Micro Credit Defaulter Project

Submitted by:

ROHIT KATTEWAR

ACKNOWLEDGEMENT

A Microfinance Institution (MFI) is an organization that offers financial services to low income populations. 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.

The case study data is provided to us from our client database who are working with 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.

INTRODUCTION

❖ BUSINESS PROBLEM FRAMING

The client we are working with has provided the dataset which relates to the Telecom Industry. The client collaborates with the MFI to provide the micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed to be defaulter if he deviates from the path of paying back the loaned amount within the time duration of 5 days. For the loan amount of 5 (in Indonesian Rupiah), payback amount should be 6 (in Indonesian Rupiah), while, for the loan amount of 10 (in Indonesian Rupiah), the payback amount should be 12 (in Indonesian Rupiah). The dataset is provided 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.

❖ CONCEPTUAL BACKGROUND OF THE DOMAIN PROBLEM

"Microfinance" is often seen as financial services for poor and low-income clients (Ayayi, 2012; Mensah, 2013; Tang, 2002). In practice, the term is often used more narrowly to refer to loans and other services from providers that identify themselves as "microfinance institutions" (MFIs) [Consultative Group to Assist the Poor (CGAP) 2010]. Microfinance can also be described as a setup of a number of different operators focusing on the financially under-served people with the aim of satisfying their need for poverty alleviation, social promotion, emancipation, and inclusion. Microfinance institutions reach and serve their target market in very innovative ways (Milana 2012). Microfinance institutions play a major role in economic development in many developing countries. However, many of these microfinance institutions are faced with the problem of default because of the non-formal nature of the business and individuals they lend money to. Default in microfinance is the failure of a client to repay a loan. The default could be in terms of the amount to be paid or the timing of the payment. MFIs can sustain and increase deployment of loans to stimulate the poverty reduction goal if repayment rates are high and consistent (Wongnaa 2013). MFIs are able to reduce interest rates and processing fees if repayment rates are high, thus increasing patronage of loans.

❖ REVIEW OF THE LITERATURE

In the given Micro-Credit Defaulter case study, we will be studying various features given in the dataset. The dataset contains both the dependent and independent variables which are going to contribute in the model building process. We will be using multiple machine learning models that gives the best scores and to predict the better customers.

❖ MOTIVATION FOR THE PROBLEM UNDERTAKEN

A Microfinance Institution (MFI) is an organization that offers financial services to low income populations. 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 forthcoming years to provide the customers with Micro Credit loan.

ANALYTICAL PROBLEM FRAMING

❖ Mathematical/Analytical Modeling of the Problem

We will begin with how the looks like in the Data frame, then we will be dealing with the Statistical summary of the data then we will look at the correlation between the various features with each other.

Data frame:

```
df = pd.read_csv('Micro_credit_defaulter.csv')
df
```

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30	med
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	...	6.0	
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	...	12.0	
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	...	6.0	
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	...	6.0	
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	...	6.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
209588	209589	1	22758I85348	404.0	151.872333	151.872333	1089.19	1089.19	1.0	0.0	...	6.0	
209589	209590	1	95583I84455	1075.0	36.936000	36.936000	1728.36	1728.36	4.0	0.0	...	6.0	
209590	209591	1	28556I85350	1013.0	11843.111667	11904.350000	5861.83	8893.20	3.0	0.0	...	12.0	
209591	209592	1	59712I82733	1732.0	12488.228333	12574.370000	411.83	984.58	2.0	38.0	...	12.0	
209592	209593	1	65061I85339	1581.0	4489.362000	4534.820000	483.92	631.20	13.0	0.0	...	12.0	

209593 rows × 37 columns

Description of the dataset:

Features:

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

Info of the dataset:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209593 entries, 0 to 209592
Data columns (total 37 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   Unnamed: 0            209593 non-null  int64
 1   label                 209593 non-null  int64
 2   msisdn                209593 non-null  object
 3   aon                   209593 non-null  float64
 4   daily_decr30          209593 non-null  float64
 5   daily_decr90          209593 non-null  float64
 6   rental30              209593 non-null  float64
 7   rental90              209593 non-null  float64
 8   last_rech_date_ma     209593 non-null  float64
 9   last_rech_date_da     209593 non-null  float64
 10  last_rech_amt_ma      209593 non-null  int64
 11  cnt_ma_rech30         209593 non-null  int64
 12  fr_ma_rech30          209593 non-null  float64
 13  sumamnt_ma_rech30     209593 non-null  float64
 14  medianamnt_ma_rech30  209593 non-null  float64
 15  medianmarechprebal30  209593 non-null  float64
 16  cnt_ma_rech90         209593 non-null  int64
 17  fr_ma_rech90          209593 non-null  int64
 18  sumamnt_ma_rech90     209593 non-null  int64
 19  medianamnt_ma_rech90  209593 non-null  float64
 20  medianmarechprebal90  209593 non-null  float64
 21  cnt_da_rech30         209593 non-null  float64
 22  fr_da_rech30          209593 non-null  float64
 23  cnt_da_rech90         209593 non-null  int64
 24  fr_da_rech90          209593 non-null  int64
 25  cnt_loans30           209593 non-null  int64
 26  amnt_loans30          209593 non-null  int64
 27  maxamnt_loans30       209593 non-null  float64
 28  medianamnt_loans30    209593 non-null  float64
 29  cnt_loans90           209593 non-null  float64
 30  amnt_loans90          209593 non-null  int64
 31  maxamnt_loans90       209593 non-null  int64
 32  medianamnt_loans90    209593 non-null  float64
 33  payback30             209593 non-null  float64
 34  payback90             209593 non-null  float64
 35  pcircle               209593 non-null  object
 36  pdate                 209593 non-null  object
dtypes: float64(21), int64(13), object(3)
memory usage: 59.2+ MB
```

The dataset has total 37 columns out of 3 columns have their data-type as 'object' and rest of the remaining columns have numerical data type.

Null Values:

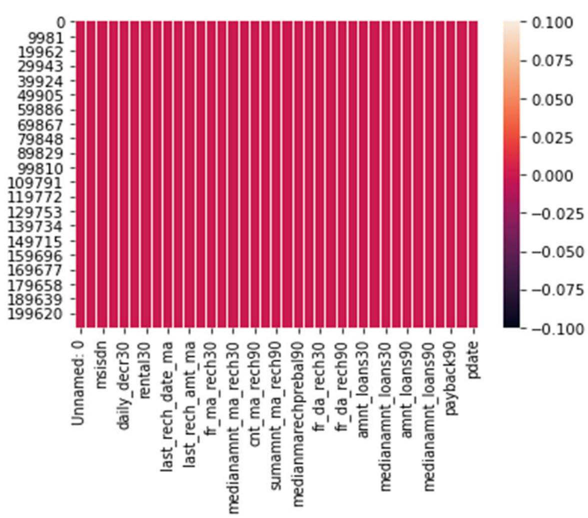
We have no null values in the dataset.

```
df.isnull().sum()
```

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

```
sns.heatmap(df.isnull())
```

```
<AxesSubplot:>
```



We have no null values in the dataset.

The heatmap above represents the Null values present in the dataset.

Statistical Summary:

It gives the basic statistics about the data like the percentile, mean, maximum, minimum etc.

Statistical summary:

```
df.describe()
```

	Unnamed: 0	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_r
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	2
mean	104797.000000	0.875177	8112.343445	5381.402289	6082.515068	2692.581910	3483.406534	3755.847800	3712.202921	
std	60504.431823	0.330519	75696.082531	9220.623400	10918.812767	4308.586781	5770.461279	53905.892230	53374.833430	
min	1.000000	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000	-29.000000	
25%	52399.000000	1.000000	246.000000	42.440000	42.692000	280.420000	300.260000	1.000000	0.000000	
50%	104797.000000	1.000000	527.000000	1469.175667	1500.000000	1083.570000	1334.000000	3.000000	0.000000	
75%	157195.000000	1.000000	982.000000	7244.000000	7802.790000	3356.940000	4201.790000	7.000000	0.000000	
max	209593.000000	1.000000	999860.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733	999171.809410	5

8 rows × 37 columns

Key Points:

- There are 209593 distinct micro-credit customers.
- The average value for number of loans taken by user in last 30 days is 2.75 and std is 2.55, max value is 50.
- The average value for Number of days till last recharge of main account is 3755.84. The standard deviation is unusually large, max value being 998650.37.
- The average value for number of times data account got recharge in last 30 days is 262.57. The standard deviation is high , amx value being 99914.44

Correlation:

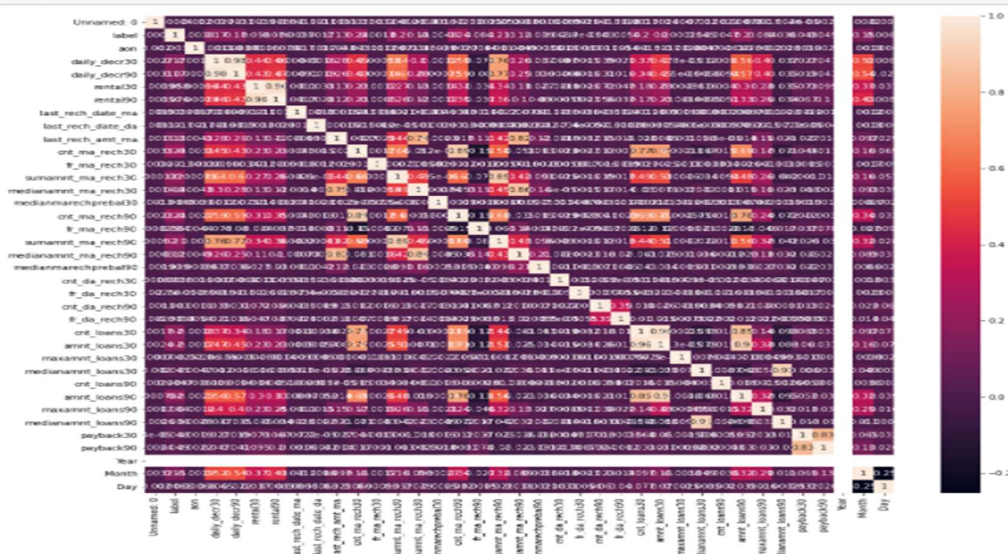
Correlation:

```
df.corr()
```

	Unnamed: 0	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_
Unnamed: 0	1.000000	0.000403	-0.002048	0.002739	0.003077	-0.003906	-0.003459	-0.001853	-0.001133	-0.0010
label	0.000403	1.000000	-0.003785	0.168298	0.166150	0.058085	0.075521	0.003728	0.001711	0.1318
aon	-0.002048	-0.003785	1.000000	0.001104	0.000374	-0.000960	-0.000790	0.001692	-0.001693	0.0042
daily_decr30	0.002739	0.168298	0.001104	1.000000	0.977704	0.442066	0.458977	0.000487	-0.001636	0.2758
daily_decr90	0.003077	0.166150	0.000374	0.977704	1.000000	0.434685	0.471730	0.000908	-0.001886	0.2641
rental30	-0.003906	0.058085	-0.000960	0.442066	0.434685	1.000000	0.955237	-0.001095	0.003261	0.1272
rental90	-0.003459	0.075521	-0.000790	0.458977	0.471730	0.955237	1.000000	-0.001688	0.002794	0.1214
last_rech_date_ma	-0.001853	0.003728	0.001692	0.000487	0.000908	-0.001095	-0.001688	1.000000	0.001790	-0.0001
last_rech_date_da	-0.001133	0.001711	-0.001693	-0.001636	-0.001886	0.003261	0.002794	0.001790	1.000000	-0.0001
last_rech_amt_ma	-0.001064	0.131804	0.004256	0.275837	0.264131	0.127271	0.121416	-0.000147	-0.000149	1.0000
cnt_ma_rech30	0.003320	0.237331	-0.003148	0.451385	0.426707	0.233343	0.230260	0.004311	0.001549	-0.0026
fr_ma_rech30	0.003181	0.001330	-0.001163	-0.000577	-0.000343	-0.001219	-0.000503	-0.001629	0.001158	0.0028
sumamnt_ma_rech30	0.000123	0.202828	0.000707	0.636536	0.603886	0.272649	0.259709	0.002105	0.000046	0.4406
medianamnt_ma_rech30	-0.001371	0.141490	0.004306	0.295356	0.282960	0.129853	0.120242	-0.001358	0.001037	0.7946
medianmarechprebal30	0.001258	-0.004829	0.003930	-0.001153	-0.000746	-0.001415	-0.001237	0.004071	0.002849	-0.0022
cnt_ma_rech90	0.002329	0.236392	-0.002725	0.587338	0.593069	0.312118	0.345293	0.004263	0.001272	0.0167
fr_ma_rech90	-0.000249	0.084385	0.004401	-0.078299	-0.079530	-0.033530	-0.036524	0.001414	0.000798	0.1062
sumamnt_ma_rech90	0.000523	0.205793	0.001011	0.762981	0.768817	0.342306	0.360601	0.002243	-0.000414	0.4187
medianamnt_ma_rech90	-0.000298	0.120855	0.004909	0.257847	0.250518	0.110356	0.103151	-0.000726	0.000219	0.8187
medianmarechprebal90	-0.001947	0.039300	-0.000859	0.037495	0.036382	0.027170	0.029547	-0.001086	0.004158	0.1246
cnt_da_rech30	0.000688	0.003827	0.001564	0.000700	0.000661	-0.001105	-0.000548	-0.003467	-0.003628	-0.0018
fr_da_rech30	-0.002504	-0.000027	0.000892	-0.001499	-0.001570	-0.002558	-0.002345	-0.003626	-0.000074	-0.0032
cnt_da_rech90	-0.001324	0.002999	0.001121	0.038814	0.031155	0.072255	0.056282	-0.003538	-0.001859	0.0147
fr_da_rech90	-0.002827	-0.005418	0.005395	0.020673	0.016437	0.046761	0.036886	-0.002395	-0.000203	0.0160
cnt_loans30	0.001725	0.196283	-0.001826	0.366116	0.340387	0.180203	0.171595	0.001193	0.000380	-0.0276
amnt_loans30	0.002387	0.197272	-0.001726	0.471492	0.447869	0.233453	0.231906	0.000903	0.000536	0.0085
maxamnt_loans30	0.000698	0.000248	-0.002764	-0.000028	0.000025	-0.000864	-0.001411	0.000928	0.000503	0.0010
medianamnt_loans30	-0.002005	0.044589	0.004664	-0.011610	-0.005591	-0.016482	-0.009467	0.001835	0.000061	0.0283
cnt_loans90	0.002241	0.004733	-0.000611	0.008962	0.009446	0.004012	0.005141	-0.000225	-0.000972	0.0000
amnt_loans90	0.000781	0.199788	-0.002319	0.563496	0.567204	0.298943	0.327436	0.000870	0.000519	0.0140
maxamnt_loans90	0.001742	0.084144	-0.001191	0.400199	0.397251	0.234211	0.251029	-0.001123	0.001524	0.1484
medianamnt_loans90	-0.002615	0.035747	0.002771	-0.037305	-0.034686	-0.035489	-0.034122	0.002771	-0.002239	0.0210
payback30	-0.000040	0.048336	0.001940	0.026915	0.019400	0.072974	0.067110	-0.002233	0.000077	-0.0273
payback90	0.002411	0.049183	0.002203	0.047175	0.040800	0.095147	0.099501	-0.001583	0.000417	-0.0142
Year	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
Month	0.003205	0.154949	-0.001863	0.518664	0.539410	0.365699	0.429407	-0.001207	-0.001800	0.0969
Day	-0.002045	0.006825	0.000662	0.006477	-0.021508	0.036537	0.008941	0.000560	0.000631	0.0288

37 rows × 37 columns

```
corr_hmap=df.corr()
plt.figure(figsize=(16,14))
sns.heatmap(corr_hmap,annot=True)
plt.show()
```



❖ 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 if they had paid back loan or not. The other file contains details of the dataset.

❖ Data Pre-processing Done

The column "pdate" has it's data-type object and we need to convert it into date-time format.

Converting the data type format of the "pdate" column

```
df["Year"] = pd.to_datetime(df.pdate, format="%Y-%m-%d").dt.year
df["Month"] = pd.to_datetime(df.pdate, format="%Y-%m-%d").dt.month
df["Day"] = pd.to_datetime(df.pdate, format="%Y-%m-%d").dt.day
```

We converted the data-type of the column and created three separate columns as "Year", "Month" and "Day".

Dropping the columns

Dropping the columns

```
df.drop(['pdate'],axis=1,inplace=True)
df.drop(['Unnamed: 0'],axis=1,inplace=True)
df.drop(['Year'],axis=1,inplace=True)
df.drop(['msisdn'],axis=1,inplace=True)
df.drop(['fr_da_rech30'],axis=1,inplace=True)
df.drop(['maxamnt_loans30'],axis=1,inplace=True)
df.drop(['pcircle'],axis=1,inplace=True)
```

MODEL DEVELOPMENT AND EVALUATION

❖ Identification of possible problem-solving approaches (methods)

Spliting the data into Feature and Target:

```
x=new_df.drop('label',axis=1)
y=new_df['label']
```

```
x.shape
```

```
(163026, 32)
```

```
y.shape
```

```
(163026,)
```

```
x
```

	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	...	amr
0	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	2	21.0	...	
1	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	5787	1	0.0	...	
2	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539	1	0.0	...	
3	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	947	0	0.0	...	
4	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309	7	2.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
209588	404.0	151.872333	151.872333	1089.19	1089.19	1.0	0.0	4048	3	2.0	...	
209589	1075.0	36.936000	36.936000	1728.36	1728.36	4.0	0.0	773	4	1.0	...	
209590	1013.0	11843.111667	11904.350000	5861.83	8893.20	3.0	0.0	1539	5	8.0	...	
209591	1732.0	12488.228333	12574.370000	411.83	984.58	2.0	38.0	773	5	4.0	...	
209592	1581.0	4489.362000	4534.820000	483.92	631.20	13.0	0.0	7526	2	1.0	...	

163026 rows × 32 columns

```
y
```

```
0         0
1         1
2         1
3         1
4         1
         ..
209588    1
209589    1
209590    1
209591    1
209592    1
Name: label, Length: 163026, dtype: int64
```

We will be split the data into target and feature as x and y respectively.

Scalling and Oversampling the x and y

Scalling:

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x=scaler.fit_transform(x)
```

```
x.shape
```

```
(163026, 32)
```

```
y.value_counts()
```

```
1    140409
0     22617
Name: label, dtype: int64
```

Oversampling:

```
import imblearn
from imblearn.over_sampling import SMOTE
SM = SMOTE()
x,y = SM.fit_resample(x,y)
```

Getting the best accuracy score and a specific random state

 # importing all the required libraries

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score, confusion_matrix,f1_score, classification_report, roc_auc_score, roc_curve,auc
```

```
lc=LogisticRegression()
from sklearn.model_selection import train_test_split
maxAccu=0
maxRS=0
for i in range(1,200):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=i)
    lc.fit(x_train,y_train)
    pred=lc.predict(x_test)
    acc=accuracy_score(y_test,pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Best Accuracy is",maxAccu," on Random_state",maxRS)
```

```
Best Accuracy is 0.778131899437362  on Random_state 24
```

Train Test Split the data:

Train Test Split the data: We got the Best Accuracy is 0.7782387294352254 on Random_state 110. Hence, using these values we are going to train our data using following models :-

Logistic Regression

Decision Tree Classifier

Random Forest Classifier

GaussianNB

KNeighbors Classifier

Gradient Boosting Classifier

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.25,random_state=110)
```

```
x_train.shape
```

```
(210613, 32)
```

```
x_test.shape
```

```
(70205, 32)
```

```
y_train.shape
```

```
(210613,)
```

```
y_test.shape
```

```
(70205,)
```

Model Building:

```
#Gaussian NB
gb=GaussianNB()
gb.fit(x_train,y_train)
pred=gb.predict(x_test)
acc=accuracy_score(y_test,pred)
cnm=confusion_matrix(y_test,pred)
cr=classification_report(y_test,pred)

# Getting the accuarcy score
print(f"Accuracy Score: {acc}")
print(" --:--  --:--  --:--  --:--  --:--  --:--  --:--  --:-- ")

# Getting the confusion matrix
print(f"Confusion Matrix : \n {cnm}\n")
print(" --:--  --:--  --:--  --:--  --:--  --:--  --:--  --:-- ")

# Getting the classification report
print(f"Classification Report : \n {cr}")

# Getting the CV score
cvgb=cross_val_score(gb,x,y,cv=5).mean()
print("Cross Validation Score for GaussianNB is : ",cvgb)
print(" --:--  --:--  --:--  --:--  --:--  --:--  --:--  --:-- ")

# Getting the difference between the accuracy score and CV score
result = acc - cvgb
print("\nAccuracy Score - Cross Validation Score :", result)
```

```
Accuracy Score: 0.7294067374118652
 --:--  --:--  --:--  --:--  --:--  --:--  --:--  --:--
Confusion Matrix :
 [[30241  4945]
 [14052 20967]]

 --:--  --:--  --:--  --:--  --:--  --:--  --:--  --:--
Classification Report :
               precision    recall  f1-score   support

           0       0.68      0.86      0.76     35186
           1       0.81      0.60      0.69     35019

    accuracy                           0.73     70205
   macro avg       0.75      0.73      0.72     70205
weighted avg       0.75      0.73      0.72     70205

Cross Validation Score for GaussianNB is :  0.7292730631822327
 --:--  --:--  --:--  --:--  --:--  --:--  --:--  --:--

Accuracy Score - Cross Validation Score : 0.00013367422963250775
```

Gaussian Naïve Bayes:

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of P(xi|y).
Above, we are getting the accuracy score, confusion matrix, Classification report and Cross Validation score for the Gaussian Naïve Bayes model.
We select that model as a best model which has the least difference between its Accuracy score and Cross Validation Score.

Hyper Parameter Tuning of the model:

Hyper Parameter Tuning:

We are selecting GaussianNB as our best model as it has least difference between it's Accuracy score and CV score

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

```
parameter={'var_smoothing': np.logspace(0,-9, num=100)}
```

```
GCV=GridSearchCV(gb,parameter,cv=5)
GCV.fit(x_train,y_train)
```

```
GridSearchCV(cv=5, estimator=GaussianNB(),
             param_grid={'var_smoothing': array([1.00000000e+00, 8.11130831e-01, 6.57933225e-01, 5.33669923e-01,
       4.32876128e-01, 3.51119173e-01, 2.84803587e-01, 2.31012970e-01,
       1.87381742e-01, 1.51991108e-01, 1.23284674e-01, 1.00000000e-01,
       8.11130831e-02, 6.57933225e-02, 5.33669923e-02, 4.32876128e-02,
       3.51119173e-02, 2.84803587e-02, 2.31...
       1.23284674e-07, 1.00000000e-07, 8.11130831e-08, 6.57933225e-08,
       5.33669923e-08, 4.32876128e-08, 3.51119173e-08, 2.84803587e-08,
       2.31012970e-08, 1.87381742e-08, 1.51991108e-08, 1.23284674e-08,
       1.00000000e-08, 8.11130831e-09, 6.57933225e-09, 5.33669923e-09,
       4.32876128e-09, 3.51119173e-09, 2.84803587e-09, 2.31012970e-09,
       1.87381742e-09, 1.51991108e-09, 1.23284674e-09, 1.00000000e-09])})
```

```
GCV.best_params_
```

```
{'var_smoothing': 5.336699231206313e-06}
```

```
micro_final=GaussianNB(var_smoothing=5.336699231206313e-06)
micro_final.fit(x_train,y_train)
pred=micro_final.predict(x_test)
acc=accuracy_score(y_test,pred)
print(acc*100)
```

```
72.94067374118653
```

We are getting the model accuracy and cross validation score both as 72.94% which shows our model is performing well.

Here we are getting our Model Accuracy Score and Cross Validation Score both as 72.94%.

AUC-ROC Curve:

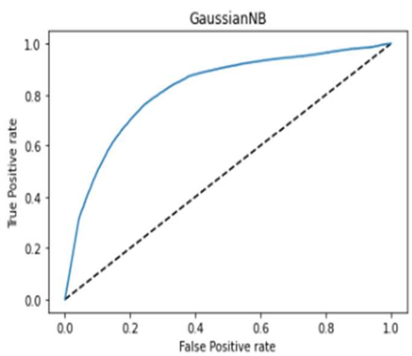
AUC - ROC Curve:

AUC Curve - A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

ROC Curves - It summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds.

```
y_pred_proba= gb.predict_proba(x_test)[:,1]
fpr,tpr,thresholds=roc_curve(y_test, y_pred_proba)
plt.plot([0,1],[0,1],'k--')
plt.plot(fpr,tpr,label='dtr')
plt.xlabel('False Positive rate')
plt.ylabel('True Positive rate')
plt.title('GaussianNB')
plt.show()

# Getting the AUC score
auc_score=roc_auc_score(y_test,gb.predict(x_test))
print('The AUC Score is ',auc_score)
```




```
The AUC Score is  0.7290966330858025
```

We are getting the AUC Score is 0.7290

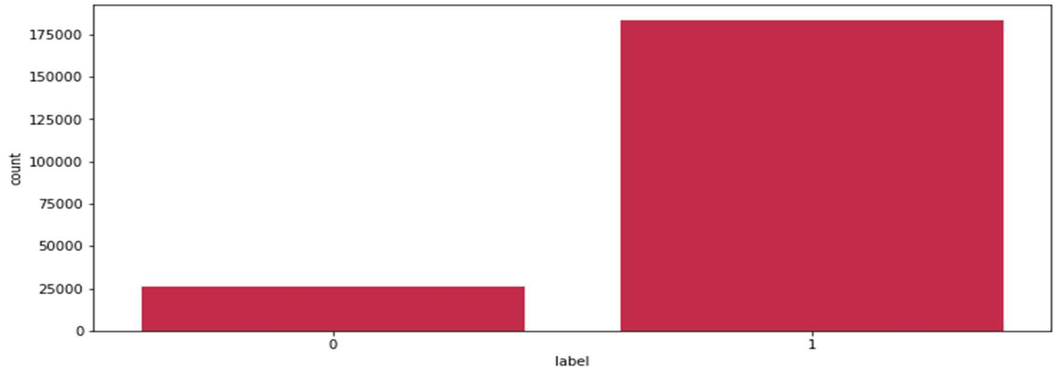
AUC Curve: - The **Area Under the Curve (AUC)** is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higherthe AUC, the better the performance of the model at distinguishing between the positive and negative classes.

ROC: - **Receiver Operating Characteristic (ROC)** summarizes the model's performance by evaluating the tradeoffs between true positive rate (sensitivity) and false positiverate (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.

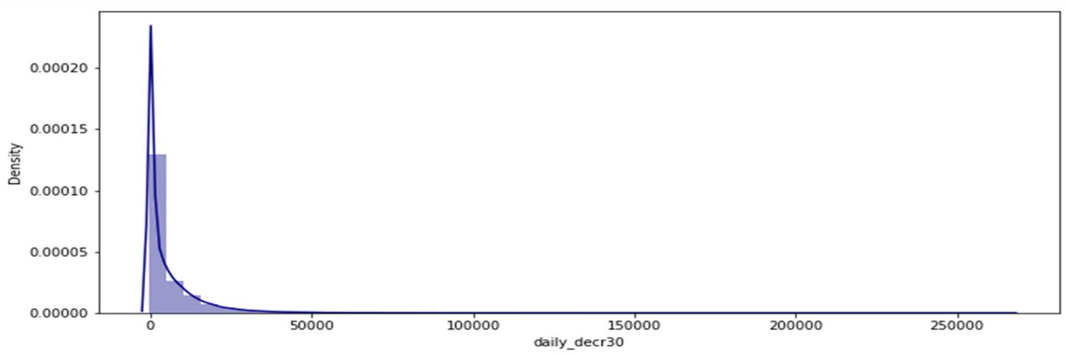
❖ Visualization

Univariate analysis:

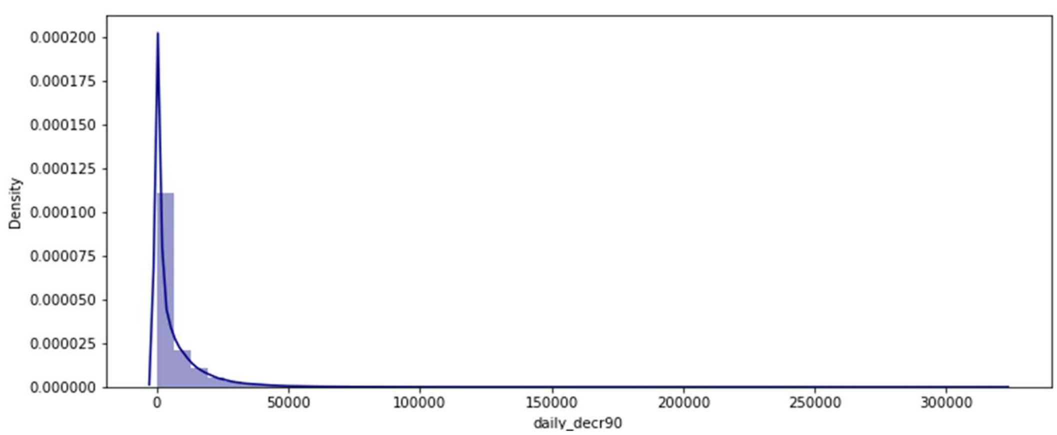
```
plt.figure(figsize=(12,5))
sns.countplot(df['label'], color='crimson')
plt.show()
```



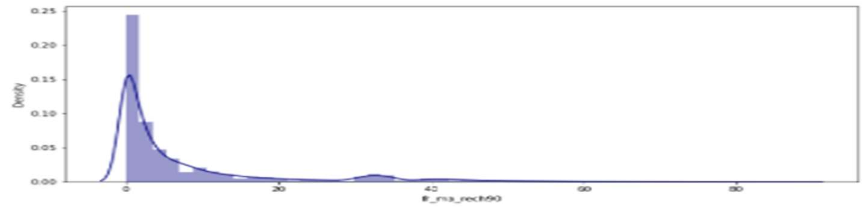

```
plt.figure(figsize=(12,5))
sns.distplot(df['daily_decr30'], color='navy')
plt.show()
```



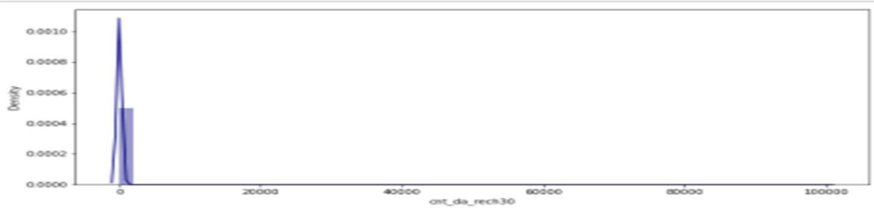

```
plt.figure(figsize=(12,5))
sns.distplot(df['daily_decr90'],color='navy')
plt.show()
```



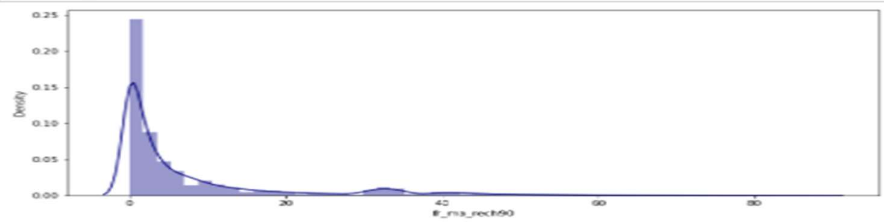
```
plt.figure(figsize=(12,5))
sns.distplot(df['fr_ma_rech90'], color='navy')
plt.show()
```



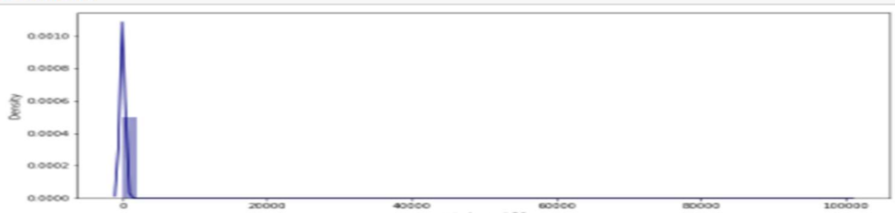

```
plt.figure(figsize=(12,5))
sns.distplot(df['cnt_da_rech30'], color='navy')
plt.show()
```




```
plt.figure(figsize=(12,5))
sns.distplot(df['fr_ma_rech90'], color='navy')
plt.show()
```

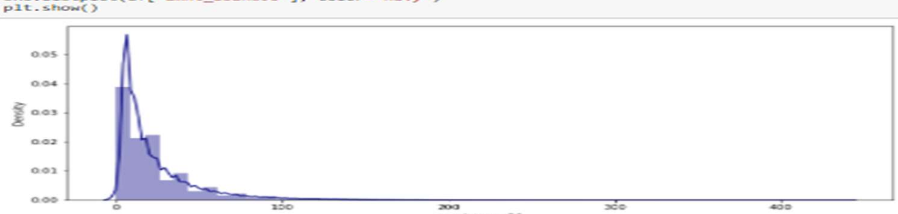



```
plt.figure(figsize=(12,5))
sns.distplot(df['cnt_da_rech30'], color='navy')
plt.show()
```



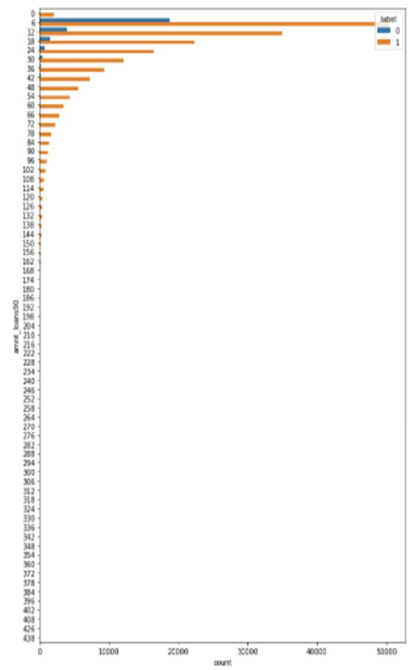

```
plt.figure(figsize=(12,5))
sns.distplot(df['amnt_loans30'], color='navy')
plt.show()
```


```
plt.figure(figsize=(12,5))
sns.distplot(df['amnt_loans90'], color='navy')
plt.show()
```



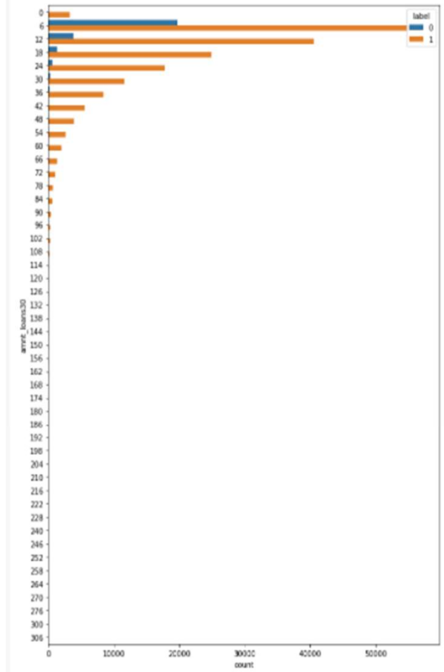
```
plt.subplots(figsize=(10,15))
sns.countplot(y="amnt_loans90",hue="label",data=df)
```

<AxesSubplot:xlabel='count', ylabel='amnt_loans90'>



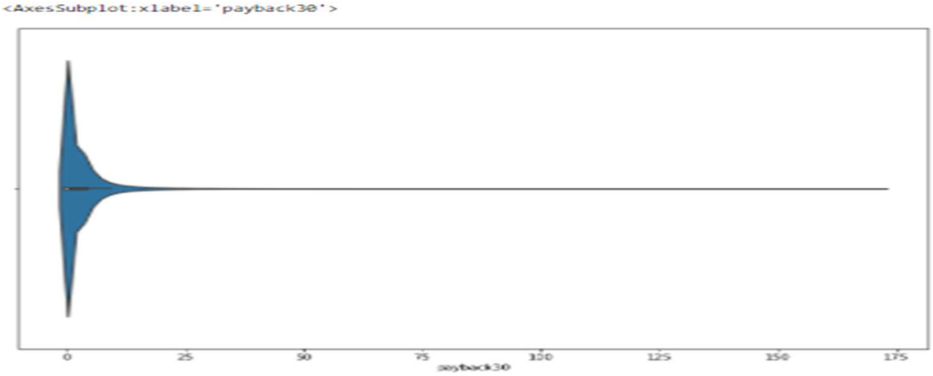

```
plt.subplots(figsize=(10,15))
sns.countplot(y="amnt_loans30",hue="label",data=df)
```

<AxesSubplot:xlabel='count', ylabel='amnt_loans30'>



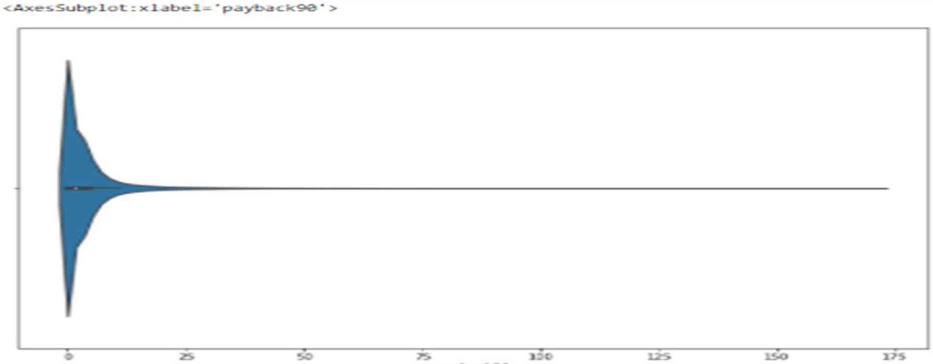

```
plt.subplots(figsize=(12,8))
sns.violinplot(x="payback30",hue="label",data=df)
```

<AxesSubplot:xlabel='payback30'>




```
plt.subplots(figsize=(12,8))
sns.violinplot(x="payback90",hue="label",data=df)
```

<AxesSubplot:xlabel='payback90'>



CONCLUSION

❖ Key Findings and Conclusion

Conclusion

```
Conclusion = pd.DataFrame([micro_model.predict(x_test)[:],gb.predict(x_test)[:]],index=["Predicted","Original"])
Conclusion
```

	0	1	2	3	4	5	6	7	8	9	...	70195	70196	70197	70198	70199	70200	70201	70202	70203	70204
Predicted	1	0	1	1	0	1	0	0	0	1	...	0	0	1	0	0	1	1	1	0	0
Original	1	0	1	1	0	1	0	0	0	1	...	0	0	1	0	0	1	1	1	0	0

2 rows × 70205 columns

----- --:-- -----

It is globally accepted as the objective of the Micro Finance Institutions is to help the poverty prone population and come forward as a poverty-reduction tool. The aim behind this case study is to determine an appropriate quantitative model for using the financial information pertaining to the loan and customer behaviour on the mobile network to predict the outcome of the loan.

This case study is an example of Classification. The Classification models are appropriate for dealing with the two distinct outcomes for customer behaviour of repayment and defaulter.

----- --:-- ----- ----- --:-- ----- ❄ ----- --:-- ---------- --:-- -----