FLIP ROBO

Micro Credit Defaulter Project

FLIP ROBO

Micro Credit Defaulter Project

Submitted by:

Bhavesh Kharve

ACKNOWLEDGMENT

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

INTRODUCTION

• Business Problem Framing

It is a project related to Telecom Industry. They have collaborated with MicrofinanceInstitution (MFI) that offers financial services to low income populations. Micro Finance Service (MFS) become very useful when we are targeting unbanked poor families living in remote areas with not much sources of income.

The client is in telecom industry and they are a fixed wireless telecommunications network provider and they understand the importance of communication and how itaffects a person's life, thus focusing on providing their services and products to low income families and customers that can help them in the need of hour. They are collaborating with an MFI to provide micro – credit on mobile balances to be paid back in 5 days.

The Consumer is believed to be defaulter if he deviates from the path of paying back the loaned amount within the time duration of 5 days. We have to build a predictionmodel which will tell us whether the person will become defaulter or not thus helping company in giving credit. This would help client company in further investment and improvement in selection of customers.

• Review of Literature

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

• Motivation for the Problem Undertaken

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

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

Let’s import our csv file into by importing some important library and loading into our jupyter Notbook

Importing Important Library

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

```
#Loading the Dataset
pd.set_option('display.max_columns',None)
df=pd.read_csv('Data file.csv',parse_dates=['pdate'])
df.head()
```

	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_rech:
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	

```
]: ### Checking shape
df.shape
```

```
]: (209593, 37)
```

The dataset contain 209593 rows and 37 columns including Target columns.

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

- There are 209593 distinct micro-credit customers.
- There are 37 attribute including target attribute.
- ‘label’ is target which indicate whether user paid the credit loan amount within 5 days of issuing the loan(1:success,0:failure)

- # Statistical Summary

  - Let's Check the Statistical Summary of our dataset.

Statistical Summary

In [18]:
```
## Statistical Summary
df.describe()
```

Out[18]:

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	c
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	2
mean	0.875177	8112.343445	5381.402289	6082.515068	2692.581910	3483.406534	3755.847800	3712.202921	2064.452797	
std	0.330519	75696.082531	9220.623400	10918.812767	4308.586781	5770.461279	53905.892230	53374.833430	2370.786034	
min	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000	-29.000000	0.000000	
25%	1.000000	246.000000	42.440000	42.692000	280.420000	300.260000	1.000000	0.000000	770.000000	
50%	1.000000	527.000000	1469.175667	1500.000000	1083.570000	1334.000000	3.000000	0.000000	1539.000000	
75%	1.000000	982.000000	7244.000000	7802.790000	3356.940000	4201.790000	7.000000	0.000000	2309.000000	
max	1.000000	999860.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733	999171.809410	55000.000000	

```
## Statistical Summary
df.describe()
```

ans30	medianamnt_loans30	cnt_loans90	amnt_loans90	maxamnt_loans90	medianamnt_loans90	payback30	payback90	Month	Day
00000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.00000
58747	0.054029	18.520919	23.645398	6.703134	0.046077	3.398826	4.321485	6.797321	14.39894
64648	0.218039	224.797423	26.469861	2.103864	0.200692	8.813729	10.308108	0.741435	8.43890
00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	6.000000	1.00000
00000	0.000000	1.000000	6.000000	6.000000	0.000000	0.000000	0.000000	6.000000	7.00000
00000	0.000000	2.000000	12.000000	6.000000	0.000000	0.000000	1.666667	7.000000	14.00000
00000	0.000000	5.000000	30.000000	6.000000	0.000000	3.750000	4.500000	7.000000	21.00000
60864	3.000000	4997.517944	438.000000	12.000000	3.000000	171.500000	171.500000	8.000000	31.00000

- In most of attributes the minimum values is zero and in some attributes like **rental30 and rental90** Which seems to a erroneous data .

  - Mean values are highly deviated from the median value which shows that data distributed is rightly skewed.

  - The difference between 3rd quantile and maximum value are too high hence we can clearly say that our dataset have huge outliers present.

  - The average value for Number of days till last recharge of data account is3712.20. The standard deviation is unusually large, max value being 999171.80.

  - The average value for Number of times main account got recharge in last30 days is 3.97 and the max value of recharge is 203.

  - The average value for number of times data account got recharge in last 30days is 262.57. The standard deviation is high , amo value being 99914.44

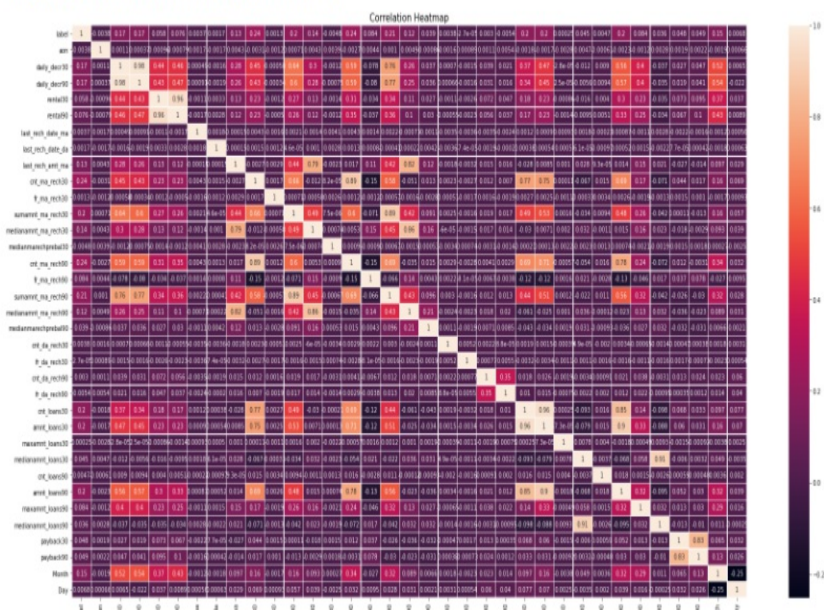
- The average value for number of loans taken by user in last 30 days is 2.75 and std is 2.55, max value

Lets See co-relation between the Columns

Correlation

```
]: plt.figure(figsize=(30,15))
   plt.title('Correlation Heatmap',fontsize=15)
   sns.heatmap(df.corr(),annot=True,linewidths=.2)
```

```
]: <AxesSubplot:title={'center':'Correlation Heatmap'}>
```



- From the above observation we can see that aon, medianmarechprebal30

, fr_da_rech30, fr_da_rech90 are negatively co-related and rest arepositively co-related with label**.**

- Feature like cnt_ma_rech30, cnt_ma_rech90, amnt_loans90, amnt_loan 30 have positive correlation values with target.
- Features like last_recharge_date_ma, fr_ma_rech30 almost have no correlation with Target variable.
- We will not drop any feature based on correlation because data is expensive.

- # Data Sources and their formats:

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

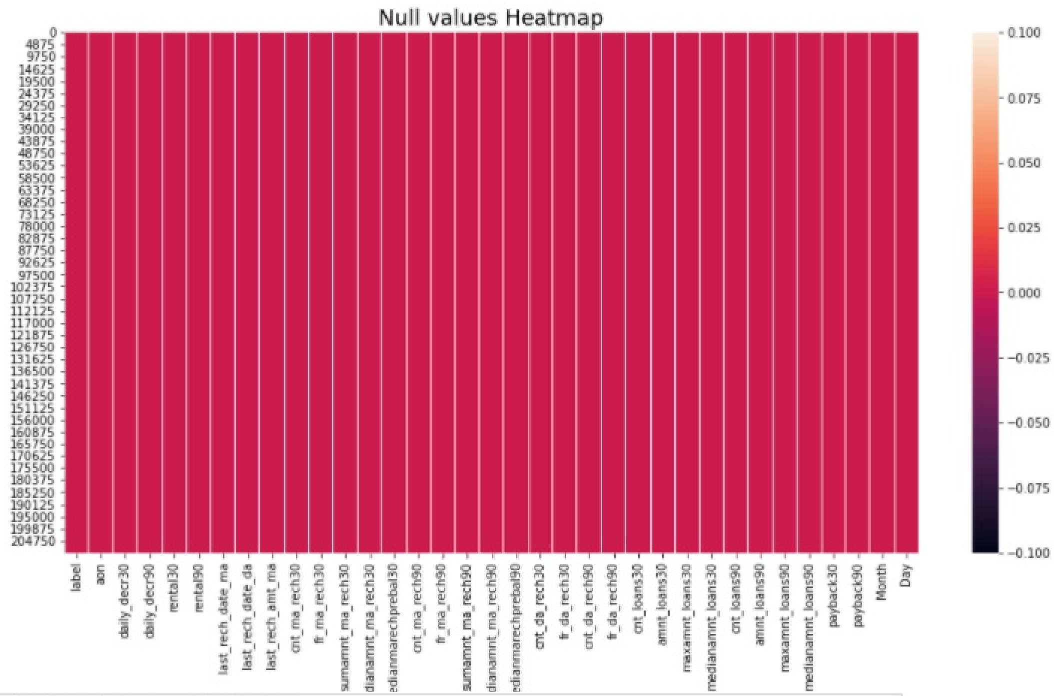
```
[5]: ## Checking the datatype of each attribute
     df.dtypes

[5]: Unnamed: 0                        int64
     label                             int64
     msisdn                           object
     aon                             float64
     daily_decr30                    float64
     daily_decr90                    float64
     rental30                        float64
     rental90                        float64
     last_rech_date_ma               float64
     last_rech_date_da               float64
     last_rech_amt_ma                  int64
     cnt_ma_rech30                     int64
     fr_ma_rech30                    float64
     sumamnt_ma_rech30               float64
     medianamnt_ma_rech30            float64
     medianmarechprebal30            float64
     cnt_ma_rech90                     int64
     fr_ma_rech90                      int64
     sumamnt_ma_rech90                 int64
     medianamnt_ma_rech90            float64
     medianmarechprebal90            float64
     cnt_da_rech30                   float64
     fr_da_rech30                    float64
     cnt_da_rech90                     int64
     fr_da_rech90                      int64
     cnt_loans30                       int64
     amnt_loans30                      int64
     maxamnt_loans30                 float64
     medianamnt_loans30              float64
     cnt_loans90                     float64
     amnt_loans90                      int64
     maxamnt_loans90                   int64
     medianamnt_loans90              float64
     payback30                       float64
     payback90                       float64
     pcircle                          object
     pdate                    datetime64[ns]
     dtype: object
```

- In this dataset there are 37 Attributes
- The Whole dataset is numeric only two feature have object data type .
- Pdate feature has datetime datatype.
- Pcircle and msisdn have object datatype.

- # Data Preprocessing Done
  - Lets check the shape and see count of the number of empty values in eachcolumn.

```
## Visulaization of null values
plt.figure(figsize=(16,8))
plt.title('Null values Heatmap',fontsize=18)
sns.heatmap(df.isnull())
```

```
<AxesSubplot:title={'center':'Null values Heatmap'}>
```




```
6]: ## Creating seperate columns for Year,Month and Date
df['Year']=pd.DatetimeIndex(df['pdate']).year
df['Month']=pd.DatetimeIndex(df['pdate']).month
df['Day']=pd.DatetimeIndex(df['pdate']).day
```

```
7]: ## Dropping the unnecessary Columns
#1-Unnamed: 0---> There is no significance of Serial number to deciding a Defaulter or not.

#2-msisdn---> Mobile number of user has no use to predict whather they pay Loan or not

#3-padte---> we have extracted the details from columns so we can drop that

df=df.drop(['Unnamed: 0','msisdn','pdate'],axis=1)
```

```
8]: df.head(2)
```

8]:

	label	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	sumami
0	0	272.0	3055.05	3065.15	220.13	260.13	2.0	0.0	1539	2	21.0	
1	1	712.0	12122.00	12124.75	3691.26	3691.26	20.0	0.0	5787	1	0.0	

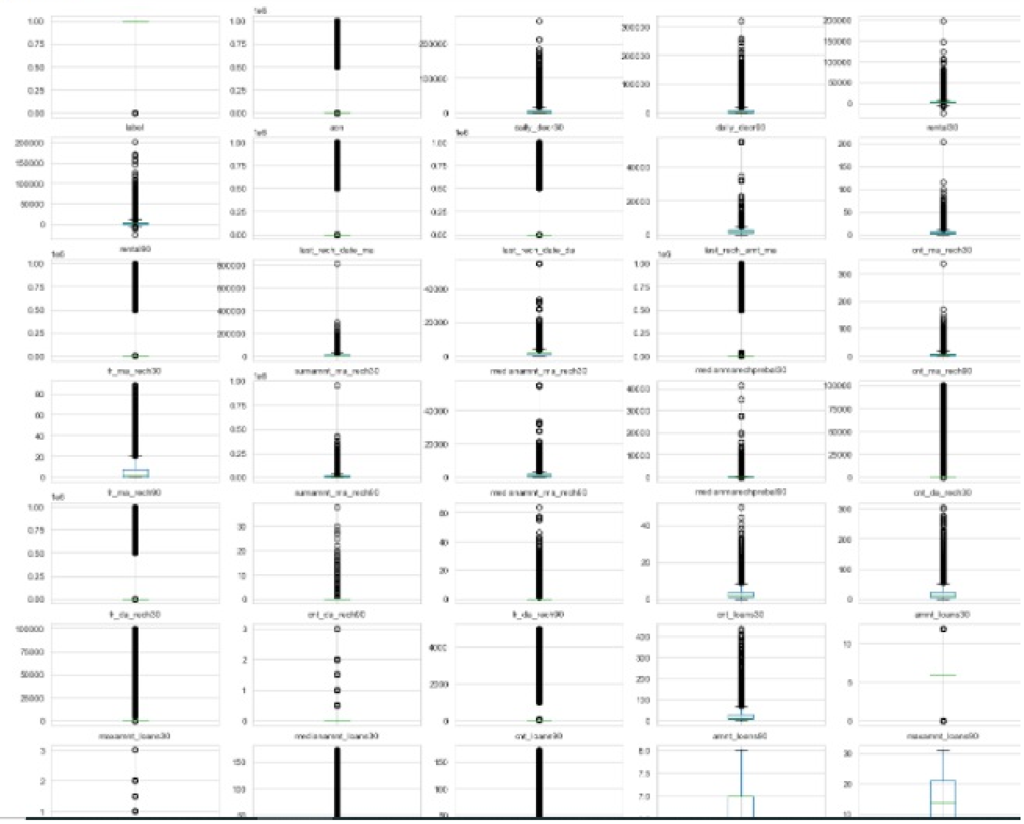
  - As we can see from above Dataset contains 209593 rows and 37 columns inwhich label is the dependent target column and rest are independent columns.
  - And we can see dataset contains no null values.

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

Outliers:

From below boxplot we can see there are many number of outliers present on our dataset.

```
In [34]: df.plot(kind='box',subplots=True,figsize=(20,20),layout=(8,5))
         plt.show()
```



Outliers and Erroneous data removal:

- There is a huge negative and imaginary data present in our dataset. But as per domain knowledge we know that that mobile recharge balance can’t be negative it could be zero only. As same we know that day can’t be negative so we replace all the negative values with zero.

- Here in our data Distribution, we can see that the outliers are present only upper to the upper whisker in box plot which shows that our data is Right skewed and we know that for skewed data we can perform IQR method to detect the outlies.

- We didn’t remove the outliers. Here instead of removing outliers cause data loss upto 10% so here we replace all the outliers with Median values.

Replacing Outliers and Erroneous Data

In [41]:
```
## Hear we found so many -ve values and too much high values .
##We will replace -ve values with zero values because amount or Days can't be in too much negative.

## we will replace all od them by zero
```

In [42]:
```
## Copying the data.
df_n=df.copy()
```

In [43]:
```
df_n.describe()
```

Out[43]:

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

In [44]:
```
## Function for replacing _ve values into zero
def replace_zero(df,col):
    val=np.where(df[col]<0,0,df[col])
    df[col]=val
```

In [45]:
```
def replace_outlier(df,col):
    IQR=df[col].quantile(.75)-df[col].quantile(.25)
    lower_limit=df[col].quantile(.25)-(1.5*IQR)
    upper_limit=df[col].quantile(.75)+(1.5*IQR)
    non_outlier=np.where((df[col]<lower_limit )|(df[col]>upper_limit),df[col].median(),df[col])
    df[col]=non_outlier
```

[47]:
```
##daily_decr30
replace_zero(df_n,'daily_decr30')
replace_outlier(df_n,'daily_decr30')
```

[48]:
```
#daily_decr90
replace_zero(df_n,'daily_decr90')
replace_outlier(df_n,'daily_decr90')
```

[49]:
```
#rental30
replace_zero(df_n,'rental30')
replace_outlier(df_n,'rental30')
```

[50]:
```
#rental90
replace_zero(df_n,'rental90')
replace_outlier(df_n,'rental90')
```

[51]:
```
##last_rech_date_ma
replace_zero(df_n,'last_rech_date_ma')
replace_outlier(df_n,'last_rech_date_ma')
```

[52]:
```
#last_rech_date_da
replace_zero(df_n,'last_rech_date_da')
replace_outlier(df_n,'last_rech_date_da')
```

[53]:
```
#last_rech_amt_ma
replace_zero(df_n,'last_rech_amt_ma')
replace_outlier(df_n,'last_rech_amt_ma')
```

[54]:
```
#cnt_ma_rech30
replace_zero(df_n,'cnt_ma_rech30')
replace_outlier(df_n,'cnt_ma_rech30')
```

[55]:
```
#fr_ma_rech30
replace_zero(df_n,'fr_ma_rech30')
replace_outlier(df_n,'fr_ma_rech30')
```

Skewness Removal:

Our dataset had positive skewness . So after removing outliers we had to remove skewness for getting a data which is close to normal distribute bell curve. So , for getting that we had to apply some transformation methods to remove skewness. Hence, we applied here a root square method to remove skewness of a Right skewed distribution.

```
30]: ## removing skewness
for col in df_2:
    if 0 in df_2[col].unique():
        pass
    if df_2[col].skew()>=.5:
        df_2[col]=np.sqrt(df_2[col])
```

```
31]: ## New, skewness has been removed upto some extent.
df_2.skew()
```

```
31]: label                  -2.270254
aon                     0.224041
daily_decr30            0.694472
daily_decr90            0.750918
rental30                0.529688
rental90                0.548382
last_rech_date_ma       0.215603
last_rech_date_da       0.000000
last_rech_amt_ma       -0.693817
cnt_ma_rech30          -0.267780
fr_ma_rech30            0.404083
sumamnt_ma_rech30      -0.056000
medianamnt_ma_rech30    0.271429
medianmarechprebal30    0.099465
cnt_ma_rech90          -0.097947
fr_ma_rech90            0.532328
sumamnt_ma_rech90       0.100126
medianamnt_ma_rech90    0.340781
medianmarechprebal90   -0.044300
cnt_da_rech30           0.000000
fr_da_rech30            0.000000
cnt_da_rech90           0.000000
fr_da_rech90            0.000000
cnt_loans30             0.506050
amnt_loans30            0.461072
maxamnt_loans30         0.000000
medianamnt_loans30      0.000000
cnt_loans90             0.745316
amnt_loans90            0.713667
maxamnt_loans90        -0.903037
medianamnt_loans90      4.038152
payback30               0.684278
payback90               0.357526
Month                   0.343242
Day                     0.199845
dtype: float64
```

Data Inputs- Logic- Output Relationships:

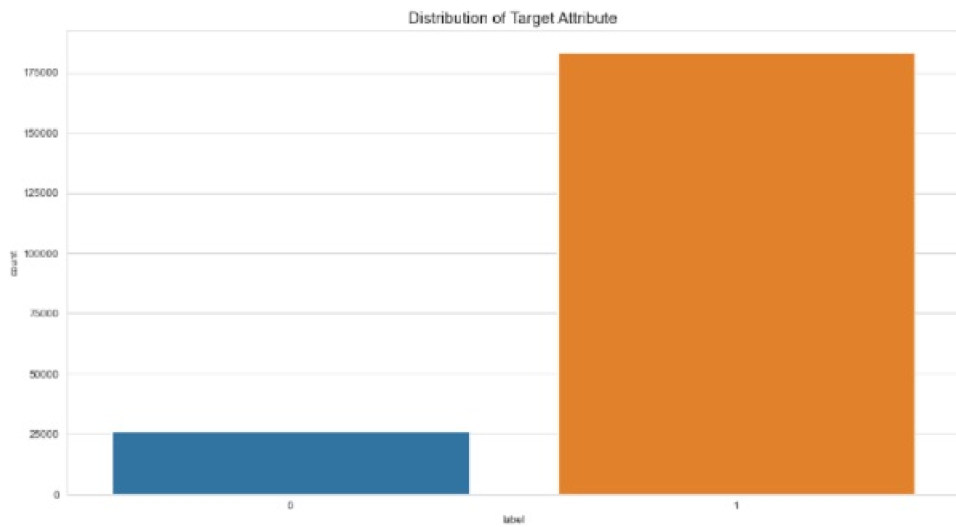
Data Visualization:

Distribution of Target variable

1- It is Clear visible that our target dataset is imbalance .

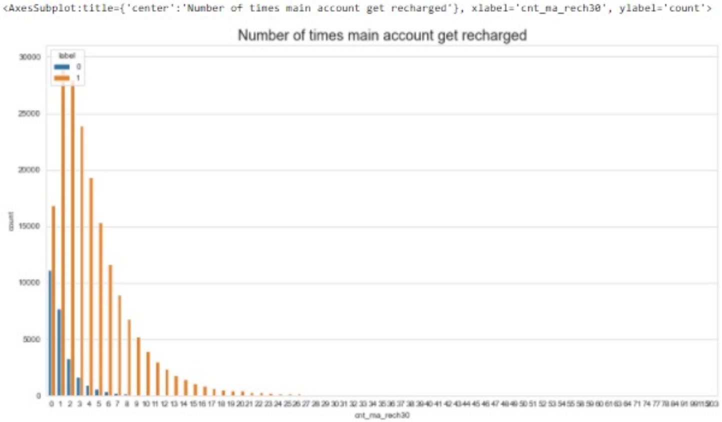
```
## Distribution of Target Variable
plt.figure(figsize=(15,8))
plt.title('Distribution of Target Attribute',fontsize=15)
sns.countplot(df['label'],data=df)
```

```
<AxesSubplot:title={'center':'Distribution of Target Attribute'}, xlabel='label', ylabel='count'>
```



Number of times main account get recharge in 30 days.

```
In [23]: # Number of times main account get recharge in 30 days
         plt.figure(figsize=(15,8))
         sns.set_style(style='whitegrid')
         plt.title('Number of times main account get recharged',fontsize=18)
         sns.countplot(df['cnt_ma_rech30'],hue='label',data=df)
Out[23]: <AxesSubplot:title={'center':'Number of times main account get recharged'}, xlabel='cnt_ma_rech30', ylabel='count'>
```



Observation:

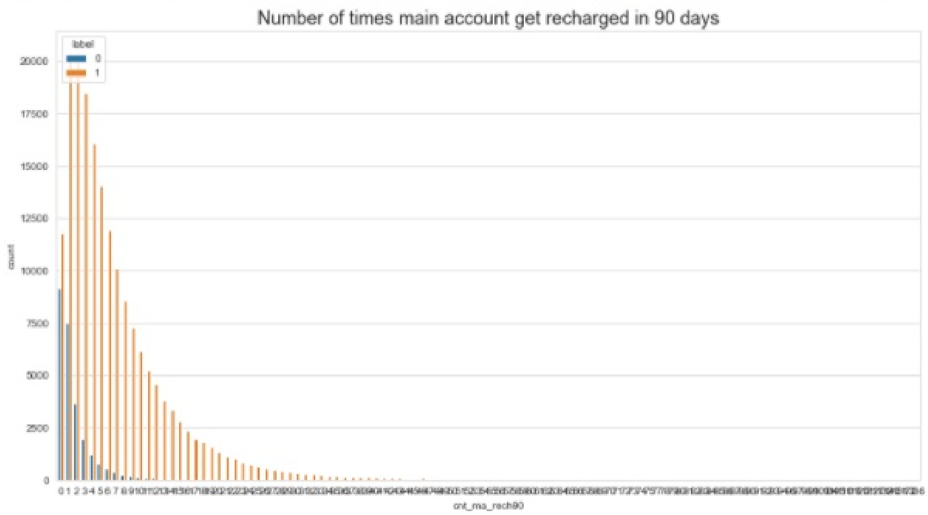
1- Above graph shows that the most people recharge there phone one time in months.

2- People who recharge there phone 3-5 times in months have also very less tendency be a defaulter.

3-People who don't recharge their phone in months have very high tendency to be a defaulter.

Number of times main account get recharge in 90 days

```
In [24]: # Number of times main account get recharge in 90 days
         plt.figure(figsize=(15,8))
         sns.set_style(style='whitegrid')
         plt.title('Number of times main account get recharged in 90 days',fontsize=18)
         sns.countplot(df['cnt_ma_rech90'],hue='label',data=df)

Out[24]: <AxesSubplot:title={'center':'Number of times main account get recharged in 90 days'}, xlabel='cnt_ma_rech90', ylabel='count'>
```



Observation

1- Here, we found the similar trend as of above

2- People who don't recharge their phone in 90 days have higher tendency to take micro loan and to be an defaulter by not paying within 5 days.

3-The trend (Taking loan and being defaulter) goes down as number of time account recharge increase in 90 days.

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

Observation (Below Graph)

1-Mostly user recharge took loan of 6(in Indonesian Rupiah).

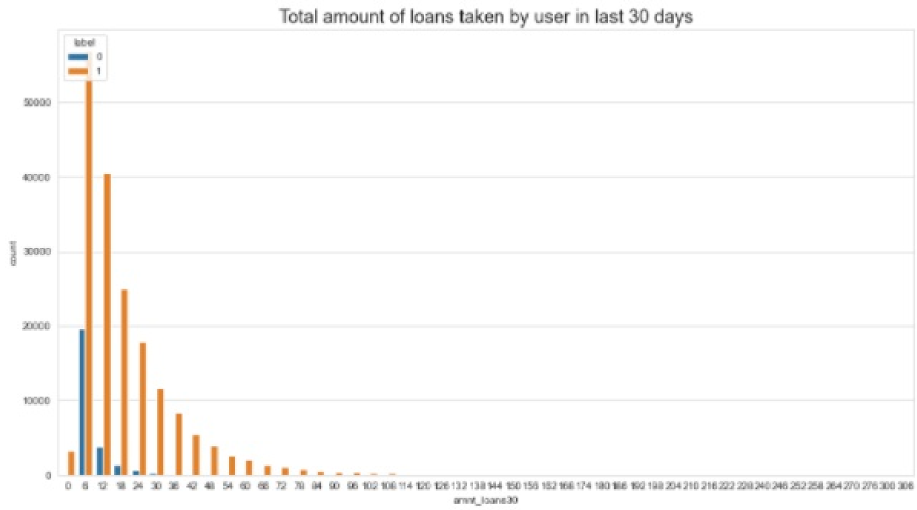
2-As of domain knowledge, if people recharge then 1st, he have to pay the loan then again user get chance to take another loan.

3-12,18,24(in Indonesian Rupiah) could be taken by those people who payback the multiple loan within a month and took another.

4- Gradual drop in loan rupee after 12 Indonesian Rupiah, People are also having tendency to pay back.

```
In [25]: #Total amount of loans taken by user in last 30 days
         plt.figure(figsize=(15,8))
         sns.set_style(style='whitegrid')
         plt.title('Total amount of loans taken by user in last 30 days',fontsize=18)
         sns.countplot(df['amnt_loans30'],hue='label',data=df)
```

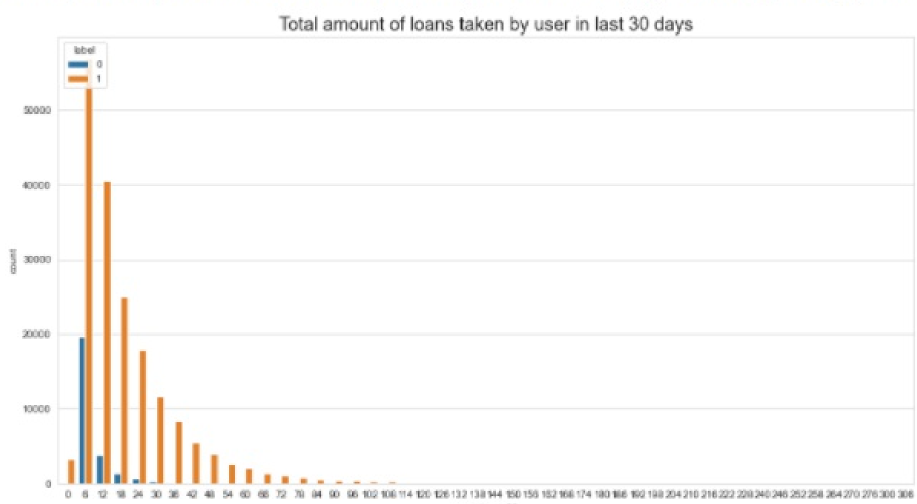
```
Out[25]: <AxesSubplot:title={'center':'Total amount of loans taken by user in last 30 days'}, xlabel='amnt_loans30', ylabel='count'>
```



Number of loans taken by user in last 30 days


```
In [25]: #Total amount of loans taken by user in last 30 days
         plt.figure(figsize=(15,8))
         sns.set_style(style='whitegrid')
         plt.title('Total amount of loans taken by user in last 30 days',fontsize=18)
         sns.countplot(df['amnt_loans30'],hue='label',data=df)
```

```
Out[25]: <AxesSubplot:title={'center':'Total amount of loans taken by user in last 30 days'}, xlabel='amnt_loans30', ylabel='count'>
```

Observation:

1-The trends show, when number of loan taken by user decreases, it's tendency to be a defaulter is also goes down.

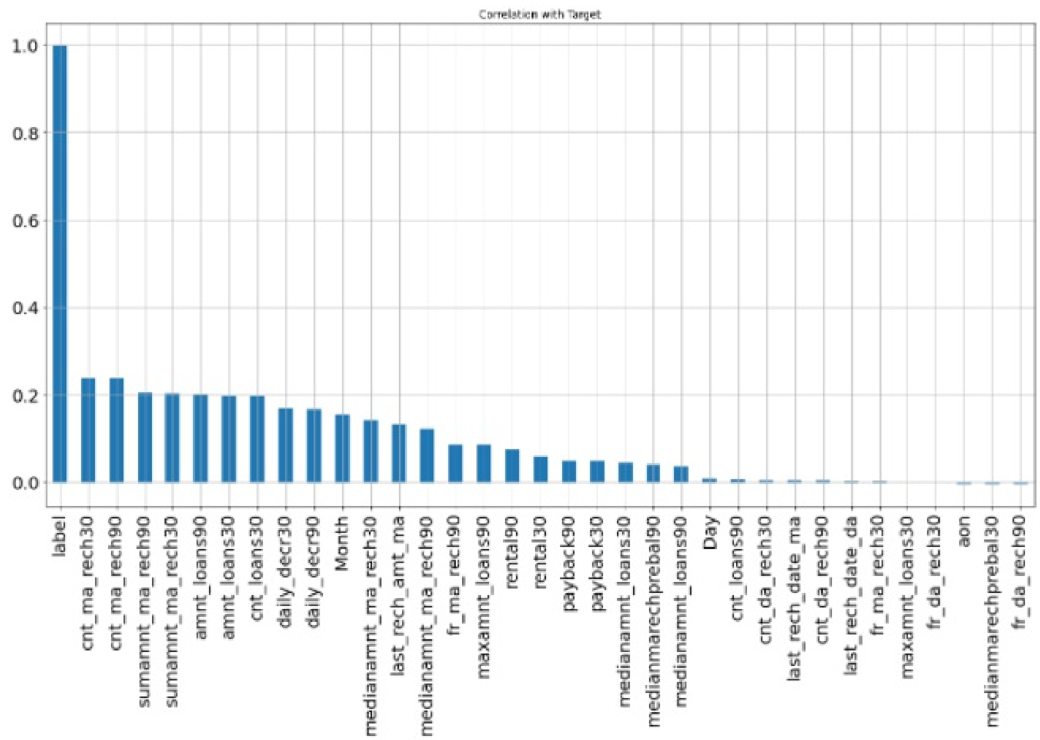
2-There is higher risk to to grant micro loan to a user who take loan once in month.

Correlation Graph:

This is a correlation plot of the of independent features with target features.

```
## Plotting the Correlation with Target.
df.corr()['label'].sort_values(ascending=False).plot.bar(figsize=(20,10),grid=True,title='Correlation with Target',fontsize=20)
```

```
<AxesSubplot:title={'center':'Correlation with Target'}>
```



1-It seems from the above graph is that negatively correlated feature is age on cellular network in days, medianmarechprebal30, but we cannot blindly removethis feature because according to me it is very important feature for prediction.

2- Features like age on age of network (aon),fr_da_rech30,medianmarechprebal30,fr_da_rech90 are negatively correlated but we won't drop these because these are important features.

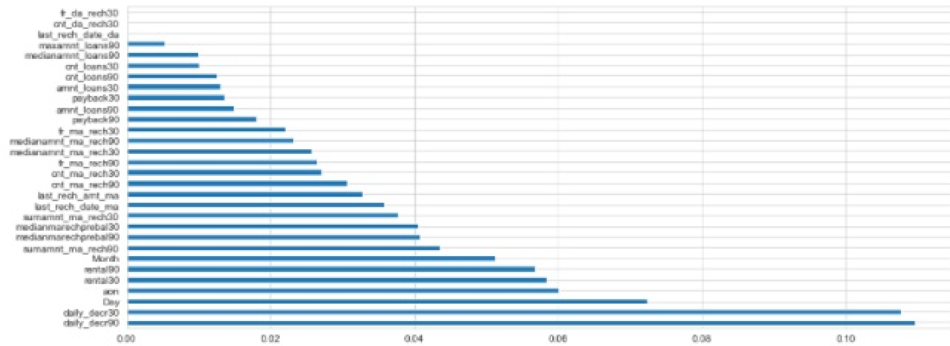
3- We will perform PCA instead of dropping columns based on correlation values.

Feature Importance:

In below diagram, we can clearly see the important features for our Model. We will not remove any data columns based on this graph. We will do dimension Reduction by PCA.

Here we can see features like fr_da_rech30,cnt_da_rech30,last_rech_data_da has no contribution to predict our outcome.

```
In [88]: plt.figure(figsize=(15,6))
         feat_importances=pd.Series(selection.feature_importances_,index=x.columns)
         feat_importances.nlargest(30).plot(kind='barh')
         plt.show()
```



PCA:

PCA is a Dimensionality Reduction Algorithm. Here we imported PCA library from sklearn, then after successfully scaling out dataset we fit our independent data (X) and get that 99% of data from 20 n_components. So , we have chosen 20 features for our model building.

PCA

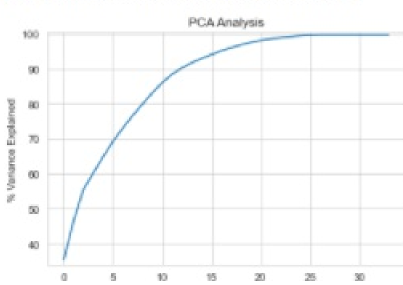
```
92]: from sklearn import decomposition
     from sklearn.decomposition import PCA
     covar_matrix=PCA(n_components=34)
```

```
93]: #Calculate Eigenvalues
     covar_matrix.fit(x)
     variance = covar_matrix.explained_variance_ratio_ #calculate variance ratios

     var=np.cumsum(np.round(covar_matrix.explained_variance_ratio_, decimals=3)*100)
     var #cumulative sum of variance explained with [n] features
```

```
93]: array([35.5, 46.6, 55.6, 60.3, 64.9, 69.3, 73.1, 76.7, 80. , 83.2, 86.1,
            88.5, 90.3, 91.8, 93. , 94.1, 95.2, 96.1, 96.9, 97.6, 98.2, 98.6,
            98.9, 99.2, 99.5, 99.7, 99.8, 99.8, 99.8, 99.8, 99.8, 99.8, 99.8,
            99.8])
```

```
94]: plt.ylabel('% Variance Explained')
     plt.xlabel('# of Features')
     plt.title('PCA Analysis')
     plt.ylim(34,100.5)
     plt.style.context('seaborn-whitegrid')


     plt.plot(var)
```

```
94]: [<matplotlib.lines.Line2D at 0x1e60728f2b0>]
```



Model/s Development and Evaluation

- Step1: Assigning Input and Output variable

Here we split the data frame into independent and dependent variables.
X is the independent variable and y is dependent variable.

Splitting Data into Input and Output Variable

83]:
```
x=df_2.drop(['label'],axis=1)
y=df_2[['label']]
```

84]:
```
x
```

84]:

	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
0	16.492423	55.272507	55.363797	14.836779	16.128546	1.414214	0.0	39.230090	1.414214	1.414214
1	26.683328	110.099955	110.112443	60.755740	60.755740	1.732051	0.0	39.230090	1.000000	0.000000
2	23.130067	37.389838	37.389838	30.002167	30.002167	1.732051	0.0	39.230090	1.000000	0.000000
3	15.524175	4.607385	4.607385	12.626163	12.626163	1.732051	0.0	30.773365	0.000000	0.000000
4	30.773365	12.272707	12.272707	33.149661	33.149661	2.000000	0.0	48.052055	2.645751	1.414214
...	...	...	...	...	...	...	...	...	...	...
209588	20.099751	12.323649	12.323649	33.002879	33.002879	1.000000	0.0	63.623895	1.732051	1.414214
209589	32.787193	6.077499	6.077499	41.573549	41.573549	2.000000	0.0	27.802878	2.000000	1.000000
209590	31.827661	108.826062	109.107058	76.582589	94.303765	1.732051	0.0	39.230090	2.236068	2.828427
209591	41.617304	111.750742	112.135498	20.293595	31.378018	1.414214	0.0	27.802878	2.236068	2.000000
209592	39.761791	67.002701	67.341072	21.998182	25.123694	3.605551	0.0	39.230090	1.414214	1.000000

209593 rows × 34 columns

85]:
```
y
```

85]:

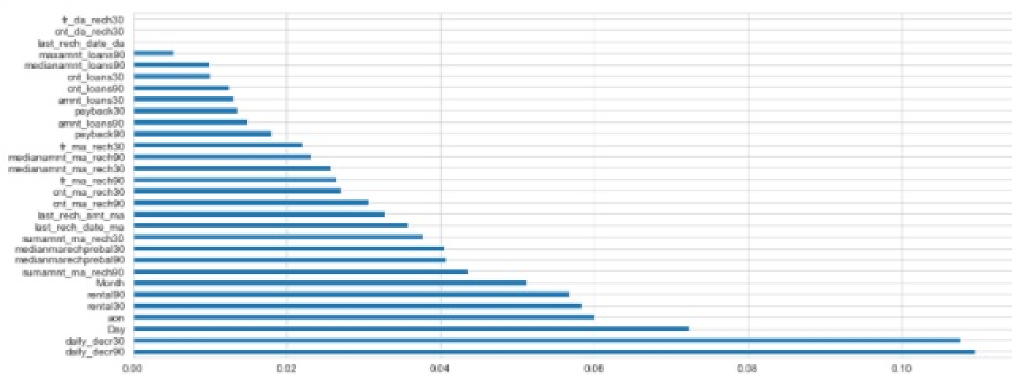
	label
0	0
1	1
2	1
3	1
4	1
...	...
209588	1
209589	1
209590	1

Lets check feature importance of the Data set.

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

Checking Feature Importance

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

]: ExtraTreesClassifier()

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

   [0.06006846 0.10774161 0.10957921 0.05831086 0.05677599 0.03580977
    0.         0.03275875 0.02710076 0.02201545 0.03773421 0.02577785
    0.04053981 0.03066627 0.02647991 0.04356788 0.0231837  0.04068402
    0.         0.         0.         0.         0.01010894 0.01317164
    0.         0.         0.01250058 0.01490529 0.00532127 0.0099474
    0.01368944 0.01802149 0.05122605 0.0723134 ]

]: plt.figure(figsize=(15,6))
   feat_importances=pd.Series(selection.feature_importances_,index=x.columns)
   feat_importances.nlargest(30).plot(kind='barh')
   plt.show()
```



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

Scaling: Standard Scaling

- Scaling is required in distance-based algorithms like Logistic Regression, PCA, KNN and Gradient Boosting.
- In our independent feature data have different units and variation is there.
- So, to scale down all features we use standard scaling.

Standard Scaling:

[89]:
```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
df_scaler=ss.fit_transform(x)
x=pd.DataFrame(df_scaler)
x.head()
```

t[89]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	-0.732246	0.358349	0.337491	-0.850338	-0.844808	-0.283300	0.0	0.317109	-0.200301	0.130937	-0.260414	0.429289	-0.765622	-0.521525	0.046739
1	0.336867	1.804645	1.743975	1.245366	0.963983	0.054503	0.0	0.317109	-0.683018	-1.076361	0.281189	0.429289	0.743068	-0.927255	-1.150579
2	-0.035900	-0.113378	-0.124257	-0.158202	-0.282494	0.054503	0.0	0.317109	-0.683018	-1.076361	-0.687794	0.429289	0.841453	-0.927255	-1.150579
3	-0.833823	-0.978148	-0.966433	-0.951229	-0.986763	0.054503	0.0	-0.250254	-1.848398	-1.076361	-1.719578	-1.669927	-1.579883	-0.927255	-1.150579
4	0.765947	-0.775944	-0.769512	-0.014553	-0.154923	0.339284	0.0	0.908976	1.234909	0.130937	2.002620	1.479579	0.021266	0.863723	0.046739

- # Testing of Identified Approaches (Algorithms)

Model Building

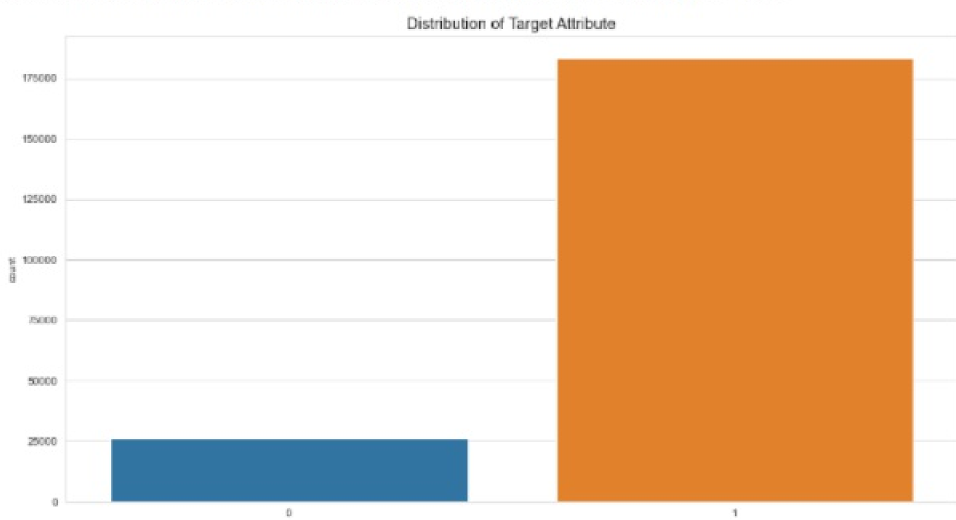
```
#importing important Libraries
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,roc_auc_score,roc_curve,f1_score,auc
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV,RandomizedSearchCV
from imblearn.over_sampling import SMOTE
```

Target Distribution:

- Our target variable is imbalanced in nature. Where 1 represent that people return the loan and 0 shows they are fail to return loan.
- Due to Imbalance dataset we applied here up sampling method (SMOTE) to our training dataset.

```
## Distribution of Target Variable
plt.figure(figsize=(15,8))
plt.title('Distribution of Target Attribute',fontsize=15)
sns.countplot(df['label'],data=df)
```

```
<AxesSubplot:title={'center':'Distribution of Target Attribute'}, xlabel='label', ylabel='count'>
```



Up sampling:

- We done up sampling by using SMOTE .

```
## Train_Test_Split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=45,test_size=.25,stratify=y)
## Upsampling
x_train_s,y_train_s=SMOTE().fit_resample(x_train,y_train)
```

Model Building:

Here we made a function to perform our Training and Testing of Machine Learning Algorithms.

```
[105]: def model_algo (clf ,x_train_s, y_train_s):
           clf.fit(x_train_s,y_train_s)
           pred = clf.predict(x_test)
           acc_scr = accuracy_score(pred, y_test)
           print("\n")
           print("Train Accuracy :",clf.score(x_train_s, y_train_s))
           print("Test Accuracy :",clf.score(x_test, y_test))
           print("\n")
           print('F_1 sccore :',f1_score(pred,y_test))
           print('\n')
           print('ROC_AUC score :',roc_auc_score(pred,y_test))
           print('\n')
           print("Classification Report :\n", classification_report(pred, y_test))
           print("\n")
           print("Confusion Matrix :\n", confusion_matrix(pred, y_test))
           print("\n")
           false_positive_rate, true_positive_rate, threshold = roc_curve(pred, y_test)
           roc_auc = auc(false_positive_rate, true_positive_rate)
           print("ROC_AUC_CURVE :", roc_auc)
```

```
[111]: # Creating the Instances for the Algorithms
       lr=LogisticRegression()
       dt=DecisionTreeClassifier()
       gnb=GaussianNB()
       rf=RandomForestClassifier()
       ada=AdaBoostClassifier()
       gbc=GradientBoostingClassifier()
```

```
[112]: models=[]
       models.append(('LogisticRegression',lr))
       models.append(('DecisionTreeClassifier',dt))
       models.append(('GaussianNB',gnb))
       models.append(('RandomForestClassifier',rf))
       models.append(('AdaBoostClassifier',ada))
       models.append(('GradientBoostingClassifier',gbc))
```

1. **Logistic Regression**

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

```
------------ LogisticRegression ------------


Train Accuracy : 0.7697476975860089
Test Accuracy : 0.7525716139620985


F_1 sccore : 0.8411288247331724


ROC_AUC score : 0.6335709568977632


Classification Report :
               precision    recall  f1-score   support

           0       0.78      0.31      0.44     16650
           1       0.75      0.96      0.84     35749

    accuracy                           0.75     52399
   macro avg       0.77      0.63      0.64     52399
weighted avg       0.76      0.75      0.71     52399



Confusion Matrix :
 [[ 5113 11537]
 [ 1428 34321]]


ROC_AUC_CURVE : 0.6335709568977632


Cross validation score : 0.765145857056040
Standard Deviationin : 0.0023043665951363052
```

2. **Decision Tree Classification**

The idea of a decision tree is to divide the data set into smaller data sets based onthe descriptive features until you reach a small enough set that contains data points that fall under one label.

Decision trees are easy to interpret. To build a decision tree requires little datapreparation from the user- there is no need to normalize data.

```
------------- DecisionTreeClassifier -------------


Train Accuracy : 0.9999745589614241
Test Accuracy : 0.8219049981869883


F_1 sccore : 0.8933461336259114


ROC_AUC score : 0.6544436681878112


Classification Report :
              precision    recall  f1-score   support

           0       0.61      0.37      0.46     10759
           1       0.85      0.94      0.89     41640

    accuracy                           0.82     52399
   macro avg       0.73      0.65      0.68     52399
weighted avg       0.80      0.82      0.80     52399


Confusion Matrix :
 [[ 3984  6775]
 [ 2557 39083]]


ROC_AUC_CURVE : 0.6544436681878112


Cross validation score : 0.86607168777707
Standard Deviationin : 0.002763194361220254
```

3. **Random Forest Classification**

Random Forest is a supervised learning algorithm, it creates a forest and makes it somehow random. The "forest “it builds, is an ensemble of Decision Trees.Step-1Pick at random K data points from the training set.

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

Step-3Choose the Number of trees(n) you want to build and repeat Step1 andStep2

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

Result :

```
------------- RandomForestClassifier -------------


Train Accuracy : 0.9999672900932596
Test Accuracy : 0.887574190347144


F_1 sccore : 0.9350145061830536


ROC_AUC score : 0.7444296999023392


Classification Report :
              precision    recall  f1-score   support

           0       0.63      0.54      0.58      7606
           1       0.92      0.95      0.94     44793

    accuracy                           0.89     52399
   macro avg       0.78      0.74      0.76     52399
weighted avg       0.88      0.89      0.88     52399



Confusion Matrix :
 [[ 4128  3478]
 [ 2413 42380]]


ROC_AUC_CURVE : 0.7444296999023392


Cross validation score : 0.9322092895779821
Standard Deviationin : 0.0013968585825475314
```

4.Gradient Boosting-

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

```
------------- GradientBoostingClassifier -------------


Train Accuracy : 0.8094720621052096
Test Accuracy : 0.7988129544456956


F_1 sccore : 0.8744432004954622


ROC_AUC score : 0.6617359389816733


Classification Report :
              precision    recall  f1-score   support

           0       0.79      0.36      0.49     14295
           1       0.80      0.96      0.87     38104

    accuracy                           0.80     52399
   macro avg       0.79      0.66      0.68     52399
weighted avg       0.80      0.80      0.77     52399



Confusion Matrix :
 [[ 5147  9148]
 [ 1394 36710]]


ROC_AUC_CURVE : 0.6617359389816733


Cross validation score : 0.8074876318360728
Standard Deviationin : 0.004064917938040483
```

5.Naive Bayes:

In statistics, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong independence assumptions between the features. They are among the simplest Bayesian network models, but coupled with kernel density estimation, they can achieve higher accuracy levels.

```
------------ GaussianNB ------------


Train Accuracy : 0.7352205738044529
Test Accuracy : 0.7076089238344243


F_1 sccore : 0.8085042558776107


ROC_AUC score : 0.603283086169113


Classification Report :
              precision    recall  f1-score   support

           0       0.72      0.26      0.38     18250
           1       0.71      0.95      0.81     34149

    accuracy                           0.71     52399
   macro avg       0.71      0.60      0.60     52399
weighted avg       0.71      0.71      0.66     52399



Confusion Matrix :
 [[ 4735 13515]
 [ 1806 32343]]


ROC_AUC_CURVE : 0.603283086169113


Cross validation score : 0.7280757883651319
Standard Deviationin : 0.003761315271607039
```

- ## Key Metrics for success in solving problem underconsideration

Accuracy Score is the number of correct predictions made as a ratio of all predictions made. It is the most common evaluation metric for classificationproblems.
Cross-validation is to call the cross_val_score helper function on the estimator andthe dataset.
To estimate the accuracy of a linear kernel support vector machine on the datasetby splitting the data, fitting a model and computing the score (n=5 or any number provided by you) consecutive times (with different splits each time):

The **Area Under the Curve (AUC)** is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higherthe AUC, the better the performance of the model at distinguishing between the positive and negative classes

Receiver Operating Characteristic(ROC) summarizes the model's performance by evaluating the trade offs between true positive rate (sensitivity) and false positive rate(1- specificity). For plotting ROC, it is advisable to assume p > 0.5 since we aremore concerned about success rate.

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

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

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

RESULT:

Here we have created the data Frame to compare the results of different Machine Learning Algorithms.

Creating DataFrame of Result

In [130]:
```
best_model=pd.DataFrame({'Model':['LogisticRegression','DecisionTreeClassifier','GaussianNB','RandomForestClassifier','AdaBoostC
                         'Accuracy Score':[75.25,82.19,70.19,88.75,76.77,79.88],
                         'F1_Score':[84.11,89.33,80.85,93.50,85.23,87.44],
                         'Cross_validation':[76.51,86.60,72.80,93.20,78.54,80.74]})
best_model
```

Out[130]:

	Model	Accuracy Score	F1_Score	Cross_validation
0	LogisticRegression	75.25	84.11	76.51
1	DecisionTreeClassifier	82.19	89.33	86.60
2	GaussianNB	70.19	80.85	72.80
3	RandomForestClassifier	88.75	93.50	93.20
4	AdaBoostClassifier	76.77	85.23	78.54
5	GradientBoostingClassifier	79.88	87.44	80.74

Observation:

1- Based on all there above results it shows that Random Forest Classifier gives us Test Accuracy : 0.887574190347144

F_1 sccore : 0.9350145061830536 and cross validation score is also Cross validation score : 0.9322092895779821 with least standard deviation(Standard Deviationin : 0.0013968585825475314).

2- After Hyperparameter tuning we will save Random Forest as our Best Model.

- Based of above result we can clearly see that our Random Forest Classifier has the highest accuracy and F_1 score among all the other machine learning models.
- To check the overfitting we also find the cross validation score to compare the model result with 5 cross validation .

- We can clearly see in result that random forest classifier cross validation has the minimum standard deviation and it is also less deviated from the randomly selected random state result with cross validation score of 5 .

Best Model:

- Hence form the above analysis it is clear the our **Random forest model** is not overfit .

- So ,for more exploration we will perform hyperparameter tuning of Random Forest Model.

Hyperparameter Tuning :

- To get better result we will do some hyperparameter tuning of our Random Forest Model.

Hyperparameter Tuning

```
:7]: # Hyper Hyper parameter tuning of RandomForest Classifier

param={'n_estimators':[10,50,100,500],'max_depth':[2,4,6],'min_samples_split':[2,4,6],'criterion':['entropy', 'gini']}

grid=GridSearchCV(rf,param,cv=5,n_jobs=-1,scoring='f1')

grid.fit(x_train_s,y_train_s)

# Print the tuned parameters and score
print("Tuned RandomForest Parameters: {}".format(grid.best_params_))
print("Best score is {}".format(grid.best_score_))
```

```
Tuned RandomForest Parameters: {'criterion': 'gini', 'max_depth': 6, 'min_samples_split': 6, 'n_estimators': 500}
Best score is 0.7901817205128919
```

```
:8]: x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=45,test_size=.25,stratify=y)
## Upsampling
x_train_s,y_train_s=SMOTE().fit_resample(x_train,y_train)
final=RandomForestClassifier(n_estimators=500,max_depth=6,min_samples_split=6,criterion='gini')
final.fit(x_train_s,y_train_s)
pred = final.predict(x_test)
```

```
:9]: print('Final Accuracy_score :',accuracy_score(pred,y_test))
print('Final f_1 score :',f1_score(pred,y_test))
print('Final roc_auc score :',roc_auc_score(pred,y_test))
print('Final classification Report :',classification_report(pred,y_test))
print('Final confusion Matrix :',confusion_matrix(pred,y_test))
```

```
Final Accuracy_score : 0.7849195595335788
Final f_1 score : 0.8651203982957539
Final roc_auc score : 0.6489558643502005
Final classification Report :               precision    recall  f1-score   support

           0       0.76      0.34      0.47     14701
           1       0.79      0.96      0.87     37698

    accuracy                           0.78     52399
   macro avg       0.78      0.65      0.67     52399
weighted avg       0.78      0.78      0.75     52399

Final confusion Matrix : [[ 4986  9715]
 [ 1555 36 Task View
```

Observation:

- We find that with Hyperparameter tuning we gets low accuracy score and F_1 score.

- Sometimes with our default variable we get a good score so we will go with our default parameters.

Final Point:

- Before hypermeter tuning, our accuracy score was 88.75, f_1 score was 93.50 and cross validation score was also 93.20 up to 5 cross validation. Some times with hyperparameter is not ideal for get improved result, As shown above we got our good accuracy and f_1 score with default hyperparameter tuning parameters so we will use Random Forest Classifier as our best model.

Saving And Loading the Model:

- Here we have saved our best model with having a 88.75 accuracy and 93.50 F_1 score.

Save the model

In [131]:
```
import joblib
joblib.dump(rf,'MicroCreditrf.pkl')
```

Out[131]:
```
['MicroCreditrf.pkl']
```

Loading the Model

In [132]:
```
model=joblib.load('MicroCreditrf.pkl')
prediction=model.predict(x_test)
```

In [134]:
```
print(accuracy_score(y_test,prediction))
print(f1_score(y_test,prediction))
print(roc_auc_score(y_test,prediction))
print(confusion_matrix(y_test,prediction))
print(classification_report(y_test,prediction))
```

```
0.887574190347144
0.9350145061830536
0.777626671764461
[[ 4128  2413]
 [ 3478 42380]]
              precision    recall  f1-score   support

           0       0.54      0.63      0.58      6541
           1       0.95      0.92      0.94     45858

    accuracy                           0.89     52399
   macro avg       0.74      0.78      0.76     52399
weighted avg       0.90      0.89      0.89     52399
```

In [ ]:

CONCLUSION

- ## Key Findings and Conclusions of the Study

Today, microfinance is widely accepted as a poverty-reduction tool, representing $70 billion in outstanding loans and a global outreach of200 million clients.

The aim was to determine an appropriate quantities model for usingfinancial information pertaining to the loan and customer behavior on the mobile network to predict the outcome of the loan.

Classification models are appropriate for dealing with the two distinct outcomes for customer behavior of repayment and defaulter.

We have used different models for the prediction.

Thank You