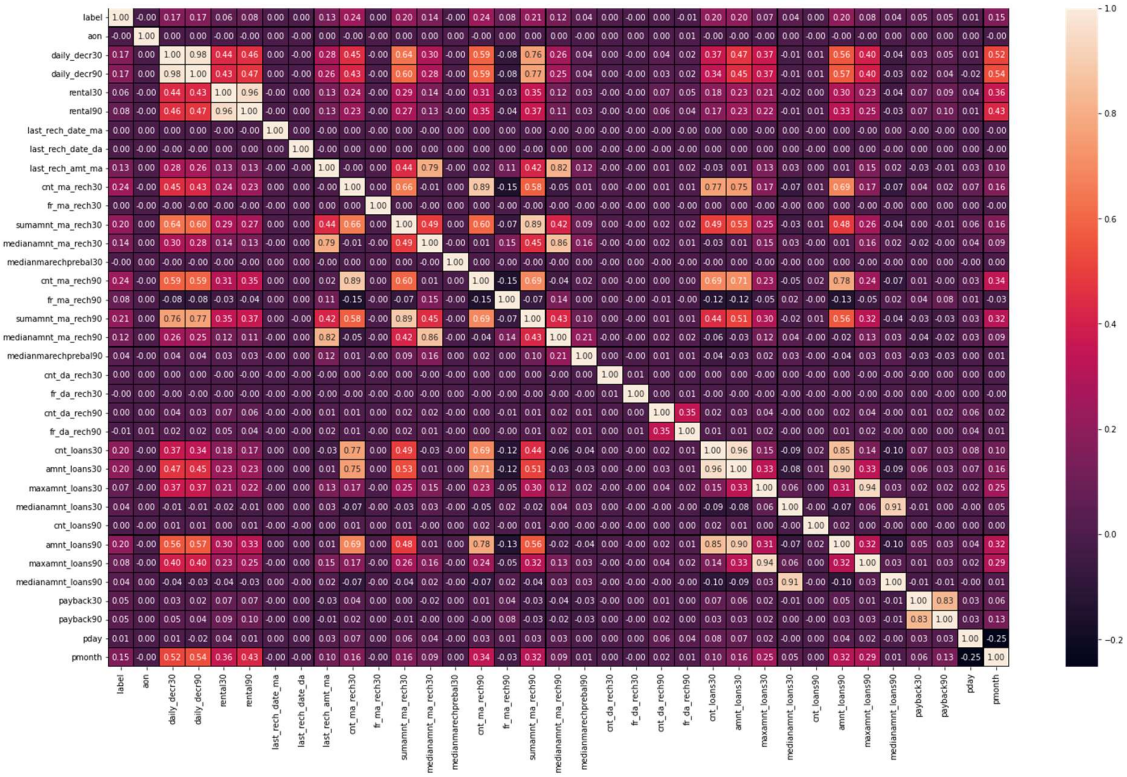
```
[ ]  1 X,y = over_smp.fit_resample(X,y)
     2 print("The number of target classes after fit{}".format(Counter(y)))
```

```
The number of target classes after fitCounter({1: 173462, 0: 138769})
```

Standard Scaling using Standard Scaler

```
[ ]  1 from sklearn.preprocessing import StandardScaler
     2 scaler= StandardScaler()
     3 X_scaler = scaler.fit_transform(X)
```

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 was checked using the variance inflation factor.



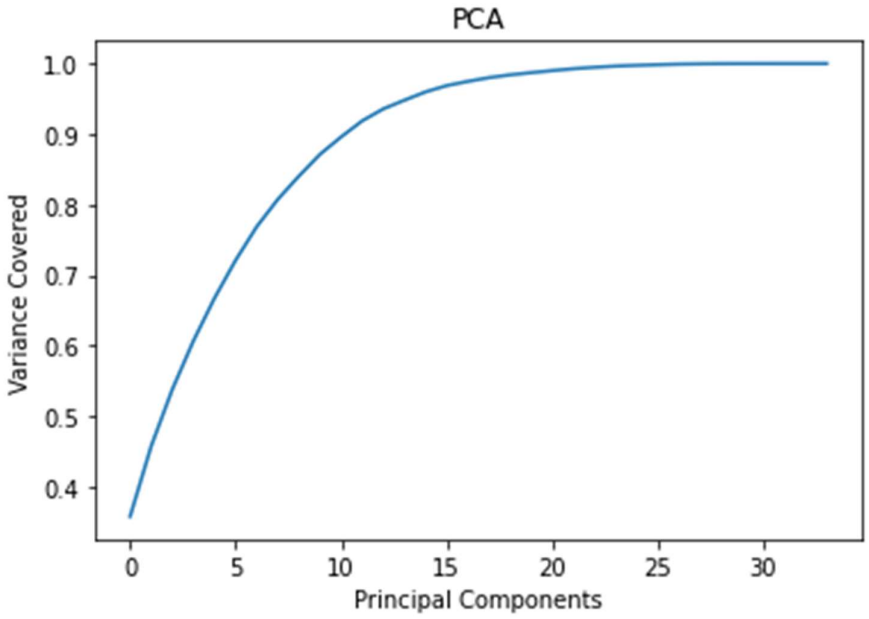
```
[ ]  1 # Checking multicollinearity in the dataset
     2 from statsmodels.stats.outliers_influence import variance_inflation_factor
     3 vif = pd.DataFrame()
     4 vif['vif'] = [variance_inflation_factor(X_scaler,i) for i in range(X.shape[1])]
     5 vif['Features'] = X.columns
     6
     7 vif
```

	vif	Features
0	1.047947	aon
1	644.191195	daily_decr30
2	707.032334	daily_decr90
3	42.018464	rental30
4	46.117698	rental90
5	2.743467	last_rech_date_ma
6	17.272265	last_rech_date_da
7	11.286203	last_rech_amt_ma
8	73.008393	cnt_ma_rech30
9	2.480985	fr_ma_rech30
10	133.074384	sumamnt_ma_rech30
11	26.728624	medianamnt_ma_rech30
12	3.556723	medianmarechprebal30
13	84.496895	cnt_ma_rech90

For most Independent features VIF exceeds the permissible limit of 10. PCA is applied to remove multicollinearity among features

```
1 from sklearn.decomposition import PCA
2 pca = PCA()
3 pca.fit_transform(X_scaler)
```

```
array([[ 1.81620442e+00, -6.15970790e-01, -2.67606744e-01, ...,
         3.57323781e-03, -6.13018202e-17, -5.39178898e-17],
       [ 1.90557073e+00,  2.48470984e+00, -1.71187254e-01, ...,
        -1.27856138e-02,  1.40055072e-16, -1.54140753e-16],
       [-1.13283576e+00,  1.54882933e+00, -1.60886245e-01, ...,
        -1.80970772e-02,  4.08960060e-18, -2.85616658e-17],
       ...,
       [-2.38947883e+00,  1.23133491e+00, -1.32452935e-01, ...,
        -3.12726281e-04,  2.56701587e-18, -8.32782559e-19],
       [-1.19963871e-01,  6.82870659e-01, -2.92113067e-01, ...,
         1.17192955e-01, -3.49234592e-18, -5.75417410e-18],
       [-3.63729171e+00,  6.13650128e-01, -2.08361784e-01, ...,
        -1.78578396e-02,  2.27059891e-18,  1.80798008e-18]])
```



We can see that 13 principal components attribute 95% of the variation in the data. We shall pick the first 13 components for our prediction

```
1 pca = PCA(n_components=13)
2 new_pcomp = pca.fit_transform(X_scaler)
3 princ_comp = pd.DataFrame(new_pcomp,columns=['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10','PC11','PC12','PC13'])
4 princ_comp
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13
0	1.816204	-0.615971	-0.267607	-1.084234	-0.771625	-1.235938	0.208227	-1.020198	-2.268716	0.464777	1.523386	0.812294	1.352166
1	1.905571	2.484710	-0.171187	3.587108	0.857446	2.244833	-2.762230	-0.417684	0.443507	-0.665439	-0.056510	0.681180	1.067776
2	-1.132836	1.548829	-0.160886	1.661554	-0.519040	0.232766	0.145651	-0.008749	0.718595	-0.284196	0.987148	-0.421634	0.856240
3	-2.679962	-0.258448	-0.204293	-0.498226	-0.053914	0.469793	-0.943817	-0.871963	1.422109	-0.166237	0.152609	1.648212	-1.859436
4	3.582822	-1.225642	-0.462003	-2.577383	-0.329176	0.064347	-0.697698	0.910816	1.078825	-0.782981	-0.218619	-0.472673	-0.291357
...	...	...	...	...	...	...	...	...	...	...	...	...	...
312226	-2.766814	0.987978	-0.421097	-1.287725	-0.112273	0.575952	-1.284603	-0.230926	0.865312	0.292764	0.171533	-0.216132	0.712653
312227	2.721696	-2.406109	-0.268197	-2.540795	-0.145482	0.093193	0.068832	1.205784	0.654848	0.155214	0.401656	-0.473738	-0.463216
312228	-2.389479	1.231335	-0.132453	1.313256	-0.539468	-0.002197	-0.696277	-0.242992	0.962842	0.539324	2.116016	1.231443	-0.955352
312229	-0.119964	0.682871	-0.292113	-1.528871	0.015036	0.764212	0.445209	-0.499940	-0.666536	-1.541603	1.451165	0.977042	-0.369316
312230	-3.637292	0.613650	-0.208362	-0.431135	-0.289287	0.365823	-1.139412	1.264336	0.776116	0.810908	1.057822	1.771844	-1.450423

312231 rows × 13 columns

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

➢ All negative values are assumed as a typing error that happened accidentally by typing '-' in front of the original value (except the feature depicting the median)

➢ The maximum value in maxamnt_loans30 is 12

- **Hardware and Software Requirements and Tools Used**

Hardware Used -

1. Processor — Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz 3.60 GHz

2. RAM — 16.0 GB

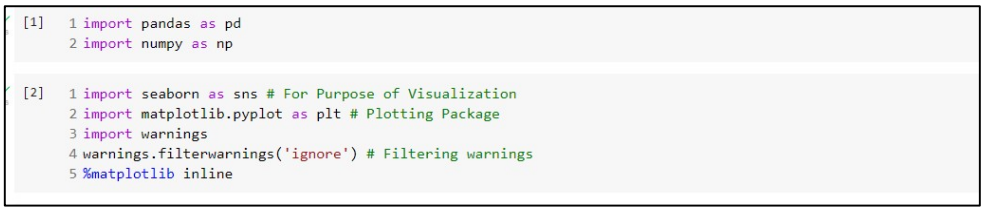
3. GPU — 2GB AMD Radeon Graphics card

Software utilised -

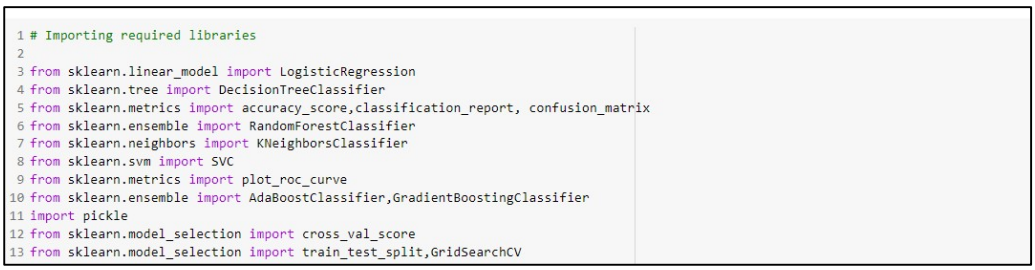
1. Google Colab Notebook - to write and execute arbitrary python code through the browser

2. Microsoft office – for making project report and ppt

Libraries Used – General libraries used for data wrangling



```
[1]  1 import pandas as pd
     2 import numpy as np

[2]  1 import seaborn as sns # For Purpose of Visualization
     2 import matplotlib.pyplot as plt # Plotting Package
     3 import warnings
     4 warnings.filterwarnings('ignore') # Filtering warnings
     5 %matplotlib inline
```



```
# Importing required libraries

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,classification_report, confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import plot_roc_curve
from sklearn.ensemble import AdaBoostClassifier,GradientBoostingClassifier
import pickle
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split,GridSearchCV
```

3. 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 the customer is a defaulter or not. This becomes a binary classification problem that can be solved using various classification algorithms. To gain high accuracy of the model we will train a model with different classification models and select the final model among them. To enhance the performance of the best model will employ hyperparameter tuning over it.

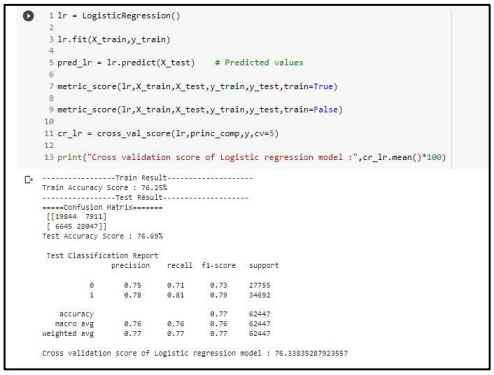
- **Testing of Identified Approaches (Algorithms)**

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

- ✓ Logistic Regression
- ✓ KNeighbors Classifier
- ✓ Decision Tree Classifier
- ✓ Random Forest Classifier
- ✓ Support Vector Classifier

- **Run and Evaluate selected models**

1. LogisticRegression Model



```
lr = LogisticRegression()

lr.fit(X_train,y_train)

pred_lr = lr.predict(X_test)    # Predicted values

metric_score(lr,X_train,X_test,y_train,y_test,train=True)

metric_score(lr,X_train,X_test,y_train,y_test,train=False)

cr_lr = cross_val_score(lr,princ_comp,y,cv=5)

print("Cross validation score of Logistic regression model :",cr_lr.mean()*100)
```

```
----------Train Result----------
Train Accuracy Score : 76.25%
----------Test Result----------
=====Confusion Matrix======
 [[19844  7911]
 [ 6645 22047]]
Test Accuracy Score : 76.69%

 Test Classification Report
              precision    recall  f1-score   support

           0       0.75      0.71      0.73     27755
           1       0.78      0.81      0.79     34692

    accuracy                           0.77     62447
   macro avg       0.76      0.76      0.76     62447
weighted avg       0.77      0.77      0.77     62447

Cross validation score of Logistic regression model : 76.33835287923557
```

2. KNN Classifier

Model 2 : KNN Classifier

```
[ ]  1 knn = KNeighborsClassifier()
     2
     3 knn.fit(X_train,y_train)
     4
     5 pred_knn = knn.predict(X_test)    # Predicted values
     6
     7 metric_score(knn,X_train,X_test,y_train,y_test,train=True)
     8
     9 metric_score(knn,X_train,X_test,y_train,y_test,train=False)
    10
    11 cr_knn = cross_val_score(knn,princ_comp,y,cv=5)
    12
    13 print("Cross validation score of K Neighbors Classifier model :",cr_knn.mean()*100)
```

```
----------Train Result----------
Train Accuracy Score : 92.72%
----------Test Result----------
=====Confusion Matrix=======
 [[27208   547]
 [ 5746 28946]]
Test Accuracy Score : 89.92%

 Test Classification Report
              precision    recall  f1-score   support

           0       0.83      0.98      0.90     27755
           1       0.98      0.83      0.90     34692

    accuracy                           0.90     62447
   macro avg       0.90      0.91      0.90     62447
weighted avg       0.91      0.90      0.90     62447

Cross validation score of K Neighbors Classifier model : 89.89594383929192
```

3. Decision Tree Classifier

Model 3 : Decision Tree Classifier

```
  1 dt = DecisionTreeClassifier()
  2
  3 dt.fit(X_train,y_train)      # Model training
  4
  5 pred_dt = dt.predict(X_test)    # Predicted values
  6
  7 metric_score(dt,X_train,X_test,y_train,y_test,train=True)  # Training result
  8
  9 metric_score(dt,X_train,X_test,y_train,y_test,train=False)  # Test result
 10
 11 cr_dt = cross_val_score(dt,princ_comp,y,cv=5)
 12
 13 print("Cross validation score of Decision Tree Classifier model :",cr_dt.mean()*100)
```

```
----------Train Result----------
Train Accuracy Score : 100.00%
----------Test Result----------
=====Confusion Matrix=======
 [[24174  3581]
 [ 4780 29912]]
Test Accuracy Score : 86.61%

 Test Classification Report
              precision    recall  f1-score   support

           0       0.83      0.87      0.85     27755
           1       0.89      0.86      0.88     34692

    accuracy                           0.87     62447
   macro avg       0.86      0.87      0.86     62447
weighted avg       0.87      0.87      0.87     62447

Cross validation score of Decision Tree Classifier model : 86.88567139857574
```

4. Random Forest Classifier

Model 4 : Random Forest Classifier

```
rf = RandomForestClassifier()

rf.fit(X_train,y_train)      # Model training

pred_rf = rf.predict(X_test)    # Predicted values

metric_score(rf,X_train,X_test,y_train,y_test,train=True)  # Training result

metric_score(rf,X_train,X_test,y_train,y_test,train=False)  # Test result

cr_rf = cross_val_score(rf,princ_comp,y,cv=5)

print("Cross validation score of Random Forest Classifier model :",cr_rf.mean()*100)
```

```
----------Train Result----------
Train Accuracy Score : 100.00%
----------Test Result----------
=====Confusion Matrix=======
 [[25782  1973]
 [ 2468 32224]]
Test Accuracy Score : 92.89%

 Test Classification Report
              precision    recall  f1-score   support

           0       0.91      0.93      0.92     27755
           1       0.94      0.93      0.94     34692

    accuracy                           0.93     62447
   macro avg       0.93      0.93      0.93     62447
weighted avg       0.93      0.93      0.93     62447

Cross validation score of Random Forest Classifier model : 93.06635280344867
```

5. Support Vector Classifier

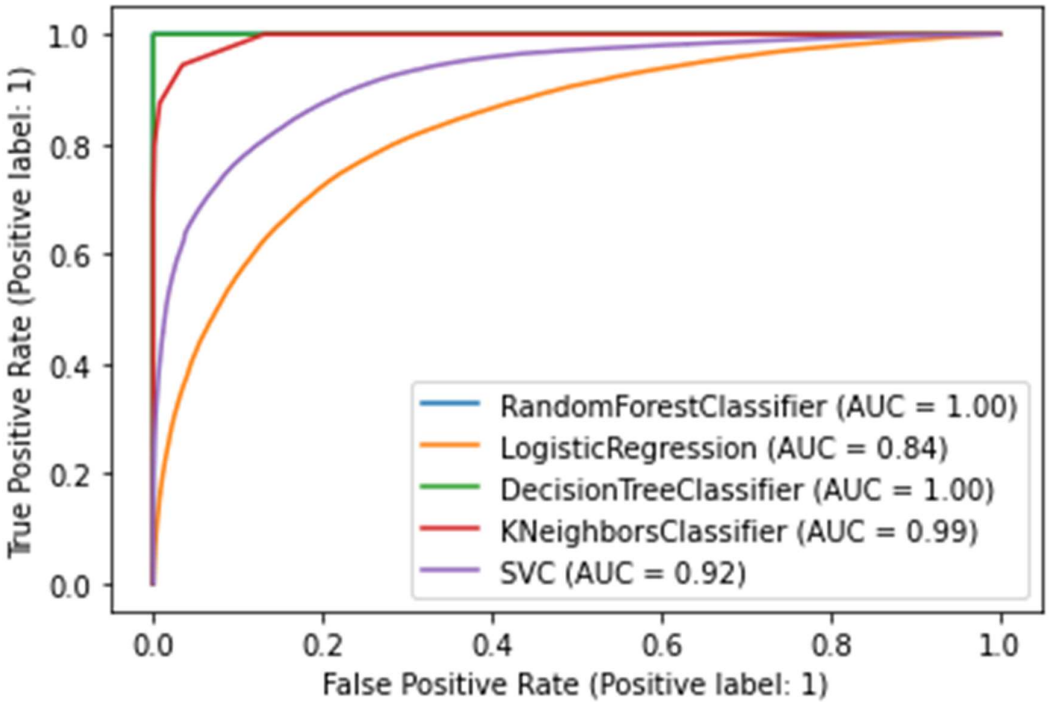
Model 5 : Support Vector Classifier

```
svc = SVC()

svc.fit(X_train,y_train)

pred_svc = svc.predict(X_test)    # Predicted values

metric_score(svc,X_train,X_test,y_train,y_test,train=True)

metric_score(svc,X_train,X_test,y_train,y_test,train=False)

cr_svc = cross_val_score(svc,princ_comp,y,cv=5)

print("Cross validation score of Support Vector Classifier model :",cr_svc.mean()*100)
```

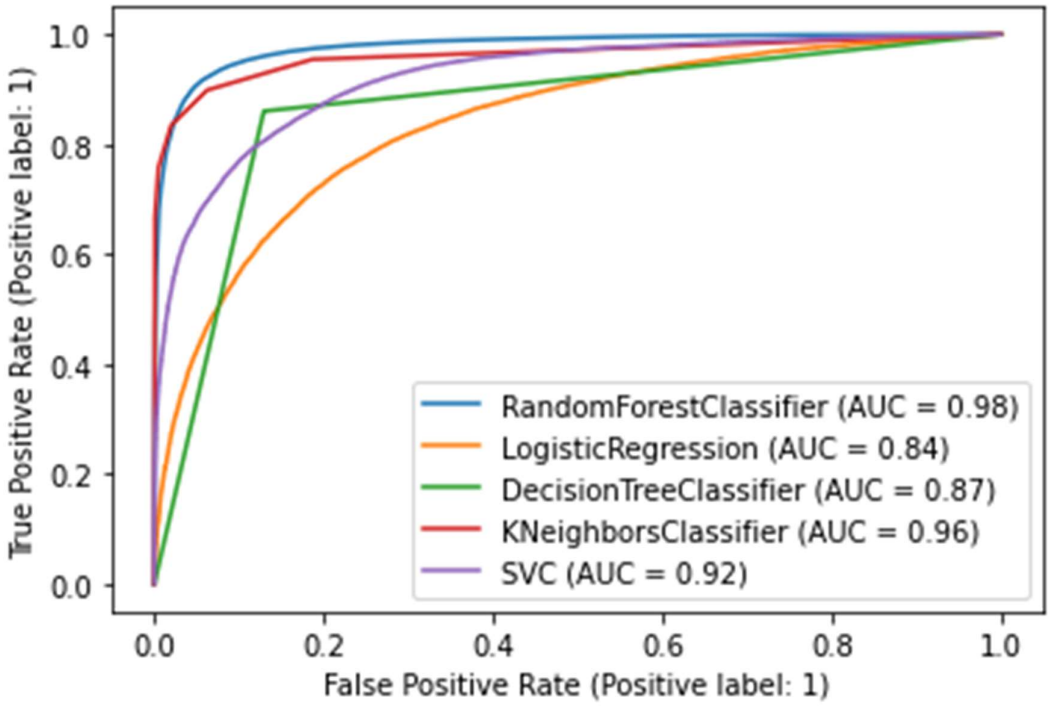
```
----------Train Result----------
Train Accuracy Score : 84.00%
----------Test Result----------
=====Confusion Matrix=======
 [[22973  4699]
 [ 5307 29468]]
Test Accuracy Score : 83.98%

 Test Classification Report
              precision    recall  f1-score   support

           0       0.81      0.83      0.82     27672
           1       0.86      0.85      0.85     34775

    accuracy                           0.84     62447
   macro avg       0.84      0.84      0.84     62447
weighted avg       0.84      0.84      0.84     62447
```

AOC -ROC CURVE OF DIFFERENT ML MODELS

✓ **For Training Dataset**



✓ **For Test Dataset**



We can see Random Forest Classifier gives maximum AUC. It also gives us the highest accuracy score and cross-validation score. Hyperparameter tuning performs on this model to enhance the accuracy of the model.

```
params_rf = {'criterion':['gini','entropy'],
             'max_depth': [10,25],
             'min_samples_split' :[2,3],
             'min_samples_leaf' :[2,3]}

grd_rf = GridSearchCV(rf,param_grid = params_rf,n_jobs =-1)

grd_rf.fit(X_train,y_train)

print("Best parameters : ",grd_rf.best_params_)
```

```
Best parameters :  {'criterion': 'entropy', 'max_depth': 25, 'min_samples_leaf': 2, 'min_samples_split': 3}
```

```
rf = grd_rf.best_estimator_

rf.fit(X_train,y_train)

pred_rf = rf.predict(X_test)    # Predicted values

metric_score(rf,X_train,X_test,y_train,y_test,train=True)  # Training result

metric_score(rf,X_train,X_test,y_train,y_test,train=False)  # Test result

cr_rf = cross_val_score(rf,princ_comp,y,cv=5)

print("Cross validation score of Random Forest Classifier model :",cr_rf.mean()*100)
```

```
----------------Train Result--------------------
Train Accuracy Score : 98.87%
----------------Test Result--------------------
=====Confusion Matrix=======
 [[25559  2196]
 [ 2749 31943]]
Test Accuracy Score : 92.08%

 Test Classification Report
               precision    recall  f1-score   support

           0       0.90      0.92      0.91     27755
           1       0.94      0.92      0.93     34692

    accuracy                           0.92     62447
   macro avg       0.92      0.92      0.92     62447
weighted avg       0.92      0.92      0.92     62447

Cross validation score of Random Forest Classifier model : 92.2073743919977
```

Hyperparameter tuning leads to a slight decrease in the difference between training and test accuracy scores from 8% to 6%. The test accuracy score decreases from 92.89% to 92.08%.

The final model is saved using a pickle

Saving the Model

```
[ ] 1 filename = 'microcredit_defaulter_clf.pkl'
    2 pickle.dump(rf,open(filename,'wb'))
```

```
[ ] 1 loaded_model = pickle.load(open('microcredit_defaulter_clf.pkl','rb'))
    2
    3 pred_rf = loaded_model.predict(X_test)
    4
    5 result = accuracy_score(y_test,pred_rf)
    6
    7 print(result*100)
```

```
92.92520057008343
```

```
1 conclusion = pd.DataFrame([pred_rf,y_test],index=['Predicted','Original'])
2
3 conclusion
```

	0	1	2	3	4	5	6	7	8	9	...	62437	62438	62439	62440	62441	62442	62443	62444	62445	62446
Predicted	0	1	0	1	1	0	0	0	1	0	...	1	1	1	1	1	0	0	1	1	0
Original	0	0	0	1	1	0	1	0	1	0	...	1	1	1	0	1	0	0	1	1	0

2 rows × 62447 columns

- Key Metrics for success in solving a problem under consideration

Following metrics used for evaluation:

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

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

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

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

5. 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

6. AUC_ROC _score: ROC curve. It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for several different candidate threshold values between 0.0 and 1.0

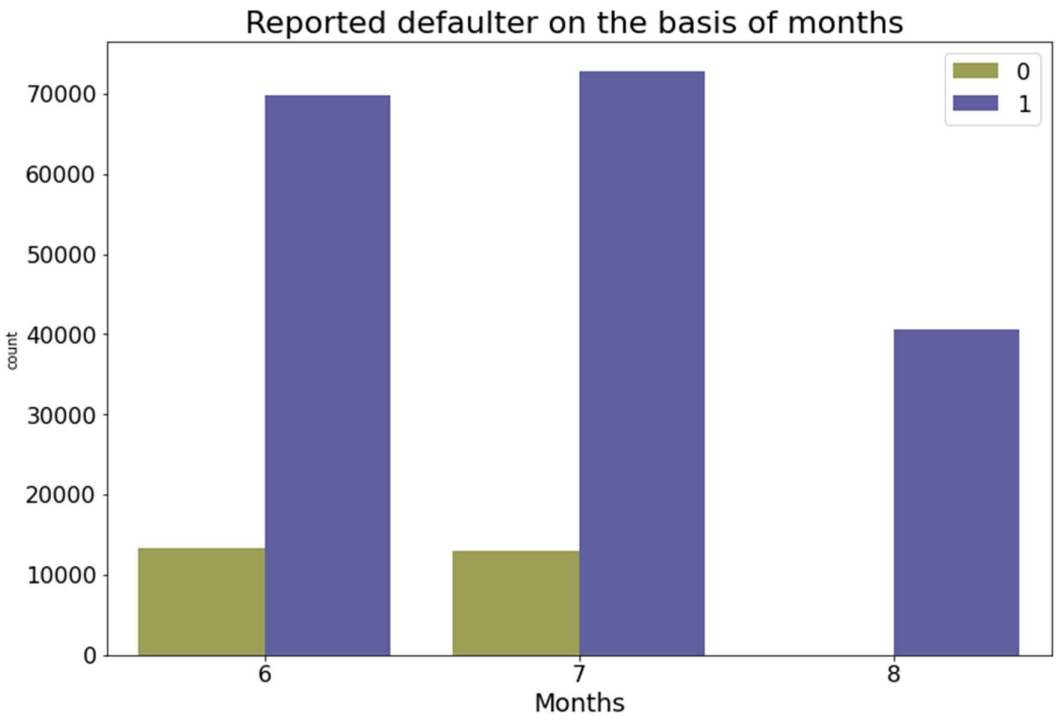
We have used the ROC curve, Accuracy score and Cross-validation score as key parameters for model evaluation in this project since the balancing of data is performed.

- Visualizations

  ➢ <u>Distribution Plot</u>

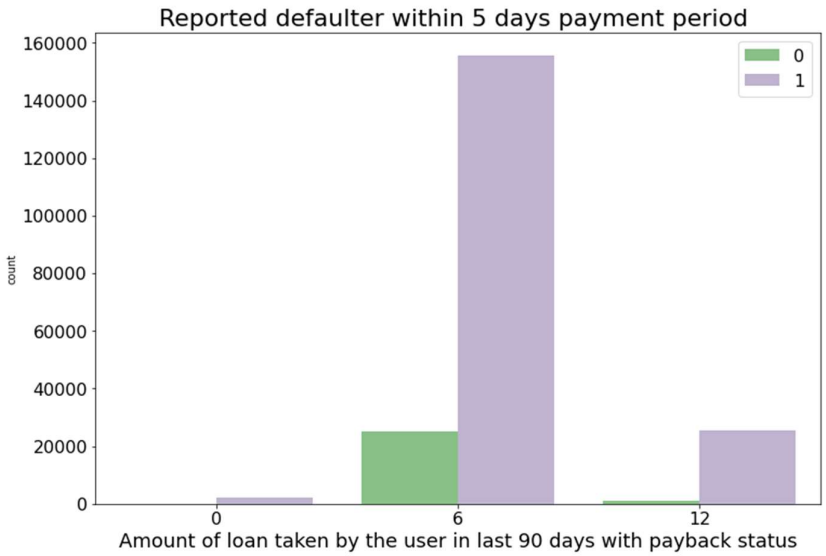
The data distribution of most of the features is positively skewed.

➢ Defaulters based on month

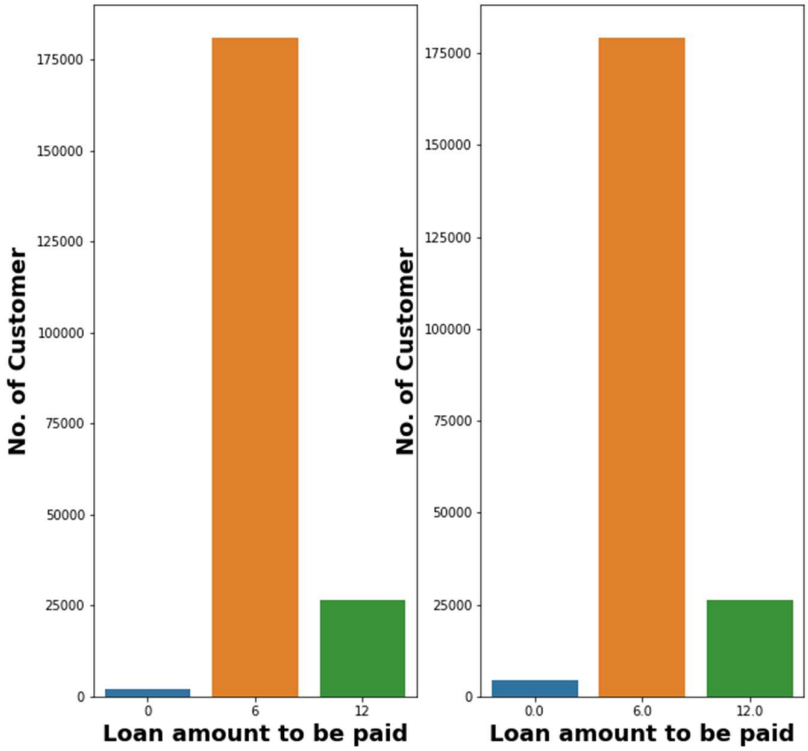


✓ **June and July observed almost equal number of defaulters but the first 20 days of August didn't observe any defaulters**

➢ Status of Defaulters within 5 days payment period



- ✓ **Very few defaulters in customers who take a loan in the amount of 12, most of the defaulters have taken a loan amount of 6.**



- ✓ In the last 30 days, 179192 people had taken 6Rs as the loan amount whereas the number of people who had not taken a loan is 4291

- ✓ In the last 90 days, 180944 people had taken 6Rs as the loan amount and the number of people is whereas the number of people who had not taken a loan is 2043

- ✓ 26605 people had taken 12Rs as the loan amount within 90 days and 26109 people had taken 12Rs as the loan amount for 30 days

- ✓ This means customers mostly opt for a 6 Rs Loan

4. CONCLUSION

- **Key Findings and Conclusions of the Study**

ML Algorithm	Accuracy Score		CV Score
	Train	Test	
Logistic Regression	76.25%	76.69%	76.34%
K Neighbors Classifier	92.72%	89.92%	89.89%
Decision Tree Classifier	100%	86.61%	86.88%
Random Forest Classifier	100%	92.89%	93.06%
Support Vector Classifier	84%	83.98%	-
Hyper Parameter Tuning for Random Forest Classifier	98.87%	92.08%	92.21%

- **Learning Outcomes of the Study in respect of Data Science**

  - ✓ I handled such a huge dataset for the first time. The data cleaning process requires huge attention because for some features, there may be values that might not be realistic and we have to observe them and treat them with a suitable explanation
  - ✓ The data was huge and required high computational capacity, which made me switch to Google Colab for running models. The Hyper Tuning for the final model was done with Google Colab GPU and still, it takes 2-3 hours for finding the best parameter

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

✓ Limited computational resources put a limitation on optimization through hyperparameter tuning. The accuracy of the model can increase with hyperparameter tuning with several different parameters. Here we use only two to three parameters for tuning.

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