FLIP ROBO

Microcredit Defaulter

Submitted by:

Dheerajkumar Pittala

ACKNOWLEDGMENT

- Firstly, I would like to thank FlipRobo for giving this opportunity. Being a fresher, it was a wonderful opportunity to work on such a project, where it was so much to explore.
- Secondly, I would also like to thank SME Shubham Yadav. He provided us with all the updates and responded quickly whenever the ticket was raised.

- I referred to many articles that help me initially with the project building:
  1. Articles about loan defaulter on www.analyticsvidya.com were of great help.
  2. I refer to www.towardsdatascience.com for the approach one should have towards such project.
  3. Whenever I was stuck with coding, I refer to www.stacksoverflow.com

INTRODUCTION

- ## Business Problem Framing

A Microfinance Institution (MFI) is an organization that offers financial services to low-income populations. MFS becomes very useful when targeting especially the unbanked poor families living in remote areas with not much sources of income.

They understand the importance of communication and how it effects a person's life and lack of communication can cause lot of uncertain problems, thus, focusing on providing their services and products to low-income families and poor customers that can help them in the need of hour.

- ## Conceptual Background of the Domain Problem

MFS 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).

- ## Review of Literature

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 are very useful in such areas, the implementation of MFS has been uneven with both significant challenges and successes.

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

- ## Motivation for the Problem Undertaken

We understand the importance of communication and how it effects a person's life and lack of communication can cause lot of uncertain problems, so we want to work to bridge this gap between people.

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

Analytical Problem Framing

- ## Mathematical/ Analytical Modelling of the Problem

We first investigate the statistics of data shown in fig 1.

```
df.describe()
```

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000
mean	104797.000000	0.875177	93100.650179	8112.343445	5381.402289	6082.515068	2692.581910	3483.406534	3755.847800
std	60504.431823	0.330519	53758.461427	75696.082531	9220.623400	10918.812767	4308.586781	5770.461279	53905.892230
min	1.000000	0.000000	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000
25%	52399.000000	1.000000	46506.000000	246.000000	42.440000	42.692000	280.420000	300.260000	1.000000
50%	104797.000000	1.000000	93073.000000	527.000000	1469.175667	1500.000000	1083.570000	1334.000000	3.000000
75%	157195.000000	1.000000	139626.000000	982.000000	7244.000000	7802.790000	3356.940000	4201.790000	7.000000
max	209593.000000	1.000000	186242.000000	999860.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733

```
df.describe()
```

last_rech_date_da	last_rech_amt_ma	cnt_ma_rech30	fr_ma_rech30	sumamnt_ma_rech30	medianamnt_ma_rech30	medianmarechprebal30	cnt_ma_rech90
209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.00000
3712.202921	2064.452797	3.978057	3737.355121	7704.501157	1812.817952	3851.927942	6.31543
53374.833430	2370.786034	4.256090	53643.625172	10139.621714	2070.864620	54006.374433	7.19347
-29.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-200.000000	0.00000
0.000000	770.000000	1.000000	0.000000	1540.000000	770.000000	11.000000	2.00000
0.000000	1539.000000	3.000000	2.000000	4628.000000	1539.000000	33.900000	4.00000
0.000000	2309.000000	5.000000	6.000000	10010.000000	1924.000000	83.000000	8.00000
999171.809410	55000.000000	203.000000	999606.368132	810096.000000	55000.000000	999479.419319	336.00000

```
df.describe()
```

fr_ma_rech90	sumamnt_ma_rech90	medianamnt_ma_rech90	medianmarechprebal90	cnt_da_rech30	fr_da_rech30	cnt_da_rech90	fr_da_rech90	cnt_loans3
209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.00000
7.716780	12396.218352	1864.595821	92.025541	262.578110	3749.494447	0.041495	0.045712	2.75898
12.590251	16857.793882	2081.680664	369.215658	4183.897978	53885.414979	0.397556	0.951386	2.55450
0.000000	0.000000	0.000000	-200.000000	0.000000	0.000000	0.000000	0.000000	0.00000
0.000000	2317.000000	773.000000	14.600000	0.000000	0.000000	0.000000	0.000000	1.00000
2.000000	7226.000000	1539.000000	36.000000	0.000000	0.000000	0.000000	0.000000	2.00000
8.000000	16000.000000	1924.000000	79.310000	0.000000	0.000000	0.000000	0.000000	4.00000
88.000000	953036.000000	55000.000000	41456.500000	99914.441420	999809.240107	38.000000	64.000000	50.00000

Fig 1 Statastical decription of data

From this statistical analysis we make some of the interpretations that,

1. Maximum standard deviation is observed in aon column.

2. In the columns aon, daily_decr30, daily_decr90, rental30, rental90, last_rech_date_ma, last_rech_date_da, maxamnt_loans30, cnt_loans90, amnt_loans90 mean is considerably greater than median so the columns are positively skewed.

3. In the columns label, month median is greater than mean so the columns are negatively skewed.

4. In the columns aon, daily_decr30, daily_decr90, rental30, rental90, last_rech_date_ma, last_rech_date_da, maxamnt_loans30, cnt_loans90, payback30, payback90 there is huge difference present between 75th percentile and maximum so outliers are present here.

We look for the skewness present in data shown in fig 2,

```
df.skew()
```

```
label                  -2.270254
aon                    10.392949
daily_decr30            3.946230
daily_decr90            4.252565
rental30                4.521929
rental90                4.437681
last_rech_date_ma      14.790974
last_rech_date_da      14.814857
last_rech_amt_ma        3.781149
cnt_ma_rech30           3.283842
fr_ma_rech30           14.772833
sumamnt_ma_rech30       6.386787
medianamnt_ma_rech30    3.512324
medianmarechprebal30   14.779875
cnt_ma_rech90           3.425254
fr_ma_rech90            2.285423
sumamnt_ma_rech90       4.897950
medianamnt_ma_rech90    3.752706
medianmarechprebal90   44.880503
cnt_da_rech30          17.818364
fr_da_rech30           14.776430
cnt_da_rech90          27.267278
fr_da_rech90           28.988083
cnt_loans30             2.713421
amnt_loans30            2.975719
maxamnt_loans30        17.658052
medianamnt_loans30      4.551043
cnt_loans90            16.594408
amnt_loans90            3.150006
maxamnt_loans90         1.678304
medianamnt_loans90      4.895720
payback30               8.310695
payback90               6.899951
day                     0.199845
month                   0.343242
```

Fig 2 skewness in data

We observe skewness in the data due to outliers, so we remove the 7-8% outliers through z score method by keeping standard deviation 5 and treat the rest outliers through winsorization technique. Now the skewness observed is shown in fig 3,

```
df_cap.skew()
```

```
label                   -2.242737
aon                      0.495635
daily_decr30             1.072841
daily_decr90             1.133561
rental30                 1.095992
rental90                 1.125867
last_rech_amt_ma         0.850541
cnt_ma_rech30            0.657301
sumamnt_ma_rech30        0.691258
medianamnt_ma_rech30     0.949679
medianmarechprebal30     1.311814
cnt_ma_rech90            0.709201
fr_ma_rech90             1.574587
sumamnt_ma_rech90        0.787981
medianamnt_ma_rech90     0.988311
medianmarechprebal90     1.232058
cnt_da_rech30            0.000000
cnt_da_rech90            0.000000
fr_da_rech90             0.000000
cnt_loans30              0.892197
amnt_loans30             0.789402
maxamnt_loans30          1.490262
medianamnt_loans30       0.000000
cnt_loans90              0.928602
amnt_loans90             1.006262
maxamnt_loans90          2.374270
medianamnt_loans90       0.000000
payback30                0.941894
payback90                0.954838
day                      0.093845
month                    0.381182
dtype: float64
```

Fig3 Skewness observed after trating outliers through winsorization

- ## Data Sources and their formats

The variable features of this problem statement are:-

Variable : Defination -> comment

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 -> Unsure of given definition

rental90: Average main account balance over last 90 days -> Unsure of given definition

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 -> Unsure of given definition

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

medianamnt_ma_rech30: Median of number 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 -> Unsure of given definition

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 -> There are only two options: 5 & 10 Rs., for which the user needs to pay back 6 & 12 Rs. respectively

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

The data types of features are shown in fig 4,

```
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                   object
```

Fig 4 Data types of features

- Data Pre-processing Done

We first done data cleaning. In data cleaning we done feature extraction, we extracted the features day and month from pdate column as shown in fig 5,

Feature extraction

```
df['pdate'] = pd.to_datetime(df['pdate'])
df['pdate']
```

```
0         2016-07-20
1         2016-08-10
2         2016-08-19
3         2016-06-06
4         2016-06-22
             ...
209588    2016-06-17
209589    2016-06-12
209590    2016-07-29
209591    2016-07-25
209592    2016-07-07
Name: pdate, Length: 209593, dtype: datetime64[ns]
```

```
df['pdate'].dt.day
```

```
0         20
1         10
2         19
3          6
4         22
          ..
209588    17
209589    12
209590    29
209591    25
209592     7
Name: pdate, Length: 209593, dtype: int64
```

```
df['day'] = df['pdate'].dt.day
```

```
df['pdate'].dt.month
```

```
0         7
1         8
2         8
3         6
4         6
         ..
209588    6
209589    6
209590    7
209591    7
209592    7
Name: pdate, Length: 209593, dtype: int64
```

```
df['month'] = df['pdate'].dt.month
```

Fig 5 Feature extraction

AS we can see we extracted day and month from pdate column, we won’t be needing year as there is only one unique value of year present in the dataset i.e 2016 as shown in fig 6.

```python
from collections import Counter
Counter(df['pdate'])
```

```
Counter({'2016-07-20': 2842,
         '2016-08-10': 2178,
         '2016-08-19': 1132,
         '2016-06-06': 2631,
         '2016-06-22': 2906,
         '2016-07-02': 2910,
         '2016-07-05': 3127,
         '2016-08-05': 2298,
         '2016-06-15': 3033,
         '2016-06-08': 2580,
         '2016-06-12': 2936,
         '2016-06-20': 3099,
         '2016-06-29': 2832,
         '2016-06-16': 2824,
         '2016-08-03': 2213,
         '2016-06-24': 2785,
         '2016-07-04': 3150,
         '2016-07-03': 2905,
         '2016-07-01': 2954,
         '2016-08-08': 2428,
         '2016-06-26': 2901,
         '2016-06-23': 2964,
         '2016-07-06': 3041,
         '2016-07-09': 2922,
         '2016-06-10': 2858,
         '2016-06-07': 2502,
         '2016-06-27': 2999,
         '2016-08-11': 2157,
         '2016-06-30': 2822,
         '2016-06-19': 2833,
         '2016-07-26': 2273,
         '2016-08-14': 1951,
         '2016-06-14': 2945,
         '2016-06-21': 2890,
         '2016-06-25': 2956,
         '2016-06-28': 2664,
         '2016-06-11': 2915,
         '2016-07-27': 2284,
         '2016-07-23': 2852,
         '2016-08-16': 1893,
         '2016-08-15': 1879,
         '2016-06-02': 2577,
         '2016-06-05': 2564,
         '2016-08-02': 2352,
         '2016-07-28': 2233,
         '2016-07-18': 2926,
         '2016-08-18': 1407,
         '2016-07-16': 2839,
         '2016-07-29': 2245,
         '2016-07-21': 2750,
         '2016-06-03': 2489,
         '2016-06-13': 2897,
         '2016-08-01': 2335,
         '2016-07-13': 2953,
         '2016-07-10': 3014,
         '2016-06-09': 2604,
         '2016-07-15': 2908,
         '2016-07-11': 3020,
         '2016-08-09': 2191,
         '2016-08-12': 2130,
         '2016-07-22': 2847,
         '2016-06-04': 1559,
         '2016-07-24': 2318,
         '2016-06-18': 2972,
         '2016-08-13': 2119,
         '2016-06-17': 3082,
         '2016-08-07': 2408,
         '2016-07-12': 2962,
         '2016-08-06': 2358,
         '2016-07-19': 2892,
         '2016-08-21': 324,
         '2016-08-04': 2445,
         '2016-07-25': 2313,
         '2016-07-30': 2307,
         '2016-08-17': 1688,
         '2016-07-08': 2891,
         '2016-07-14': 2920,
         '2016-06-01': 2535,
         '2016-07-07': 3116,
         '2016-07-17': 2873,
         '2016-07-31': 2178,
         '2016-08-20': 788})
```

Fig 6 Unique values of pdate column

We then explored categorical variables as shown in fig 6.

Exploring categorical columns

```
for column in df.columns:
    if df[column].dtypes == object:
        print(str(column) + ' : ' + str(df[column].unique()))
        print(df[column].value_counts())
        print('*****************************************************************************************************')
        print('\n')
```

```
msisdn : ['21408I70789' '76462I70374' '17943I70372' ... '22758I85348' '59712I82733'
 '65061I85339']
47819I90840    7
04581I85330    7
43096I88688    6
94119I84456    6
22038I88658    6
              ..
71605I88649    1
70877I82736    1
18632I70379    1
04889I70375    1
11685I89234    1
Name: msisdn, Length: 186243, dtype: int64
*****************************************************************************************************


pcircle : ['UPW']
UPW    209593
Name: pcircle, dtype: int64
*****************************************************************************************************
```

Fig 6 Exploring categorical variables

We observed that there is only one unique value present in pcircle column which is 'UPW' so will be dropping this column. Then we observed that column msisdn was present in categorical column so we encode it to numbers using label encoder as shown in fig 7, to check it's correlation with other feature variables and target varaible.

Encoding categorical column

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['msisdn']=le.fit_transform(df['msisdn'].astype(str))
```

```
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_rech30	fr
0	1	0	40191	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	2	
1	2	1	142291	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	5787	1	
2	3	1	33594	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539	1	
3	4	1	104157	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	947	0	
4	5	1	6910	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309	7	

Fig 7 Encoding column msisdn

Then we checked the correlation with the help of heatmap as shown in fig 8,

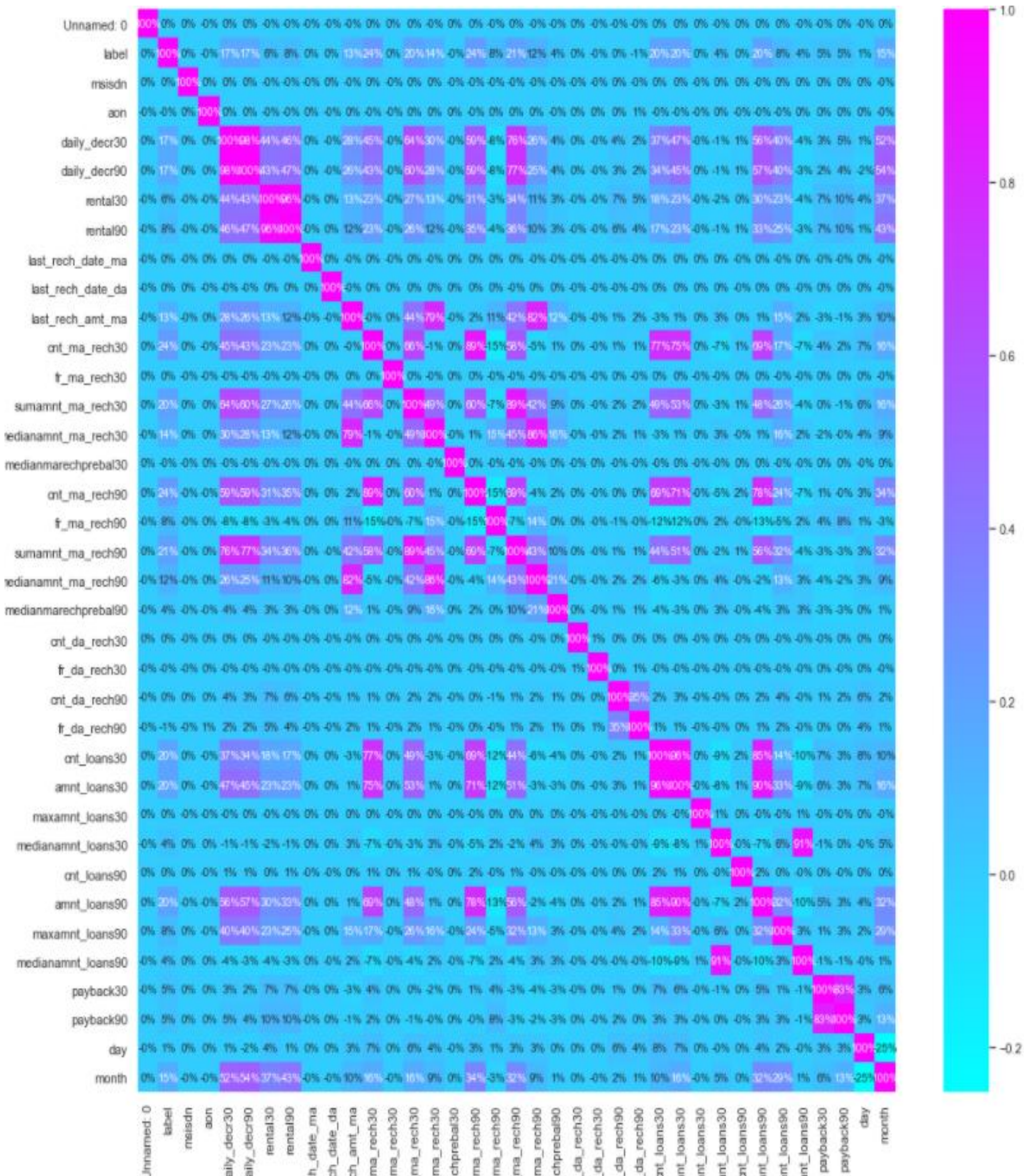


Fig 9 Heatmap of correlation

While checking the heatmap of correlation we observed that there exists multicollinearity in between columns.

We also observed that no correlation was present in unnamed: 0, msisdn, last_rechdate_ma, last_rechdate_da columns so we will be dropping these columns.

We then removed the outliers from the dataset through zscore method.

- Data Inputs- Logic- Output Relationships

Here we check the correlation between all our feature variables with target variable label as shown in fig 10.

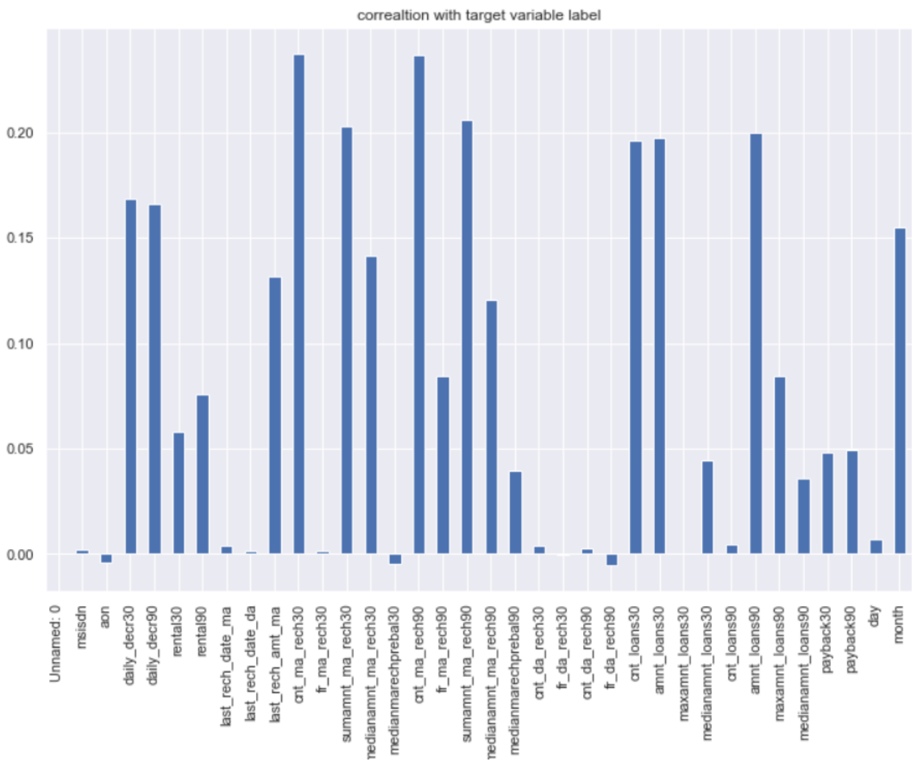


Fig 10 correlation with target variable label

We observe that the columns cnt_ma_rech30 and cnt_ma_rech90 are highly positively correlated with label this means as the cnt_ma_rech30 and cnt_ma_rech90 are increasing the probability of cutomer being non-fraudulent is also increasing.

We also observe that the columns aon, medianmarechprebal30 and fr_da_rech90 are negatively correlated with label this means as the aon, medianmarechprebal30 and fr_da_rech90 are increasing the probability of customer being fraudulent is also increasing.

- Set of assumptions related to the problem under consideration

  By looking into the target vaariable label we assumed that it was a classification type of problem.

  We observed multicollinearity in between columns so we assumed that we will be using Principal Component Analysis (PCA).

  We also observed that only one single unique value was present in pcircle and in year in pdate column and in Unnamed: 0 all the numbers were unique without any correlation so we assumed that we will be dropping these columns.

- Hardware and Software Requirements and Tools Used

  This project was done on laptop with i5 processor with quad cores and eight threads with 8gb of ram and latest GeForce GTX 1650 GPU on Anaconda, jupyter notebook.

  The tools, libraries and packages we used for accomplishing this project are pandas, numpy, matplotlib, seaborn, scipy stats, sklearn.decomposition pca, sklearn standardscaler, collections counter, imblearn SmoteTomek, GridSearchCV, joblib.

  Through pandas library we loaded our csv file 'Data file' into dataframe and performed data manipulation and analysis. Through pandas library we converted pdate column to datetime format from which we were able to extract day and month column.

  With the help of numpy we worked with arrays.

  With the help of matplotlib and seaborn we did plot various graphs and figures and done data visualization.

  With scipy stats we treated outliers through winsorization technique.

  With sklearn.decomposition's pca package we reduced the number of feature variables from 34 to 7 by plotting scrre plot with their

Eigenvalues and chose the number of columns on the basis of their nodes.

With sklearn’s standardscaler package we scaled all the feature variables onto single scale.

With collection’s counter package we were able to display all the unique values of the pdate column.

Through imblearn’s SmoteTomek package we were able to handle the imbalanced data by increasing the number of fraudulent transactions on relevant data points.

Through GridSearchCV we were able to find the right parameters for hyperparameter tuning.

Through joblib we saved our model in csv format.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

We first converted all our categorical variables to numeric variables with the help of label encoder to check out the correlation between them and dropped the columns which we felt were unnecessary.

We observed skewness in data, so we tried to remove the skewness through treating outliers with winsorization technique as shown in fig 3.

The data was imbalanced so through imblearn’s SmoteTomek package we were able to handle the imbalanced data by increasing the number of fraudulent transactions on relevant data points.

The data was improper scaled, so we scaled the feature vaariables on a single scale using sklearn’s StandardScaler package.

There were too many (37) feature variables in the data so we reduced it to 7 with the help of Principal Component Analysis (PCA) by plotting Eigenvalues and taking the number of nodes as our number of feature variables.

- ## Testing of Identified Approaches (Algorithms)

The algorithms we used for the training and testing are as follows: -

- Extreme gradient boosting classifier
- Decision tree classifier
- KNeighbors classifier
- Logistic Regression
- GaussianNB
- Random forest classifier
- Ada boost classifier
- GradientBoostingClassifier
- Bagging classifier
- Extra trees classifier

- ## Run and Evaluate selected models

The algorithms we used are shown in fig 11,

```
#Importing all the model library

from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB

#Importing Boosting models
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier

#Importing error metrics
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,roc_curve,auc
from sklearn.model_selection import GridSearchCV,cross_val_score
```

Fig 11 Algorithms used

The results observed over different evaluation metrics are shown in fig 12,

	Model	Accuracy_score	Cross_val_score	Roc_auc_curve
0	KNeighborsClassifier	77.205681	87.873177	75.631662
1	LogisticRegression	77.131929	87.737849	75.797481
2	DecisionTreeClassifier	81.304885	84.364999	69.500930
3	XGBClassifier	82.237925	89.008266	78.611521
4	RandomForestClassifier	86.374863	89.042740	74.565563
5	AdaBoostClassifier	77.310305	87.978145	75.801378
6	GaussianNB	72.272914	80.704007	74.240251
7	GradientBoostingClassifier	81.308315	88.577077	77.726668
8	BaggingClassifier	83.772983	88.182418	74.849858
9	ExtraTreesClassifier	86.927141	88.890949	72.870659

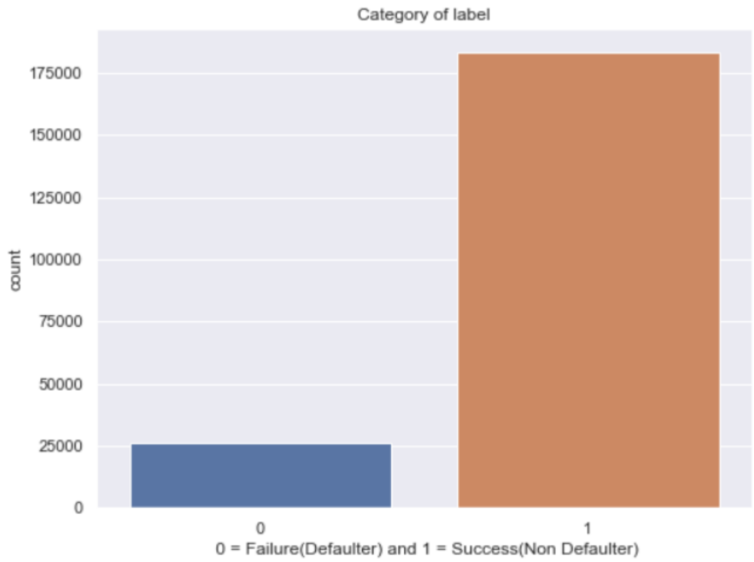
Fig 12 Results observed

- Key Metrics for success in solving problem under consideration

  Accuracy is not a appropriate measure of model performance here and we used the metric AREA UNDER ROC CURVE to evaluate models performance because high rocscore will mean high recall which means the model does well by not classifying legit transactions as fraudulent.

- Visualizations

Countplot of label:-




```
1    183431
0     26162
Name: label, dtype: int64
```

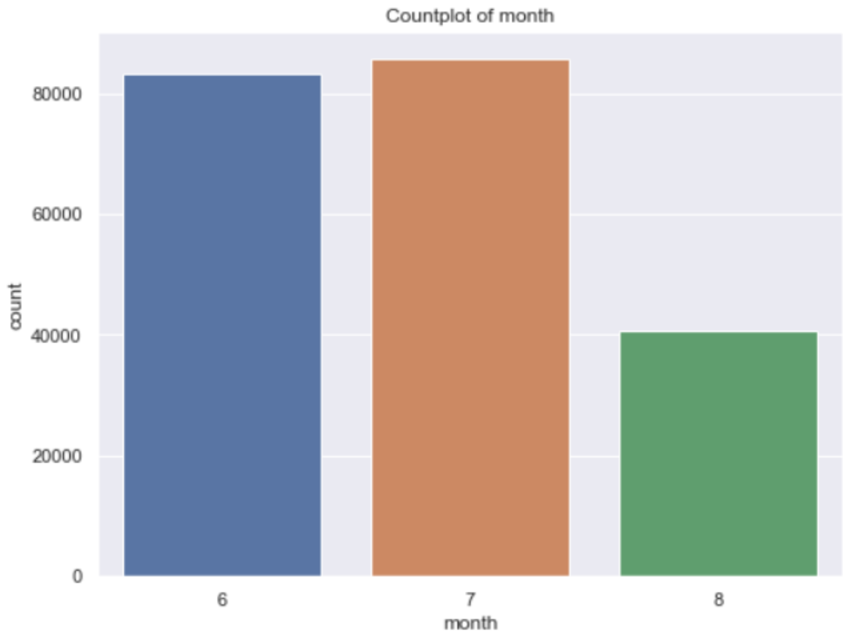
Fig 13 Countplot of label

Observation:

1. We observe 183431 number of Non defaulters whereas 26162 number of defaulters.

2. We observe that this is a very imbalanced data set.

Countplot of month: -




```
7    85765
6    83154
8    40674
Name: month, dtype: int64
```

Fig 14 Countplot of month

Observation:

Maximum (85765) number of users has taken credit on 7th month.

Checking column aon with label: -

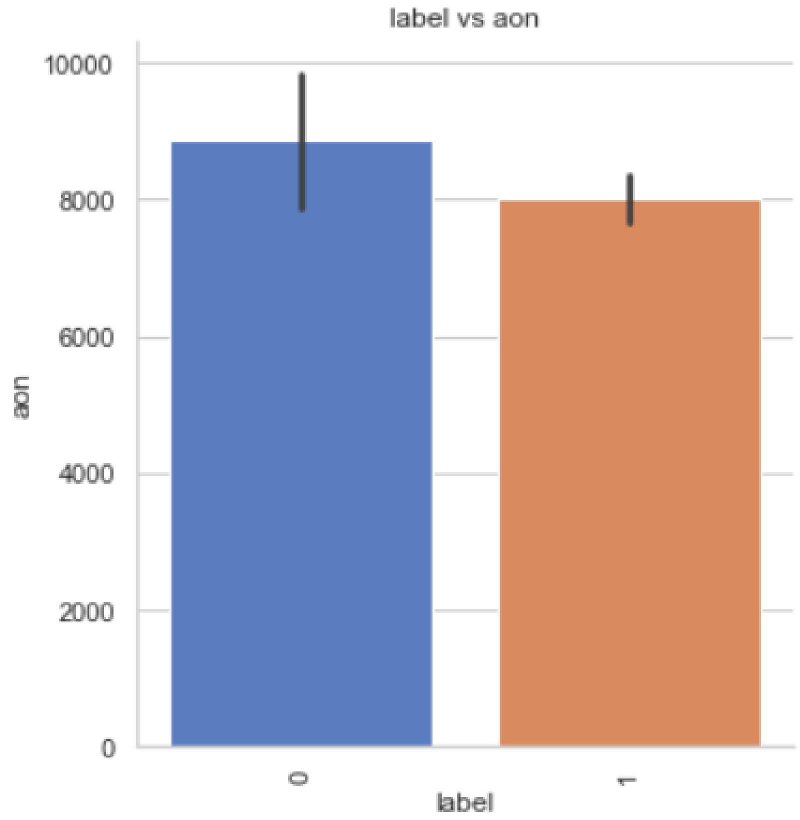


Fig 15 label vs aon

Observation:

If the aon is high the number of defaulters is more.

Checking column cnt_ma_rech30 with label: -

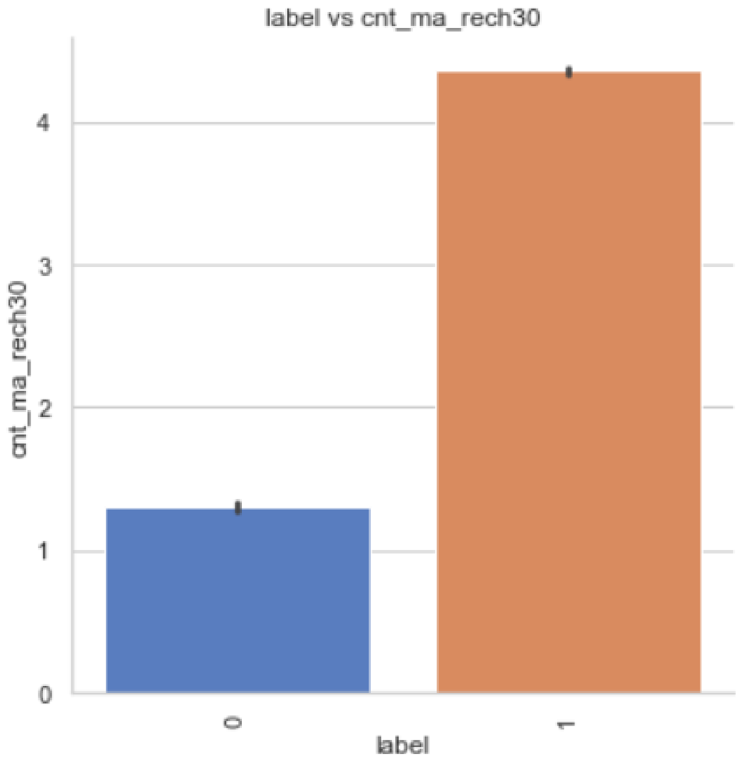


Fig 16 label vs cnt_ma_rech30

Observation:

If Number of times main account got recharged in last 30 days(cnt_ma_rech30) is more then there is less chance of default.

Checking the column sumamnt_ma_rech30 with label: -

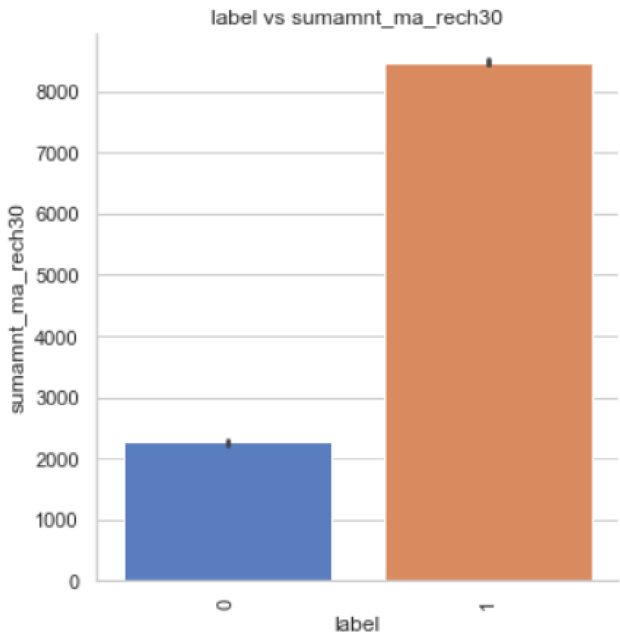


Fig 17 label vs sumamnt_ma_rech30

Observation:

If Total amount of recharge in main account over last 30 days(sumamnt_ma_rech30) is more the chances of default are less.

Checking cnt_ma_rech30 and cnt_ma_rech90 with label: -

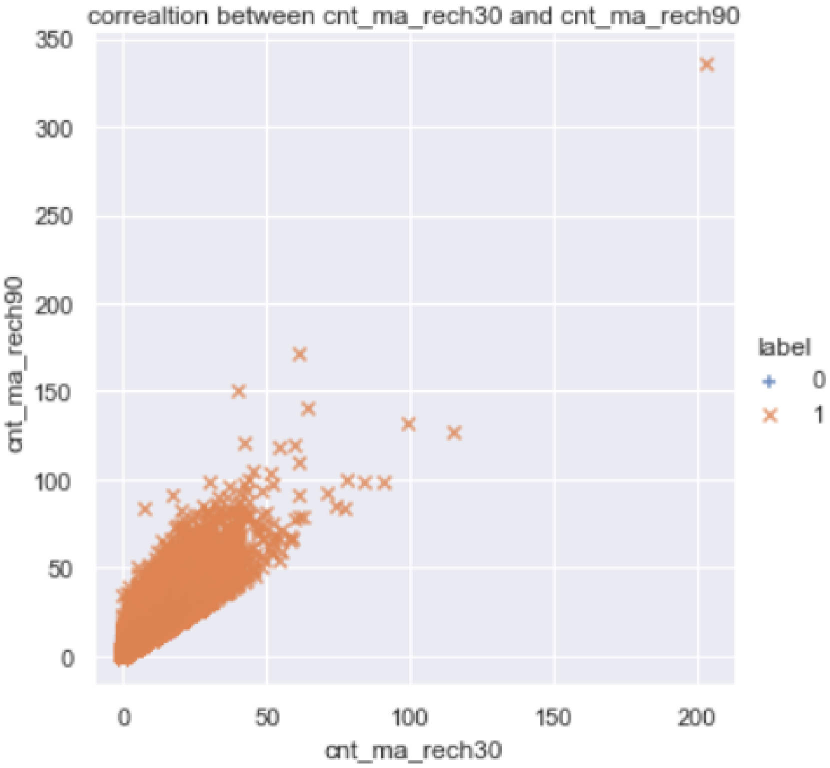


Fig 18 Scatter plot between cnt_ma_rech30 and cnt_ma_rech90 with respect to label

Observation:

As cnt_ma_rech30 and cnt_ma_rech90 are increasing the number of non-defaulters is also increasing.

Checking sumamnt_ma_rech90 and amnt_loans90 with label: -

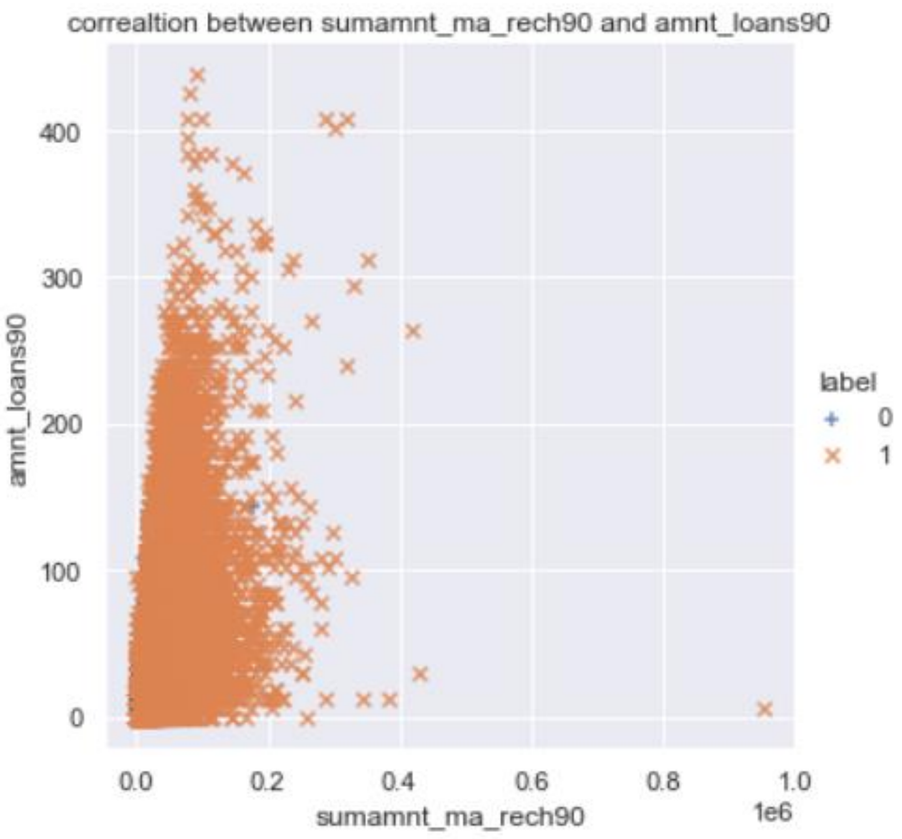


Fig 19 Scatter plot between sumamnt_ma_rech90 and amnt_loans90 with respect to label

Observation:

As sumamnt_rech90 and amnt_loans30 are increasing the number of non-defaulters is also increasing.

- Interpretation of the Results

From the visualization we interpreted that the data was very imbalanced, and the target variable was highly positively correlated with the columns cnt_ma_rech30 and cnt_ma_ma_rech90.

From the pre-processing we interpreted that data was improper scaled, there were hidden features present in the data which needed to be extracted.

From the modelling we interpreted that XGBClassifier works best with respect to our model with rocscore 0.90 as shown in fig 20

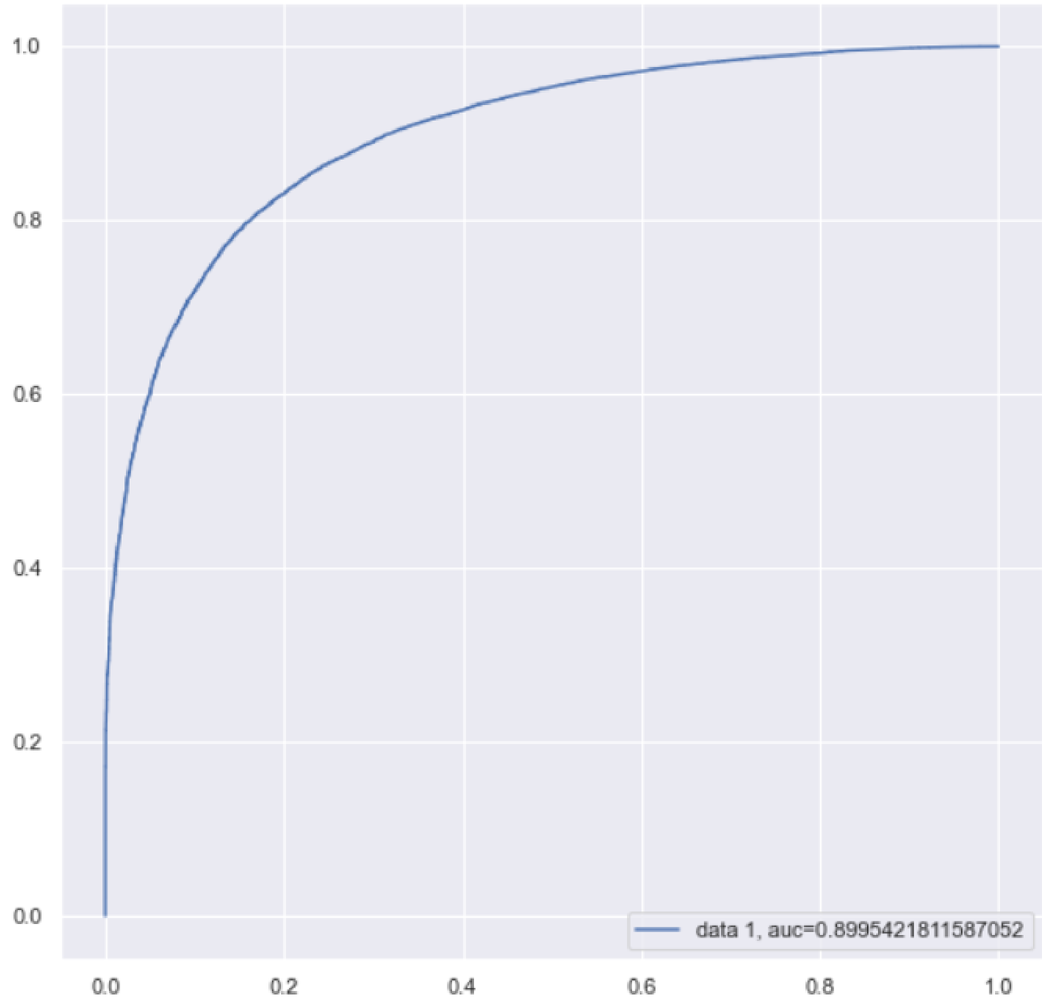


Fig 20 auc roc curve using XGBClassifier

CONCLUSION

- Key Findings and Conclusions of the Study

  In this project we have tried to show how to deal with unbalanced datasets like the MicroCreditDefaulter where the instances of fraudulent cases are few compared to the instances of non-fraudulent cases. We have argued why accuracy is not an appropriate measure of model performance here and used the metric AREA UNDER ROC CURVE to evaluate how method of SmoteTomek technique can lead to better model training.

  The best score of 0.90 was achieved using the best parameters of XGBClassifier through GridSearchCV though both random forest and gradient boosting models performed well too.

- Learning Outcomes of the Study in respect of Data Science

  This project has demonstrated the importance of sampling effectively, modelling and predicting data with an imbalanced dataset.

  Through different powerful tools of visualization, we were able to analyse and interpret different hidden insights about the data.

  Through data cleaning we were able to remove unnecessary columns and outliers from our dataset due to which our model would have suffered from overfitting or underfitting.

  The few challenges while working on this project were: -

  - Improper scaling
  - Too many features
  - Hidden features
  - Imbalanced data
  - Skewed data due to outliers

The data was improper scaled, so we scaled it to a single scale using sklearns's package StandardScaler.

There were too many (37) features present in the data so we applied Principal Component Analysis (PCA) and found out the Eigenvalues and on the basis of number of nodes we were able to reduce our features up to 7 columns.

There were hidden features present in pdate column, so we converted the column in datetime format in order to extract day and month column by doing feature extraction.

The data was imbalanced, so we handled the unbalanced data through SmoteTomek technique by creating more number of fraudulent cases on relevant data points.

The columns were skewed due to presence of outliers which we handled through winsorization technique.

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

While we couldn't reach out goal of 100% accuracy in fraud detection, we did end up creating a system that can with enough time and data get very close to that goal. As with any project there is room for improvement here. The very nature of this project allows for multiple algorithms to be integrated together as modules and their results can be combined to increase the accuracy of the final result. This model can further be improved with the addition of more algorithms into it. However, the output of these algorithms needs to be in the same format as the others. Once that condition is satisfied, the modules are easy to add as done in the code. This provides a great degree of modularity and versatility to the project.