FLIP ROBO

MICRO CREDIT DEFAULTER PROJECT

Submitted by:

Bhushan Kumar Sharma,

Intern Data Scientist

ACKNOWLEDGMENT

*****

I would like to express my special gratitude to “Flip Robo” team, who has given me this opportunity to deal with this dataset and it has helped me to improve my model building skills.

A huge thanks to my academic team “Data trained” who is the reason behind what I am today. Last but not least my parents who have been my backbone in every step of my life. And also thank you for many other persons who has helped me directly or indirectly to complete the project.

Following are the external references which I used:

www.geeksforgeeks.org

www.stackoverflow.com

www.w3school.com

www.google.com

Datatrained Lectures

. INTRODUCTION

➢ Business Problem Framing

They are collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed to be defaulter if he deviates from the path of paying back the loaned amount within the time duration of 5 days. For the loan amount of 5 (in Indonesian Rupiah), payback amount should be 6 (in Indonesian Rupiah), while, for the loan amount of 10 (in Indonesian Rupiah), the payback amount should be 12 (in Indonesian Rupiah).

➢ Conceptual Background of the Domain Problem

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

Many microfinance institutions (MFI), experts and donors are supporting the idea of using mobile financial services (MFS) which they feel are more convenient and efficient, and cost saving, than the traditional high-touch model used since long for the purpose of delivering microfinance services. Though, the MFI industry is primarily focusing on low income families and is very useful in such areas, the implementation of MFS has been uneven with both significant challenges and successes.

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

FlipRobo is 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 its business and organization based on the budget operator model, offering better products at Lower Prices to all value conscious customers through a strategy of disruptive innovation that focuses on the subscriber.

They understand the importance of communication and how it affects a person's life, thus, focusing on providing their services and products

to low income families and poor customers that can help them in the need of hour.

➢ Review of Literature

- From the dataset I get to know that it is a classification problem and there are two categories which are successor and the defaulters. And there are so many features which help to find it.

- Loan giving capacity will get decided based on below parameters-Daily amount spend & average main account balance in last 30 days, Frequency of recharge for data account & main account in 30/90 days , loan taken in last 90 days & payback time for last 30 days.

➢ Motivation for the Problem Undertaken

In order to understand to whom loan to be given from lower income earning people and data from telecom industry clearly stats parameters to be taken into consideration to declare borrower as defaulter or not & amount limit also can be decide based on this.

In every country poor population exists to some scale and financial services to be provided to them at affordable level of loan amount to uplift their financial situation, which may reduce the vulnerability factor.

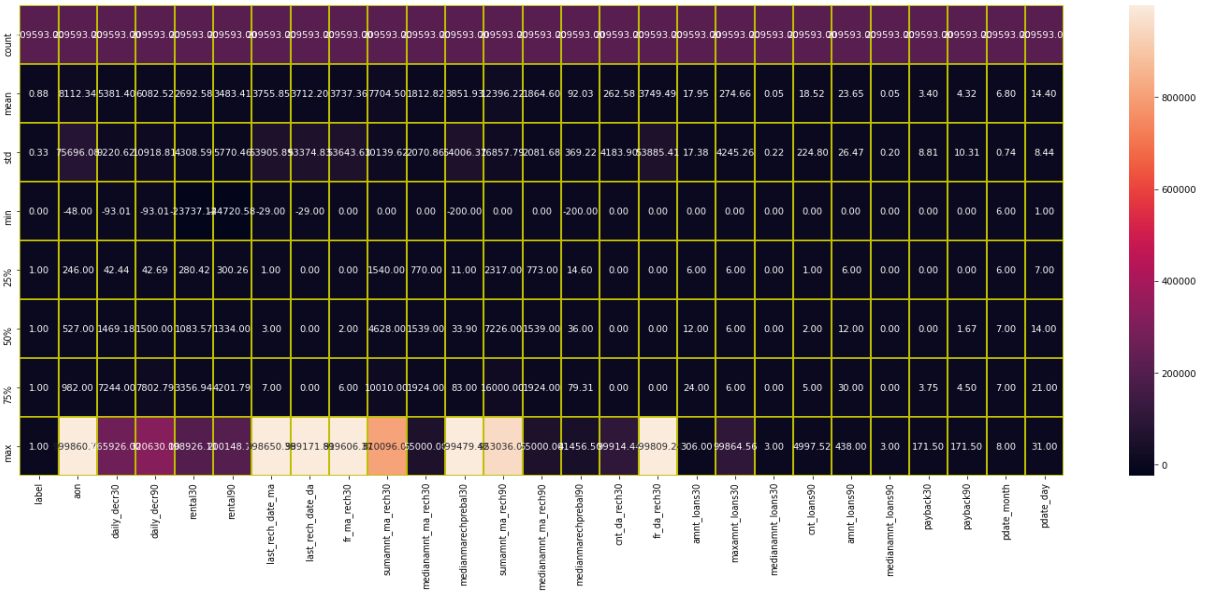
➢ Data Sources and their formats

In this particular problem I had label as my target column and it was having two classes Label '1' indicates that the loan has been payed i.e. Non- defaulter, while, Label '0' indicates that the loan has not been payed i.e. defaulter. So clearly it is a binary classification problem and I have to use all classification algorithms while building the model. There was no null values in the dataset. Also, I observed some unnecessary entries in some of the columns like in some columns I found more than 90% zero values so I decided to drop those columns. If I keep those columns as it is, it will create high skewness in the

model. To get better insight on the features I have used plotting like distribution plot, bar plot and count plot. With these plotting I was able to understand the relation between the features in better manner. Also, I found outliers and skewness in the dataset so I removed outliers using percentile method and I removed skewness using yeo-Johnson method. I have used all the classification algorithms while building model then tuned the best model and saved the best model. At last I have predicted the label using saved model.

Analytical Problem Framing

- **Describe dataset:**



i) As almost every column is showing high difference between mean and 50% percentile, which indicating the skewness of the column

ii) Very high difference found between min and max value of few columns, like: “aon”, “daily_decr30”, “daily_decr90”, “rental30”

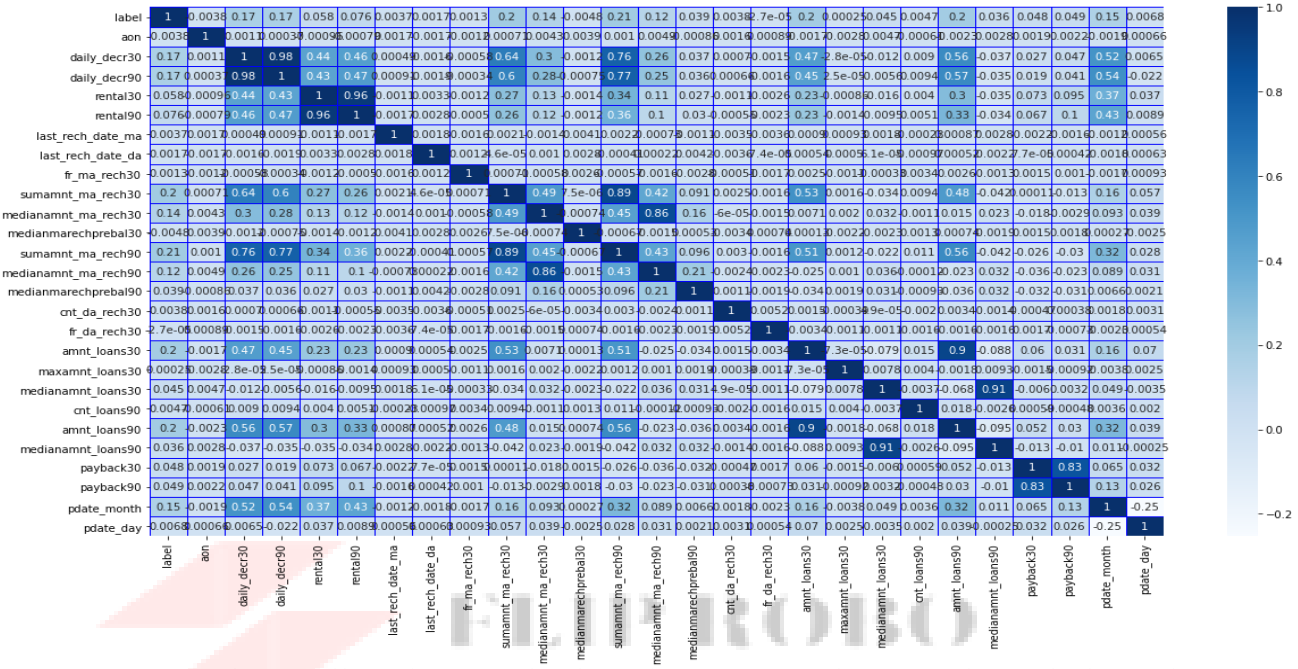
iii) Information the min max standard deviation the 25 percentile the 50th percentile the 75 percentile found in each column.

iv) Less difference found in 50 percentile and Mean value of pdate_month, day

The statistical figure I get to know by the data.describe() so many information the min max standard deviation the 25 percentile the 50th percentile the 75 percentile .Then by the help of correlation function I get to know the correlation of each columns with each other. From the heat map I can visualized to see them clearly that they are positive correlated or the negative correlated the dark side is show the negative correlation among each other the lighter side represent the positive correlation

among the each other. The z-score function computes the relative Z-score of the input data, relative to the sample mean and standard deviation

- **Correlation :**



i) Darker cell is having high value of correlation where light colour cell is having less value of correlation.

ii) Amt_loan30 and amt_loan90 named columns are showing high correlation with each other

iii) Medianamt_loan30 and medianamt_loan90 columns are also showing high correlation with each other

iv) Maximum column are showing correlation value in acceptable range for Machine learning model building

➢ Data Sources and their formats:

The data was collected for my internship company – Flip Robo Technologies in excel format. The sample data is provided to us from our client database. It is hereby given to us for this exercise. In order to improve the selection of customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers.

Also, my dataset was having 209593 rows and 37 columns including target. In this particular datasets I have object, float and integer types of data. The information about features is as follows.

Loading Database

```
url = 'D:/Bhushan Sharma/Internship/Project/Data file.csv'
df = pd.read_csv(url, parse_dates = ['pdate'])
df.shape
```

```
(209593, 37)
```

Variable Definition

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

msisdn: mobile number of user

aon: age on cellular network in days

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

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

rental30: Average main account balance over last 30 days

rental90: Average main account balance over last 90 days

last_rech_date_ma: Number of days till last recharge of main account

last_rech_date_da: Number of days till last recharge of data account

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

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

fr_ma_rech30: Frequency of main account recharged in last 30 days

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

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

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

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

fr_ma_rech90: Frequency of main account recharged in last 90 days

sumamnt_ma_rech90 Total amount of recharge in main account over last 90 days (in Indonasian Rupiah)

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

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

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

fr_da_rech30: Frequency of data account recharged in last 30 days

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

fr_da_rech90: Frequency of data account recharged in last 90 days

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

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

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

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

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

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

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

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

payback30: Average payback time in days over last 30 days

payback90: Average payback time in days over last 90 days

pcircle: telecom circle

pdate: date

```
df.info()
# All columns are having same non-null values mean,
# null values are not present in the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209593 entries, 0 to 209592
Data columns (total 36 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   label                 209593 non-null  int64
 1   msisdn                209593 non-null  object
 2   aon                   209593 non-null  float64
 3   daily_decr30          209593 non-null  float64
 4   daily_decr90          209593 non-null  float64
 5   rental30              209593 non-null  float64
 6   rental90              209593 non-null  float64
 7   last_rech_date_ma     209593 non-null  float64
 8   last_rech_date_da     209593 non-null  float64
 9   last_rech_amt_ma      209593 non-null  int64
 10  cnt_ma_rech30         209593 non-null  int64
 11  fr_ma_rech30          209593 non-null  float64
 12  sumamnt_ma_rech30     209593 non-null  float64
 13  medianamnt_ma_rech30  209593 non-null  float64
 14  medianmarechpreba130  209593 non-null  float64
 15  cnt_ma_rech90         209593 non-null  int64
 16  fr_ma_rech90          209593 non-null  int64
 17  sumamnt_ma_rech90     209593 non-null  int64
 18  medianamnt_ma_rech90  209593 non-null  float64
 19  medianmarechpreba190  209593 non-null  float64
 20  cnt_da_rech30         209593 non-null  float64
 21  fr_da_rech30          209593 non-null  float64
 22  cnt_da_rech90         209593 non-null  int64
 23  fr_da_rech90          209593 non-null  int64
 24  cnt_loans30           209593 non-null  int64
 25  amnt_loans30          209593 non-null  int64
```

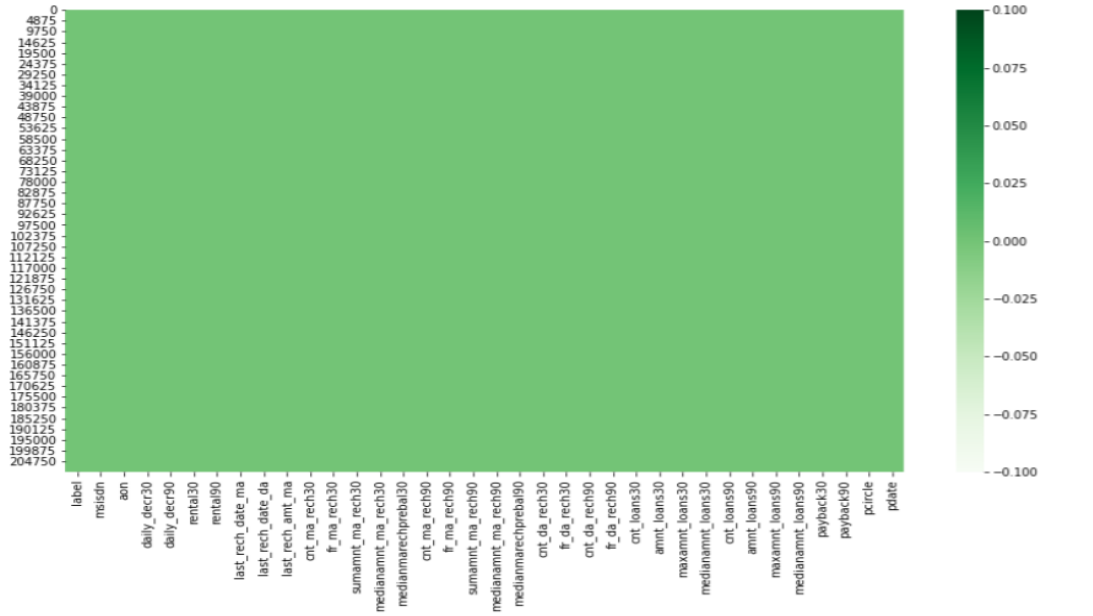
```
 26  maxamnt_loans30       209593 non-null  float64
 27  medianamnt_loans30    209593 non-null  float64
 28  cnt_loans90           209593 non-null  float64
 29  amnt_loans90          209593 non-null  int64
 30  maxamnt_loans90       209593 non-null  int64
 31  medianamnt_loans90    209593 non-null  float64
 32  payback30             209593 non-null  float64
 33  payback90             209593 non-null  float64
 34  pcircle               209593 non-null  object
 35  pdate                 209593 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(21), int64(12), object(2)
memory usage: 57.6+ MB
```

➢ Data Pre-processing:

i) I have checked presence of null value, and found not null values are present in the dataset

```
plt.figure(figsize = (15, 8))
sns.heatmap(df.isnull(), cmap = 'Greens')
```

```
<AxesSubplot:>
```



ii) Extraction of relevant information from the pdate column, I have extracted Day, month, and year from the pdate column

Extraction of relevent information from the pdate column (Day, month, year)

```
]: df['pdate_year'] = df['pdate'].dt.year
   df['pdate_month'] = df['pdate'].dt.month
   df['pdate_day'] = df['pdate'].dt.day
   df.shape

]: (209593, 38)
```

iii) After these steps, I have converted few columns into class based on their min and maximum value of the column. Those columns are listed below:

cnt_ma_reach30 column

```
# Loop for creating class of cnt_ma_rech30 class
cnt_ma_rech30_class = []
for i in df['cnt_ma_rech30']:
    if i in range (1, 51):
        cnt_ma_rech30_class.append('1-50 times')
    elif i in range (51, 101):
        cnt_ma_rech30_class.append('51-100 times')
    elif i in range(101,151):
        cnt_ma_rech30_class.append('101-150 times')
    elif i in range(151, 201):
        cnt_ma_rech30_class.append('151-200 times')
    elif i in range(201, 250):
        cnt_ma_rech30_class.append('200+ times')
    elif i == 0:
        cnt_ma_rech30_class.append('Zero times')
len(cnt_ma_rech30_class)
```

```
209593
```

```
df['new_cnt_ma_rech30'] = cnt_ma_rech30_class
```

last_reach_amt_ma

```
# Loop for creating class of tenure class
last_rech_amt_ma_class = []
for i in df['last_rech_amt_ma']:
    if i in range (1, 5001):
        last_rech_amt_ma_class.append('1-5000 Amt')
    elif i in range (5001, 10001):
        last_rech_amt_ma_class.append('5001-10000 Amt')
    elif i in range(10001,20001):
        last_rech_amt_ma_class.append('10001-20000 Amt')
    elif i in range(20001, 30001):
        last_rech_amt_ma_class.append('20001-30000 Amt')
    elif i in range(30001, 40001):
        last_rech_amt_ma_class.append('30001-40000 Amt')
    elif i in range(40001, 50001):
        last_rech_amt_ma_class.append('40001-50000 Amt')
    elif i in range(50001, 60000):
        last_rech_amt_ma_class.append('50001-60000 Amt')
    elif i == 0:
        last_rech_amt_ma_class.append('0 Amt')
len(last_rech_amt_ma_class)
```

```
209593
```

```
df['New_last_rech_amt_ma'] = last_rech_amt_ma_class
```

cnt_ma_reach90:

```
# Loop for creating class of tenure class
cnt_ma_rech90_class = []
for i in df['cnt_ma_rech90']:
    if i in range (1, 51):
        cnt_ma_rech90_class.append('1-50 Amt')
    elif i in range (51, 101):
        cnt_ma_rech90_class.append('51-100 Amt')
    elif i in range(101,151):
        cnt_ma_rech90_class.append('101-150 Amt')
    elif i in range(151, 201):
        cnt_ma_rech90_class.append('151-200 Amt')
    elif i in range(201, 251):
        cnt_ma_rech90_class.append('201-250 Amt')
    elif i in range(251, 301):
        cnt_ma_rech90_class.append('251-300 Amt')
    elif i in range(300, 350):
        cnt_ma_rech90_class.append('300+ ')
    elif i == 0:
        cnt_ma_rech90_class.append('0 Amt')
len(cnt_ma_rech90_class)
```

```
209593
```

```
df['cnt_ma_rech90_class'] = cnt_ma_rech90_class
```

fr_ma_reach90

```
# Loop for creating class of fr_ma_rech90 class
fr_ma_rech90_class = []
for i in df['fr_ma_rech90']:
    if i in range (0, 11):
        fr_ma_rech90_class.append('0-10 times')
    elif i in range (11, 31):
        fr_ma_rech90_class.append('11-30 times')
    elif i in range(31,51):
        fr_ma_rech90_class.append('31-50 times')
    elif i in range(51, 71):
        fr_ma_rech90_class.append('51-70 times')
    elif i in range(71, 91):
        fr_ma_rech90_class.append('71-91 times')

len(fr_ma_rech90_class)
```

```
209593
```

```
df['New_fr_ma_rech90'] = fr_ma_rech90_class
```

fr_da_reach90,

```
# Loop for creating class of fr_da_rech90 class
fr_da_rech90_class = []
for i in df['fr_da_rech90']:
    if i in range (0, 16):
        fr_da_rech90_class.append('0-15 times')
    elif i in range (16, 31):
        fr_da_rech90_class.append('16-30 times')
    elif i in range(31,46):
        fr_da_rech90_class.append('31-45 times')
    elif i in range(46, 61):
        fr_da_rech90_class.append('46-60 times')
    elif i in range(61, 76):
        fr_da_rech90_class.append('61-75 times')

len(fr_da_rech90_class)
```

```
209593
```

```
df['fr_da_rech90_class'] = fr_da_rech90_class
```

cnt_da_reach90,

```
# Loop for creating class of fr_da_rech90 class
cnt_da_rech90_class = []
for i in df['cnt_da_rech90']:
    if i in range (0, 11):
        cnt_da_rech90_class.append('0-10 times')
    elif i in range (11, 21):
        cnt_da_rech90_class.append('11-20 times')
    elif i in range(21,31):
        cnt_da_rech90_class.append('21-30 times')
    elif i in range(31, 41):
        cnt_da_rech90_class.append('31-40 times')

len(cnt_da_rech90_class)
```

```
209593
```

```
df['New_cnt_da_rech90'] = cnt_da_rech90_class
```

cnt_loan30

```
# Loop for creating class of fr_da_rech90 class
cnt_loans30_class = []
for i in df['cnt_loans30']:
    if i in range (0, 11):
        cnt_loans30_class.append('0-10 times')
    elif i in range (11, 21):
        cnt_loans30_class.append('11-20 times')
    elif i in range(21,31):
        cnt_loans30_class.append('21-30 times')
    elif i in range(31, 41):
        cnt_loans30_class.append('31-40 times')
    elif i in range(41, 51):
        cnt_loans30_class.append('41-50 times')

len(cnt_loans30_class)
```

```
209593
```

```
df['cnt_loans30_class'] = cnt_loans30_class
```

maxamnt_loan90

```
# Loop for creating class of fr_da_rech90 class
maxamnt_loans90_class = []
for i in df['maxamnt_loans90']:
    if i in range (0, 11):
        maxamnt_loans90_class.append('0 amt')
    elif i in range (11, 21):
        maxamnt_loans90_class.append('6 amt')
    elif i in range(21,31):
        maxamnt_loans90_class.append('12 amt')

len(maxamnt_loans90_class)
```

```
209593
```

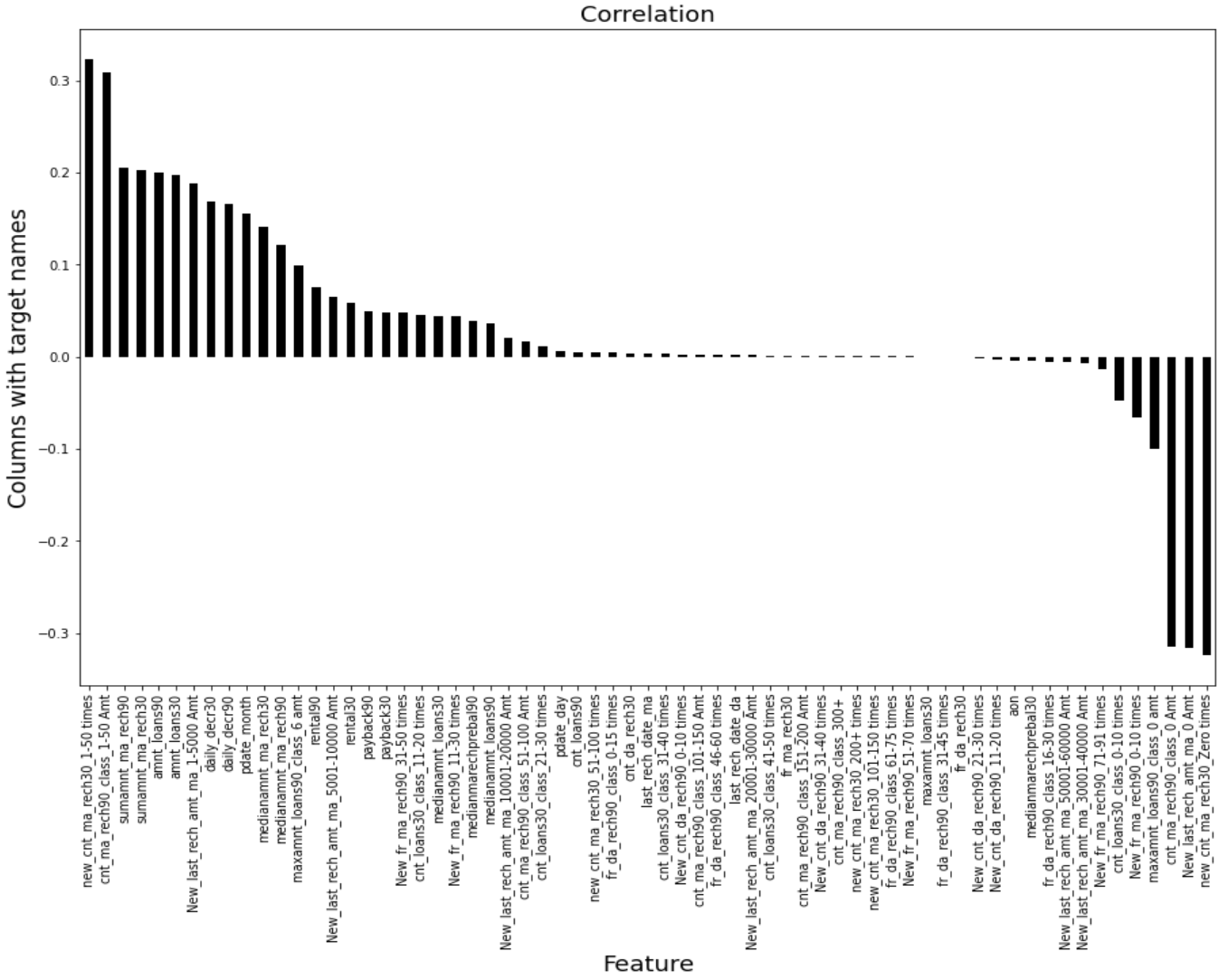
```
df['maxamnt_loans90_class'] = maxamnt_loans90_class
```

iv) After classifying these column I have deleted un necessary columns of dataset:

```
df.drop(columns = ['last_rech_amt_ma'], inplace = True)
df.drop(columns = ['cnt_ma_rech90'], inplace = True)
df.drop(columns = 'fr_ma_rech90', inplace = True)
df.drop(columns= ['fr_da_rech90'], inplace = True)
df.drop(columns = 'cnt_da_rech90', inplace = True)
df.drop(columns = 'cnt_loans30', inplace  = True)
```

```
df.drop(columns = ['maxamnt_loans90'], inplace = True)
df.drop(columns = 'pdate', inplace = True)
```

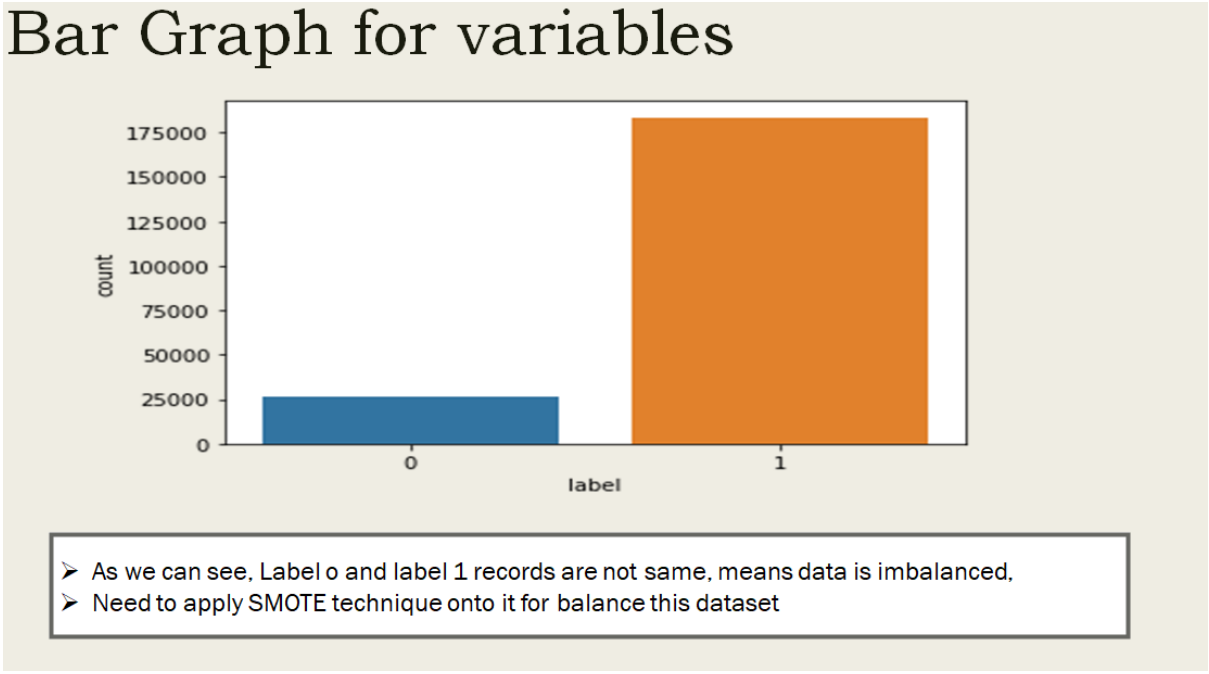
- **Data Inputs- Logic- Output Relationships**

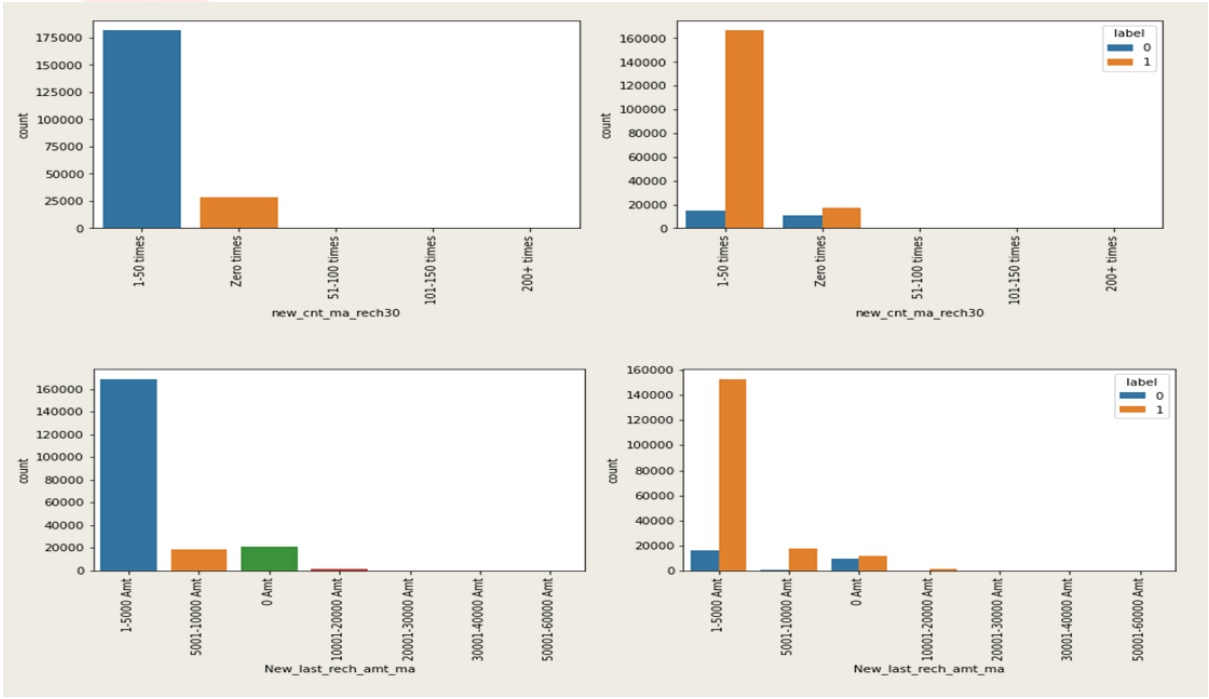


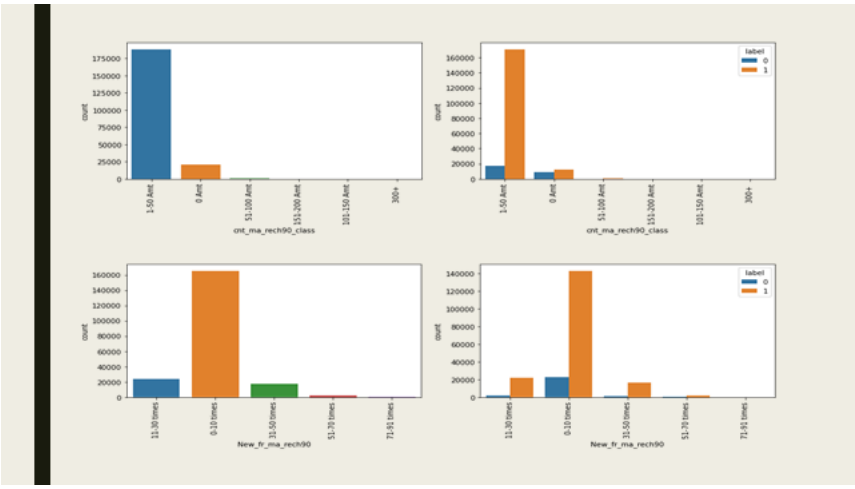
We can see in above diagram, high positive impact on target variable of “new_cnt_ma_reach30_1-50 times” and “cnt_ma_reach90_class_1-50 Amt” columns

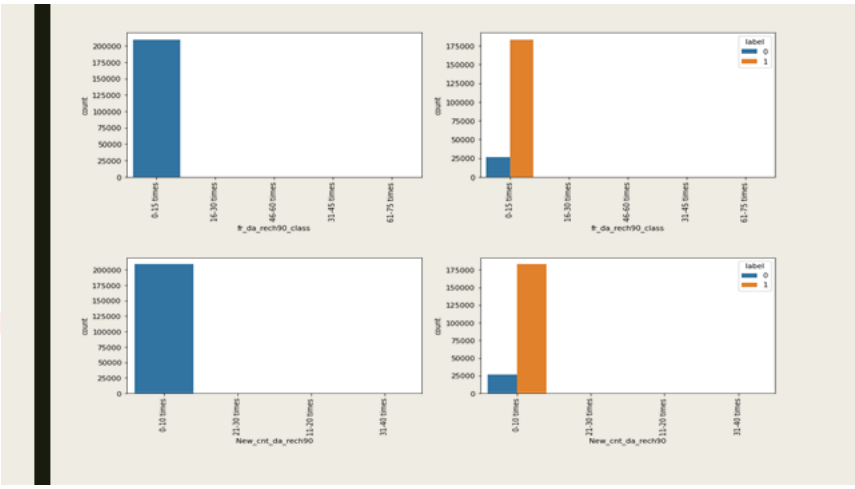
Accordingly we can also see, the high negative impact of “new_cnt_ma_reach30_Zero times” and “New_last_rech_amt_ma 0 amt” on target variable

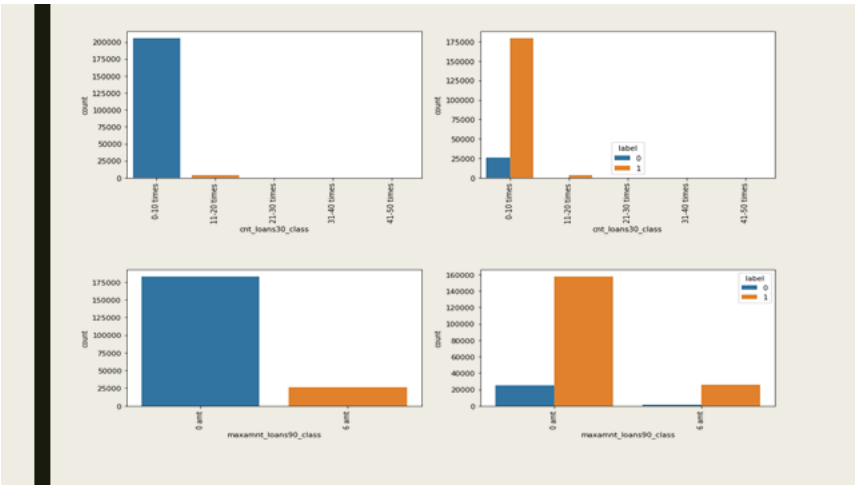
Visualisation:

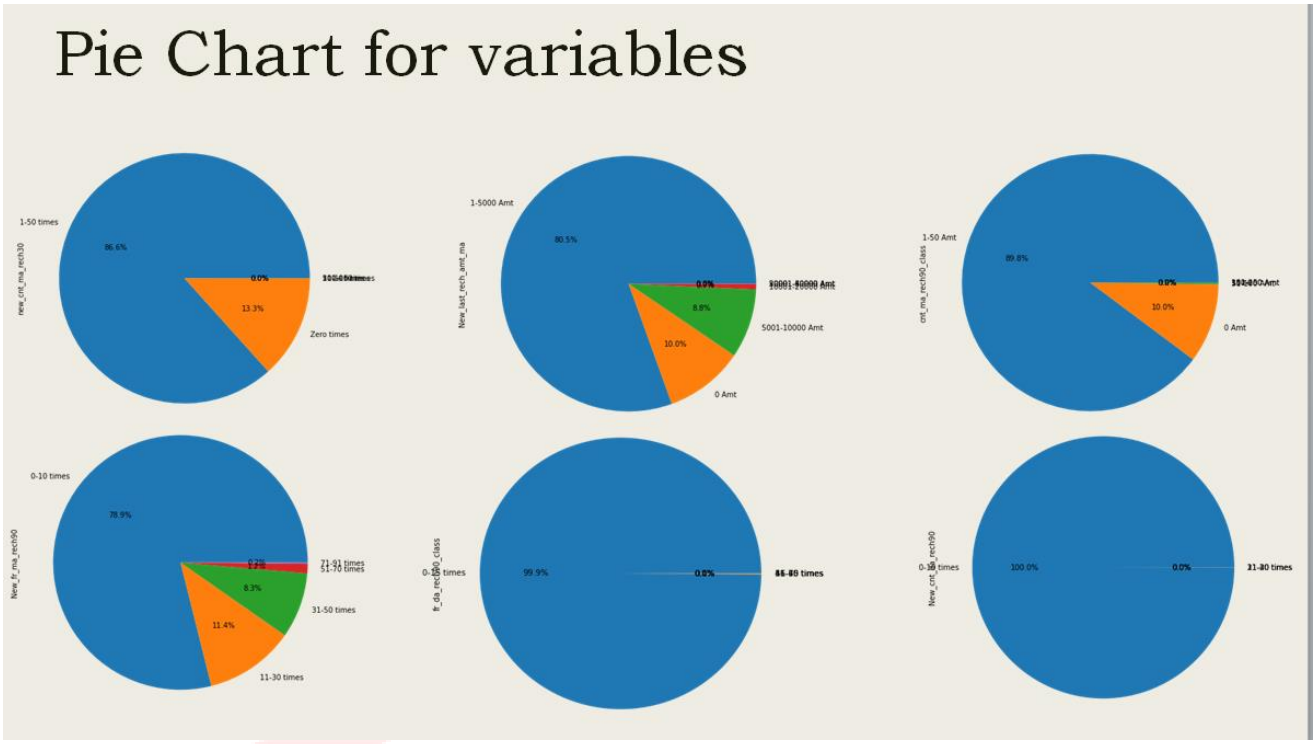


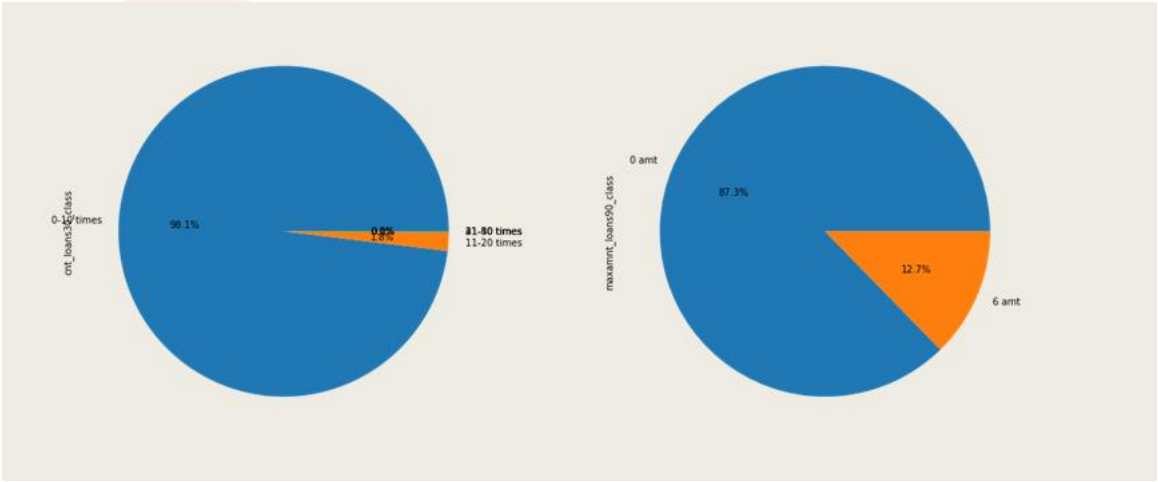



count
cnt_ma_rech90_class
label
0
1
1-50 Amt
0 Amt
51-100 Amt
151-200 Amt
101-150 Amt
300+
New_fr_ma_rech90
11-30 times
0-10 times
31-50 times
51-70 times
71-91 times


count
fr_da_rech90_class
label
0
1
0-15 times
16-30 times
46-60 times
31-45 times
61-75 times
New_cnt_da_rech90
0-10 times
21-30 times
11-20 times
31-40 times


count
cnt_loans30_class
label
0
1
0-10 times
11-20 times
21-30 times
31-40 times
41-50 times
maxamnt_loans90_class
0 amt
6 amt


Pie Chart for variables
1-50 times
86.6%
13.3%
Zero times
1-5000 Amt
80.5%
8.8%
5001-10000 Amt
10.0%
0 Amt
1-50 Amt
89.8%
10.0%
0 Amt
0-10 times
78.9%
8.3%
31-50 times
11.4%
11-30 times
99.9%
100.0%
0.0%


98.1%
11-20 times
0 amt
87.3%
12.7%
6 amt

➢ State the set of assumptions

OUTLIERS:

Outliers are not removed from the dataset because it was giving high loss of data; therefore I have worked on existing dataset without removing outliers

using zscore technique

```
from scipy.stats import zscore
```

```
# Zscore operation to remove outliers
z = np.abs(zscore(encoded_df) )
df_z = encoded_df[(z < 3).all(axis = 1)]
df_z.shape

# (123134, 66)
```

```
(123134, 66)
```

```
((encoded_df.shape[0] - df_z.shape[0] ) / encoded_df.shape[0] ) * 100

# As by this method we are getting total loss, we cannot use this method
# 41.25
```

```
41.25090055488494
```

Using IQR technique

```
Q1 = encoded_df.quantile(0.25)
Q3 = encoded_df.quantile(0.75)
IQR = Q3 - Q1
```

```
df_IQR = encoded_df[~((encoded_df < (Q1 - 1.5*IQR) ) | (encoded_df > (Q3 + 1.5*IQR) )).any(axis = 1) ]
df_IQR.shape
# (0, 138)
```

```
(45631, 66)
```

```
((encoded_df.shape[0] - df_IQR.shape[0] ) / encoded_df.shape[0] ) * 100

# Total data loss found by both methods zscore and IQR therefore we will leave data as it is
# 100.0
```

```
78.22875763980667
```

.

Seperating Data into x and y form ¶

```
# x is training dataset and y is target variable
x = encoded_df.drop(columns = ['label'])
y = encoded_df['label']

print(x.shape)
print(y.shape)

# (209593, 65)
# (209593,)
```

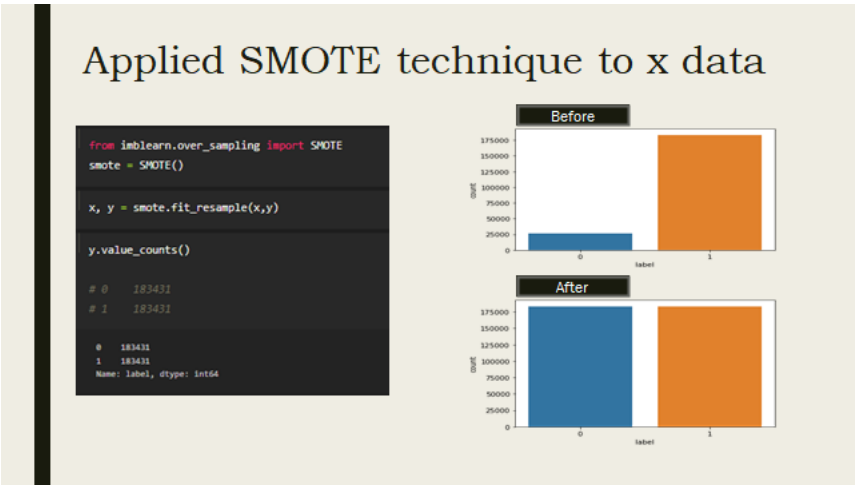
```
(209593, 65)
(209593,)
```

```
y.unique()
# array([0, 1], dtype=int64)
```

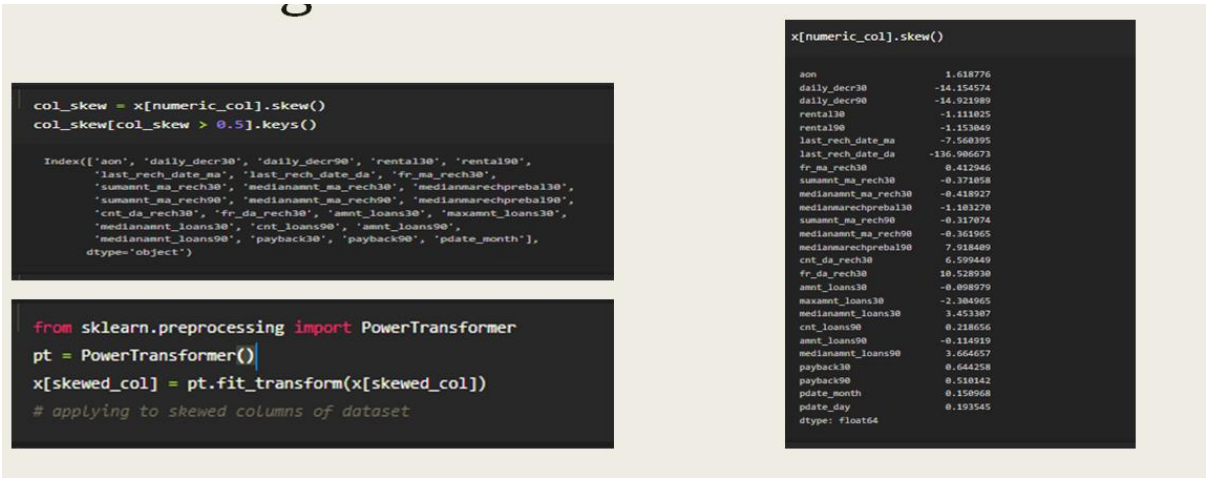
```
array([0, 1], dtype=int64)
```

Separating Dataset into x and y form, where x is feature and y is target variable

Removing Imbalances of dataset:



Treatment of Skewness of columns



```
col_skew = x[numeric_col].skew()
col_skew[col_skew > 0.5].keys()
```

```
Index(['aon', 'daily_decr30', 'daily_decr90', 'rental30', 'rental90',
       'last_rech_date_ma', 'last_rech_date_da', 'fr_ma_rech30',
       'sumamnt_ma_rech30', 'medianamnt_ma_rech30', 'medianmarechpreba130',
       'sumamnt_ma_rech90', 'medianamnt_ma_rech90', 'medianmarechpreba190',
       'cnt_da_rech30', 'fr_da_rech30', 'amnt_loans30', 'maxamnt_loans30',
       'medianamnt_loans30', 'cnt_loans90', 'amnt_loans90',
       'medianamnt_loans90', 'payback30', 'payback90', 'pdate_month'],
      dtype='object')
```

```
from sklearn.preprocessing import PowerTransformer
pt = PowerTransformer()
x[skewed_col] = pt.fit_transform(x[skewed_col])
# applying to skewed columns of dataset
```

```
x[numeric_col].skew()
```

```
aon                   1.618776
daily_decr30        -14.154574
daily_decr90        -14.921989
rental30             -1.111025
rental90             -1.153049
last_rech_date_ma    -7.560395
last_rech_date_da  -136.906673
fr_ma_rech30          0.412946
sumamnt_ma_rech30    -0.371058
medianamnt_ma_rech30 -0.418927
medianmarechpreba130 -1.103270
sumamnt_ma_rech90    -0.317074
medianamnt_ma_rech90 -0.361965
medianmarechpreba190  7.918489
cnt_da_rech30         6.599449
fr_da_rech30         10.528930
amnt_loans30         -0.898979
maxamnt_loans30      -2.304965
medianamnt_loans30    3.453387
cnt_loans90           0.218656
amnt_loans90         -0.114919
medianamnt_loans90    3.664657
payback30             0.644258
payback90             0.510142
pdate_month           0.150968
pdate_day             0.193545
dtype: float64
```

I have used applied many skewness treatment on this dataset in all applied skewness removing technique this PowerTransformer method is giving best result.

Removing Multicolinearity

Multicolinearity is removed using variance inflation factor,

Here cal_vif is a fuction to calculate vif value of column, this function is created using variance inflation factor imported from statsmodels library

```
cal_vif(x[numeric_col])
```

	Columns Name	VIF
0	aon	1.008185
1	daily_decr30	400.687764
2	daily_decr90	410.325419
3	rental30	19.079928
4	rental90	20.175668
5	last_rech_date_ma	1.014720
6	last_rech_date_da	1.015816
7	fr_ma_rech30	1.858745
8	sumamnt_ma_rech30	42.842021
9	medianamnt_ma_rech30	24.638785
10	medianmarechprebal30	1.290484
11	sumamnt_ma_rech90	32.704835
12	medianamnt_ma_rech90	15.994066
13	medianmarechprebal90	1.363557
14	cnt_da_rech30	1.105161
15	fr_da_rech30	1.087847
16	amnt_loans30	11.442356
17	maxamnt_loans30	1.778754
18	medianamnt_loans30	6.043714
19	cnt_loans90	18.850225
20	amnt_loans90	26.901073
21	medianamnt_loans90	6.031922
22	payback30	7.027642
23	payback90	6.729061
24	pdate_month	3.639056
25	pdate_day	1.033496

```
x.drop(columns = 'daily_decr90', inplace = True )
```

```
cal_vif(x[numeric_col])
```

	Columns Name	VIF
0	aon	1.008185
1	daily_decr30	4.775565
2	rental30	17.553816
3	rental90	18.464472
4	last_rech_date_ma	1.014632
5	last_rech_date_da	1.015815
6	fr_ma_rech30	1.858649
7	sumamnt_ma_rech30	40.782765
8	medianamnt_ma_rech30	23.962955
9	medianmarechprebal30	1.290481
10	sumamnt_ma_rech90	30.382945
11	medianamnt_ma_rech90	15.352285
12	medianmarechprebal90	1.363523
13	cnt_da_rech30	1.105109
14	fr_da_rech30	1.087826
15	amnt_loans30	11.275459
16	maxamnt_loans30	1.778726
17	medianamnt_loans30	6.040424
18	cnt_loans90	18.834640
19	amnt_loans90	26.660375
20	medianamnt_loans90	6.029768
21	payback30	7.014424
22	payback90	6.715125
23	pdate_month	3.632815
24	pdate_day	1.032028

```
x.drop(columns = 'sumamnt_ma_rech30', inplace = True )
```

```
cal_vif(x[numeric_col])
```

	Columns Name	VIF
0	aon	1.008173
1	daily_decr30	4.765544
2	rental30	17.312711
3	rental90	18.205886
4	last_rech_date_ma	1.014257
5	last_rech_date_da	1.015815
6	fr_ma_rech30	1.835296
7	medianamnt_ma_rech30	4.415399
8	medianmarechprebal30	1.290400
9	sumamnt_ma_rech90	10.135536
10	medianamnt_ma_rech90	7.295064
11	medianmarechprebal90	1.363498
12	cnt_da_rech30	1.104874
13	fr_da_rech30	1.087825
14	amnt_loans30	10.247949
15	maxamnt_loans30	1.749097
16	medianamnt_loans30	6.026380
17	cnt_loans90	18.826201
18	amnt_loans90	26.261176
19	medianamnt_loans90	6.017311
20	payback30	6.877509
21	payback90	6.653768
22	pdate_month	3.557753
23	pdate_day	1.031897

Scaling:

Scaling of column, this step apply just before applying Machine learning of dataset, by followed this I have applied this Standard Scaling technique to the dataset

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()

# x_scale = ss.fit_transform(x)
# x = pd.DataFrame(x_scale, columns = x.columns)
x[numeric_col] =  ss.fit_transform(x[numeric_col])
```

```
x.head(2)
```

	aon	daily_decr30	rental30	rental90	last_rech_date_ma	last_rech_date_da	fr_ma_rech30	medianamnt_ma_rech30	m
0	-0.148932	0.652316	-0.531417	-0.518099	-0.050519	-0.078231	1.683565	0.512358	-0
1	0.057822	1.392223	0.432241	0.260566	0.209411	-0.078231	-0.898008	1.542994	0.(

```
y.unique()
```

```
array([0, 1], dtype=int64)
```

➢ Model/s Development and Evaluation

In Machine Learning, I have applied 4 models in which I have selected one of them based on their performance and applied Hyper Parameter tuning technique over it.
And after applying these operation, then finalize the a final model for this dataset

In this machine learning process, I have created a function named "ML_Model" for smooth functioning of machine learning,
This function will give Training and testing data accuracy along with classification report, cv score and AUC-ROC score

Libraries used in process of Machine Learning :

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
```


Creating instances for ML model and storing them into a list of models.
After storing instances into model list, apply ML_Model function with parameter model and dataset (x, y)

```
lr = LogisticRegression()
abc = AdaBoostClassifier()
dtc = DecisionTreeClassifier()
gbc = GradientBoostingClassifier()


models = [lr,abc, dtc, gbc]
ML_Model(models, x, y)
```

Logistic Regression Model Performance

```
LogisticRegression()  is giving best accurancy 0.8285737649806013 on random state of 23
Training accuracy is :  0.8254187061677628
Testing accuracy is : 0.8285737649806013
--------------------------------------------------------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.82      0.83     56520
           1       0.81      0.84      0.83     53539

    accuracy                           0.83    110059
   macro avg       0.83      0.83      0.83    110059
weighted avg       0.83      0.83      0.83    110059

Confusion Matrix:
 [[46245 10275]
 [ 8592 44947]]
--------------------------------------------------------------------------------
Cross value score
cv score 0.8262071296563831 at 2 cross fold
cv score 0.8260218393782893 at 3 cross fold
cv score 0.8260818115678312 at 4 cross fold
cv score 0.8256211827515895 at 5 cross fold
cv score 0.8258228594076938 at 6 cross fold
cv score 0.8256619994309513 at 7 cross fold
--------------------------------------------------------------------------------
AUC-ROC Score
auc_score:  0.8286251604906663
```

AdaBoostClassifier Model Performance

```
AdaBoostClassifier()  is giving best accurancy 0.8749488910493463 on random state of 32
Training accuracy is :  0.8725209596461101
Testing accuracy is : 0.8749488910493463
--------------------------------------------------------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.88     56955
           1       0.86      0.89      0.87     53104

    accuracy                           0.87    110059
   macro avg       0.88      0.88      0.87    110059
weighted avg       0.88      0.87      0.87    110059

Confusion Matrix:
 [[48972  7983]
 [ 5780 47324]]
--------------------------------------------------------------------------------
Cross value score
cv score 0.8666555816628596 at 2 cross fold
cv score 0.8712350347248291 at 3 cross fold
cv score 0.8693433244862384 at 4 cross fold
cv score 0.8715158499246716 at 5 cross fold
cv score 0.871030610233178 at 6 cross fold
cv score 0.8706816628755096 at 7 cross fold
--------------------------------------------------------------------------------
AUC-ROC Score
auc_score:  0.8750466519558004
```

DecisionTreeClassifier Model Performance

```
DecisionTreeClassifier()  is giving best accurancy 0.911919970197803 on random state of 45
Training accuracy is :  0.9999883178934825
Testing accuracy is : 0.9118654539837724
--------------------------------------------------------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.91      0.91     55707
           1       0.91      0.92      0.91     54352

    accuracy                           0.91    110059
   macro avg       0.91      0.91      0.91    110059
weighted avg       0.91      0.91      0.91    110059

Confusion Matrix:
 [[50548  5159]
 [ 4541 49811]]
--------------------------------------------------------------------------------
Cross value score
cv score 0.9001286587327115 at 2 cross fold
cv score 0.9042065703880451 at 3 cross fold
cv score 0.9056049414355278 at 4 cross fold
cv score 0.9053133374526526 at 5 cross fold
cv score 0.9059620391499484 at 6 cross fold
cv score 0.9067661000075274 at 7 cross fold
--------------------------------------------------------------------------------
AUC-ROC Score
auc_score:  0.9118592796039544
```

GradientBoostingClassifier Model Performance

```
GradientBoostingClassifier()  is giving best accurancy 0.9072497478625101 on random state of 28
Training accuracy is :  0.9064457969727768
Testing accuracy is : 0.9072497478625101
--------------------------------------------------------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.90      0.91     55515
           1       0.90      0.91      0.91     54544

    accuracy                           0.91    110059
   macro avg       0.91      0.91      0.91    110059
weighted avg       0.91      0.91      0.91    110059

Confusion Matrix:
 [[50129  5386]
 [ 4822 49722]]
--------------------------------------------------------------------------------
Cross value score
cv score 0.8982969072839379 at 2 cross fold
cv score 0.900038809854932 at 3 cross fold
cv score 0.9007829682714582 at 4 cross fold
cv score 0.8996927004551327 at 5 cross fold
cv score 0.9010937132153534 at 6 cross fold
cv score 0.9010255119166087 at 7 cross fold
--------------------------------------------------------------------------------
AUC-ROC Score
auc_score:  0.9072568693098981
```

Based on model performance, we can see, GradientBoosting is performing best as compare to other ML Models, Its cv value is also very close to accuracy of model and it is also giving very near value of AUC ROC score.
Hence I had applied hyper parameter tuning for this Model to increase the accuracy of model

Hyper Parameter Tuning

```
parameter = {'loss' : ['deviance', 'exponential'],
             'learning_rate' : [0.1, 0.01],
             'criterion' : ['friedman_mse', 'squared_error'],
             'n_estimators': [100, 50, 10]  }
```

```
gcv = GridSearchCV(estimator = GradientBoostingClassifier(), param_grid = parameter, cv = 3)
gcv.fit(x_train, y_train)
```

```
GridSearchCV(cv=3, estimator=GradientBoostingClassifier(),
             param_grid={'criterion': ['friedman_mse', 'squared_error'],
                         'learning_rate': [0.1, 0.01],
                         'loss': ['deviance', 'exponential'],
                         'n_estimators': [100, 50, 10]})
```

```
gcv.best_params_
```

```
{'criterion': 'friedman_mse',
 'learning_rate': 0.1,
 'loss': 'exponential',
 'n_estimators': 100}
```



```
ensemble_gbc = GradientBoostingClassifier(criterion = 'friedman_mse', learning_rate = 0.1, loss = 'exponential', n_estimators= 100 )
models = [ensemble_gbc]
ML_Model(models, x, y)
```

```
GradientBoostingClassifier(loss='exponential')  is giving best accurancy 0.9075223289326634 on random state of 39
Training accuracy is :  0.9060213471026429
Testing accuracy is : 0.9075223289326634
--------------------------------------------------------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.90      0.91     56050
           1       0.90      0.91      0.91     54009

    accuracy                           0.91    110059
   macro avg       0.91      0.91      0.91    110059
weighted avg       0.91      0.91      0.91    110059

Confusion Matrix:
 [[50498  5552]
 [ 4626 49383]]
--------------------------------------------------------------------------------
Cross value score
cv score 0.8986131024745 at 2 cross fold
cv score 0.898294283521366 at 3 cross fold
cv score 0.8996899146978862 at 4 cross fold
cv score 0.8995018949072137 at 5 cross fold
cv score 0.8990875095744556 at 6 cross fold
cv score 0.8997334788626585 at 7 cross fold
--------------------------------------------------------------------------------
AUC-ROC Score
auc_score:  0.9075076077105115
```

I had observed that, is not increase accuracy but it is decreasing the value of cv, and for selecting best model, I suppose to choose that model which is having least cv and accuracy difference,

Therefore, I had selected GradientBoostingClassifier model without parameter (Hyper Parameter) where we are getting least difference between accuracy and cv score

Final Model (GradientBoostingClassifier)

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.30, random_state = 28)
final_model = GradientBoostingClassifier()
final_model.fit(x_train, y_train)
predict_train = final_model.predict(x_train)
predict_test = final_model.predict(x_test)

training = accuracy_score(predict_train, y_train)
testing = accuracy_score(predict_test, y_test)

print('At random state', i, 'the training accuracy is :', training)
print('At random state', i, 'the testing accuracy is :'  , testing)
print('_____________________________________________________________')
print('Classification Report: \n', classification_report(predict_test, y_test) )
print('Confusion Matrix: \n', confusion_matrix(predict_test, y_test) )
print('_____________________________________________________________')
print('Cross value score')

# perform cross-validation
cv_score = cross_val_score(GradientBoostingClassifier(), x, y, cv = 6 ).mean()
print('cv score',  cv_score ,'at', i, 'cross fold')
print('-------------------------------------------------------------------------------')
print('AUC-ROC Score')
auc_score = roc_auc_score( y_test, predict_test)
print('auc_score: ', auc_score)
```

```
At random state 7 the training accuracy is : 0.905554062841945
At random state 7 the testing accuracy is : 0.9077131356817707
_____________________________________________________________
Classification Report:
               precision    recall  f1-score   support

           0       0.91      0.90      0.91     55494
           1       0.90      0.91      0.91     54565

    accuracy                           0.91    110059
   macro avg       0.91      0.91      0.91    110059
weighted avg       0.91      0.91      0.91    110059

Confusion Matrix:
 [[50144  5350]
 [ 4807 49758]]
_____________________________________________________________
Cross value score
cv score 0.9013008720002693 at 7 cross fold
-------------------------------------------------------------------------------
AUC-ROC Score
auc_score:  0.9077199858842878
```

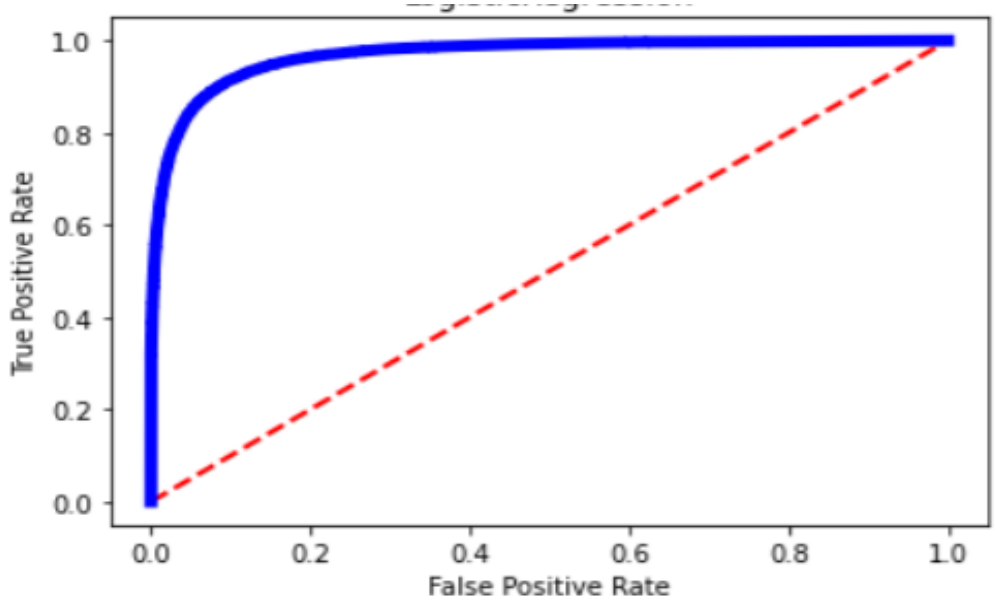
As it accuracy of training and testing dataset is almost same, therefore we can say, model is neither over fitted nor under fitted.

AUC-ROC Curve:

```
final_pred_prob = final_model.predict_proba( x_test)[:, 1]  #  probability of getting 1
```

```
fpr, tpr, threshols = roc_curve(y_test, final_pred_prob)
#  By the use of fpr and tpr we create AUC ROC curve
```

```
# fpr
# tpr
# threshols
```




```
auc_score = roc_auc_score(y_test, final_model.predict(x_test))
auc_score
```

```
0.9077199858842878
```

Deploying & Loading Model:

Deploy Model

```
import pickle
filename = 'mcd_model.pkl'                # model name
pickle.dump(final_model, open(filename, 'wb'))         # operation to deploy model
```

Loading model

```
load_model =  pickle.load(open('mcd_model.pkl', 'rb'))    # loading deployed model
result = load_model.score(x_test, y_test)
print(result)
```

```
0.9077131356817707
```

Conclusion:

```
original = np.array(y_test)
predicted = np.array(load_model.predict(x_test))
# convert columns in to np.array
```

```
print(predicted.shape)
print(original.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(110059,)
(110059,)
(110059, 63)
(110059,)
```

```
conclusion = pd.DataFrame({'Original fraud_reported': original, 'Predicted fraud_reported': predicted}, index = range(len(original)))
# Dataframe creation
```

```
conclusion.sample(10)
```

	Original fraud_reported	Predicted fraud_reported
109134	0	0
8189	1	1
91611	1	1
50954	0	0
96633	1	1
17250	1	1
91788	1	1
4980	0	0
18754	0	0
23627	0	0

➢ Hardware and Software Requirements and Tools Used

All used libraries :

```
# Importing important Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import PowerTransformer
from imblearn.over_sampling import SMOTE
from scipy.stats import zscore
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.filterwarnings('ignore')
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
```

Pandas: This library used for dataframe operations

Numpy: This library gives statistical computation for smooth functioning

Matplotlib: Used for visualization

Seaborn: This library is also used for visualization

Sklearn: This library having so many machine learning module and we can import them from this library

Pickle: This is used for deploying the model

Imblearn: This library is import to get SMOTE technique for balance the data

Scipy: It is import to perform outlier removing technique using zscore

Warning: To avoid unwanted warning shows in the output

I am giving this requirement and tool used, based on my laptop configuration.

Operating System:	**Window 11**
RAM:	**8 GB**
Processor :	**i5 10th Generation**
Software:	**Jupyter Notebook,**

➢ Observations from the whole problem.

i) Data was imbalanced, then applied SMOTE technique to over from this
ii) Multicolinearity was present in the dataset
iii) Column was so much skewed, applied PowerTransformer technique to over this issue.
iv) Columns are having very large value, which may create issue in machine learning , then applied scaling to overcome this issue

➢ Learning Outcomes of the Study in respect of Data Science

My learnings :- the power of visualization is helpful for the understanding of data into the graphical representation its help me to understand that what data is trying to say, Data cleaning is one of the most important step to remove missing value or null value.

Various algorithms I have used in this dataset and to get out best result and save that model. The best algorithm is GradientBoostingClassifier.

The challenges I faced while working on this project is execution time

As it is very large dataset, therefore It was taking so much time for machine learning

.

➢ Limitations of this work and Scope for Future Work

Not as such limitation found for this model.