Micro Credit Defaulter Prediction Project

Submitted by:
Vijaya Mukati

ACKNOWLEDGMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to Flip Robo Technologies Bangalore for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I want to thank my SME Miss. Khushboo Garg for providing the projectand helping us to solve the problem and addressing out our Query in right time.

I would like to express my gratitude towards my parents & members of Flip Robo for their kind co-operation and encouragement which help me in completion of this project.

I would like to express my special gratitude and thanks to industry persons for giving me such attention and time.

INTRODUCTION

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.

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

We are working with one such client that is in Telecom Industry. They are a fixed wireless telecommunications network service 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.

And we have collected the following data of mobile network user from our client database in using which we are training our model for prediction.

1. **label** - 1 not-Defaulter, 0 Defaulter (TARGET VARIABLE)
2. **msisdn** - mobile number of users
3. **aon** - age on cellular network in days
4. **daily_decr30** - averaged over last 30 days
5. **daily_decr90** - averaged over last 90 days
6. **rental30** - Average main account balance over last 30 days
7. **rental90** - Average main account balance over last 90 days
8. **last_rech_date_ma** - Number of days till last recharge of main account
9. **last_rech_date_da** - Number of days till last recharge of data account
10. **last_rech_amt_ma** - Amount of last recharge of main account
11. **cnt_ma_rech30** - No. of times recharge in last 30 days
12. **fr_ma_rech30** - Frequency of recharge in last 30 days
13. **sumamnt_ma_rech30** - Sum of recharge in 30 days
14. **medianamnt_ma_rech30** - median of recharge in 30 days
15. **medianmarechprebal30** - median of balance in last 30 days
16. **cnt_ma_rech90** - No. of times recharge in last 90 days

17. **fr_ma_rech90** - Frequency of recharge in last 90 days
18. **sumamnt_ma_rech90** - Sum of recharge in 90 days
19. **medianamnt_ma_rech90** - median of recharge in 90 days
20. **medianmarechprebal90** - median of balance in last 90 days
21. **cnt_da_rech30** - No. of times data recharge in last 30 days
22. **fr_da_rech30** - Frequency of data recharge in last 30 days
23. **cnt_da_rech90** - No. of times data recharge in last 90 days
24. **fr_da_rech90** - Frequency of data recharge in last 90 days
25. **cnt_loans30** - Number of loans taken by user in last 30 days
26. **amnt_loans30** - Total amount of loans taken by user in last 30 days
27. **maxamnt_loans30** - max amount taken in last 30 days
28. **medianamnt_loans30** - Median of amounts of loan taken by the user in last 30 days
29. **cnt_loans90 -** Number of loans taken by user in last 90 days
30. **amnt_loans90** - Total amount of loans taken by user in last 90 days
31. **maxamnt_loans90** - maximum amount of loan taken by the user in last 90 days
32. **medianamnt_loans90** - Median of amounts of loan taken by the user in last 90 days
33. **payback30** - Average payback time in days over last 30 days
34. **payback90** - Average payback time in days over last 90 days
35. **pcircle** - telecom circle
36. **pdate** – Date

Many (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.

The sample data is provided to us from the client database. In understanding the above data, I have tried to plot said factors and with the use of machine learning, tried and successfully arrived at a model that can detect whether the customer will pay back the loaned amount within 5 days of insurance of loan. In this case, Label '1' indicates that the loan has been paid i.e., Non- defaulter, while, Label '0' indicates that the loan has not been paid i.e., defaulter bases these factors. We can use this model in order to improve the selection of customers for the credit and do some predictions that could help in further investment and improvement in selection of customers.

• Business Problem Framing.

The Data is provided to us from the Client Database which has 209593 records of mobile phone user in Indonesia. These records are about the mobile phone user does transaction with the mobile network service providers such as recharges done in 30 days and 90 days, loan taken within last 30 and 90 days, the trend of repayment for every loan taken and also the tenure of the users in the same network. Understanding the above factors as said I have tried to plot and predict using machine learning, tried and successfully arrived at a model.

So, using the model in the future the Client can predict and analyse the customer behaviour and also come to a conclusion whether to sanction a loan to a specified customer or not and also in order to improve the onboarding of the customers for the credit.

• Conceptual Background of the Domain Problem

Comprehensive support from financial institutions is required in effort to drive community empowerment, particularly middle to low-income society and micro, small and medium enterprises (UMKM). This group of enterprises has limited access to formal financial institutions so far. Therefore, in order to deal with such problems, many non-bank financial institutions have grown and developed in society, running services in business development and community empowerment, and are established by government or society. Those institutions are well-known as microfinance institution (MFIs). However, many of the MFIs still do not have legal entity or business license yet. In order to provide a strong legal groundwork for MFIs` operation, Law Number 1 of 2013 on MFIs has been issued on January 8, 2013.

Legal Groundwork

1. Law Number 1 of 2013 on microfinance institutions (MFI Law).
2. Government Regulation Number 89 of 2014 on loan interest rate or yield of financing and MFI`s business coverage.
3. OJK Regulations (POJK):
   (a) OJK Regulation Number 12/POJK.05/2014 on business licensing and institutional matters of MFIs.
   (b) OJK Regulation Number 13/POJK.05/2014 on business management of MFIs.
   (c) OJK Regulation Number 14/POJK.05/2014 on fostering and supervision of MFIs.

MFI`s Business Activities

1) MFI`s business activities cover business development and community empowerment through loan or financing for micro-scaled business of MFI members and society, deposit management, or giving consultancy services in business development.
2) Business activities as mentioned above can be carried out using conventional practices or based on Sharia principles.

Objectives of MFI

1) To improve access to micro-scaled funding for society;
2) To help improving economic empowerment and productivity in society; and
3) To help increasing society`s income and prosperity, mainly of disadvantaged and/or low-income society.

MFI Ownership

An MFI can be owned by:

1) Indonesian citizen;
2) Village/rural enterprise;
3) Regional government (regency/city); and/or
4) Cooperative

MFI is not allowed to be owned, whether directly or indirectly, by foreign citizen and/or enterprise owned in whole or in part by foreign citizen or foreign enterprise.

This is the journal of Otoritas Jasa Keuangan Indonesia (OJK, 2017)

• Review of Literature

This paper provides a systematic assessment of customer behaviour and the trend of loan taken Vs loan paid back to the Microfinance Institution (MFI) in Indonesia bases on the above given factors. From our analysis and research done one can understand that depending on what factors customers are ending up on being a defaulter, using this study a Microfinance Institution (MFI) can predict that a customer can be a defaulter or not basis on which Microfinance Institution (MFI) can confer short-term loans.

Further our study helps the Micro finance institution to understand, analysis and predict the futuristic behaviour of a customer that could help them in further speculation and improvement in selection of customers.

• Motivation for the Problem Undertaken

Build a model using the best machine learning algorithm which can be used to predict in terms of a probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance of loan. In this case, Label '1' indicates that the loan has been paid i.e., Non- defaulter, while, Label '0' indicates that the loan has not been paid i.e., defaulter.

Analytical Problem Framing

- ## Mathematical/ Analytical Modelling of the Problem

Our data consist of 36 columns with 209592 records using which we have done few Modelling and derived mathematical summaries and we have also got many insights from the same. Let's visualize the modelling in detail.

1. Mathematical summary of the data.

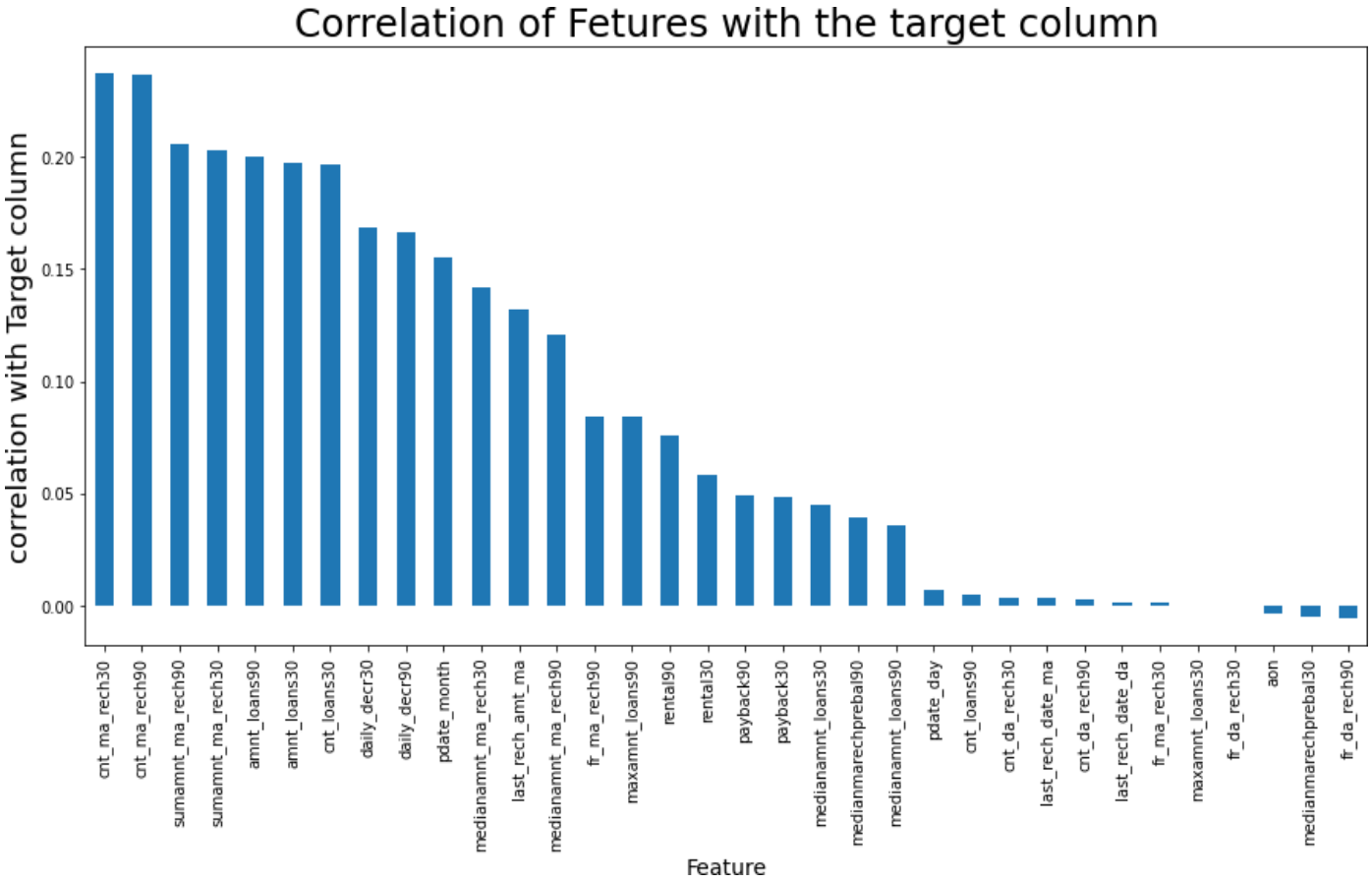
	count	mean	std	min	25%	50%	75%	max
label	209592.0	0.875177	0.330519	0.000000	1.000000	1.000000	1.0000	1.000000
aon	209592.0	8112.380399	75696.261220	-48.000000	246.000000	527.000000	982.0000	999860.755168
daily_decr30	209592.0	5381.412999	9220.644093	-93.012667	42.439500	1469.091833	7244.0960	265926.000000
daily_decr90	209592.0	6082.529123	10918.836919	-93.012667	42.691917	1500.000000	7802.7950	320630.000000
rental30	209592.0	2692.578912	4308.596841	-23737.140000	280.417500	1083.540000	3356.9450	198926.110000
rental90	209592.0	3483.407309	5770.475034	-24720.580000	300.260000	1334.000000	4201.7925	200148.110000
last_rech_date_ma	209592.0	3755.865715	53906.020204	-29.000000	1.000000	3.000000	7.0000	998650.377733
last_rech_date_da	209592.0	3712.220632	53374.960145	-29.000000	0.000000	0.000000	0.0000	999171.809410
last_rech_amt_ma	209592.0	2064.458973	2370.790003	0.000000	770.000000	1539.000000	2309.0000	55000.000000
cnt_ma_rech30	209592.0	3.978053	4.256099	0.000000	1.000000	3.000000	5.0000	203.000000
fr_ma_rech30	209592.0	3737.372947	53643.752523	0.000000	0.000000	2.000000	6.0000	999606.368132
sumamnt_ma_rech30	209592.0	7704.496570	10139.645685	0.000000	1540.000000	4628.000000	10010.0000	810096.000000
medianamnt_ma_rech30	209592.0	1812.819258	2070.869474	0.000000	770.000000	1539.000000	1924.0000	55000.000000
medianmarechprebal30	209592.0	3851.945862	54006.502647	-200.000000	11.000000	33.900000	83.0000	999479.419319
cnt_ma_rech90	209592.0	6.315437	7.193487	0.000000	2.000000	4.000000	8.0000	336.000000
fr_ma_rech90	209592.0	7.716812	12.590273	0.000000	0.000000	2.000000	8.0000	88.000000
sumamnt_ma_rech90	209592.0	12396.236149	16857.832129	0.000000	2317.000000	7226.000000	16000.0000	953036.000000
medianamnt_ma_rech90	209592.0	1864.597375	2081.685508	0.000000	773.000000	1539.000000	1924.0000	55000.000000
medianmarechprebal90	209592.0	92.025522	369.216539	-200.000000	14.600000	36.000000	79.3100	41456.500000
cnt_da_rech30	209592.0	262.579362	4183.907920	0.000000	0.000000	0.000000	0.0000	99914.441420
fr_da_rech30	209592.0	3749.512336	53885.542905	0.000000	0.000000	0.000000	0.0000	999809.240107
cnt_da_rech90	209592.0	0.041495	0.397557	0.000000	0.000000	0.000000	0.0000	38.000000
fr_da_rech90	209592.0	0.045713	0.951388	0.000000	0.000000	0.000000	0.0000	64.000000
cnt_loans30	209592.0	2.758975	2.554507	0.000000	1.000000	2.000000	4.0000	50.000000
amnt_loans30	209592.0	17.951992	17.379778	0.000000	6.000000	12.000000	24.0000	306.000000
maxamnt_loans30	209592.0	274.660029	4245.274734	0.000000	6.000000	6.000000	6.0000	99864.560864
medianamnt_loans30	209592.0	0.054029	0.218039	0.000000	0.000000	0.000000	0.0000	3.000000
cnt_loans90	209592.0	18.520988	224.797957	0.000000	1.000000	2.000000	5.0000	4997.517944
amnt_loans90	209592.0	23.645397	26.469924	0.000000	6.000000	12.000000	30.0000	438.000000
maxamnt_loans90	209592.0	6.703138	2.103869	0.000000	6.000000	6.000000	6.0000	12.000000
medianamnt_loans90	209592.0	0.046078	0.200692	0.000000	0.000000	0.000000	0.0000	3.000000
payback30	209592.0	3.398639	8.813330	0.000000	0.000000	0.000000	3.7500	171.500000
payback90	209592.0	4.321302	10.307791	0.000000	0.000000	1.666667	4.5000	171.500000
pdate_day	209592.0	14.398899	8.438899	1.000000	7.000000	14.000000	21.0000	31.000000
pdate_month	209592.0	6.797321	0.741437	6.000000	6.000000	7.000000	7.0000	8.000000

Key observations:

1. From the above data it is clear that the data has no null values.
2. Categorical Columns:
   "label"
3. Continuous Data Columns:
   Remaining all 35 Columns are continuous data
4. There is large difference between 75% percentile and Max Values which means it has more outliers.
5. Mean is greater than median which also means data have skewness present.

2. Correlation of Features with the Target column

Let's see the correlations of the data with Target variable so we can analyse depending on which feature variable the Target variable is decided.



	label		label
cnt_ma_rech30	0.237331	**payback90**	0.049178
cnt_ma_rech90	0.236393	**payback30**	0.04833
sumamnt_ma_rech90	0.205794	**medianamnt_loans30**	0.04459
sumamnt_ma_rech30	0.202828	**medianmarechprebal90**	0.0393
amnt_loans90	0.199788	**medianamnt_loans90**	0.035747
amnt_loans30	0.197272	**pdate_day**	0.006824
cnt_loans30	0.196283	**cnt_loans90**	0.004733

daily_decr30	0.168298	**cnt_da_rech30**	0.003827
daily_decr90	0.166151	**last_rech_date_ma**	0.003728
pdate_month	0.154948	**cnt_da_rech90**	0.002999
medianamnt_ma_rech30	0.141491	**last_rech_date_da**	0.001711
last_rech_amt_ma	0.131805	**fr_ma_rech30**	0.00133
medianamnt_ma_rech90	0.120855	**maxamnt_loans30**	0.000248
fr_ma_rech90	0.084386	**fr_da_rech30**	-2.6E-05
maxamnt_loans90	0.084144	**aon**	-0.00379
rental90	0.075521	**medianmarechprebal30**	-0.00483
rental30	0.058084	**fr_da_rech90**	-0.00541

Key Observations:

1. From above we can observe that **"cnt_ma_rech30", "cnt_ma_rech90", "sumamnt_ma_rech90", "sumamnt_ma_rech30"** have more correlation with the Target variable "**label**".
2. We have more positive correlated variable than the negative correlated variable.

• Data Sources and their formats

The Data is provided to us from the Client Database which has 209593 records of mobile phone user in Indonesia. These records are about the mobile phone user does transaction with the mobile network service providers such as recharges done in 30 days and 90 days, loan taken within last 30 and 90 days, the trend of repayment for every loan taken and also the tenure of the users in the same network. These Data are collected and stored in CSV format. Which we are using for the study and analysis.

• Data Pre-processing Done

We started our pre-processing pipeline in importing the required libraries and we imported the given data.

[2]:
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

importing the Data.

[3]:
```
df = pd.read_csv("Data file.csv", index_col=0, parse_dates=['pdate'])
df
```

[3]:

	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	...	maxamn
1	0	21408170789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	...	
2	1	76462170374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	5787	...	
3	1	17943170372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539	...	
4	1	55773170781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	947	...	
5	1	03813182730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309	...	
...	...	...	...	...	...	...	...	...	...	...	...	
209589	1	22758185348	404.0	151.872333	151.872333	1089.19	1089.19	1.0	0.0	4048	...	
209590	1	95583184455	1075.0	36.936000	36.936000	1728.36	1728.36	4.0	0.0	773	...	
209591	1	28556185350	1013.0	11843.111667	11904.350000	5861.83	8893.20	3.0	0.0	1539	...	
209592	1	59712182733	1732.0	12488.228333	12574.370000	411.83	984.58	2.0	38.0	773	...	
209593	1	65061185339	1581.0	4489.362000	4534.820000	483.92	631.20	13.0	0.0	7526	...	

209593 rows × 36 columns

Then we removed the duplicate data from our records and also identified that '**msisdn'** and '**pcircle'** have unique values and same value respectively for all the columns, so I have drooped these two columns to make the data more predictable.

```
[4]: df.duplicated().sum()
```

```
[4]: 1
```

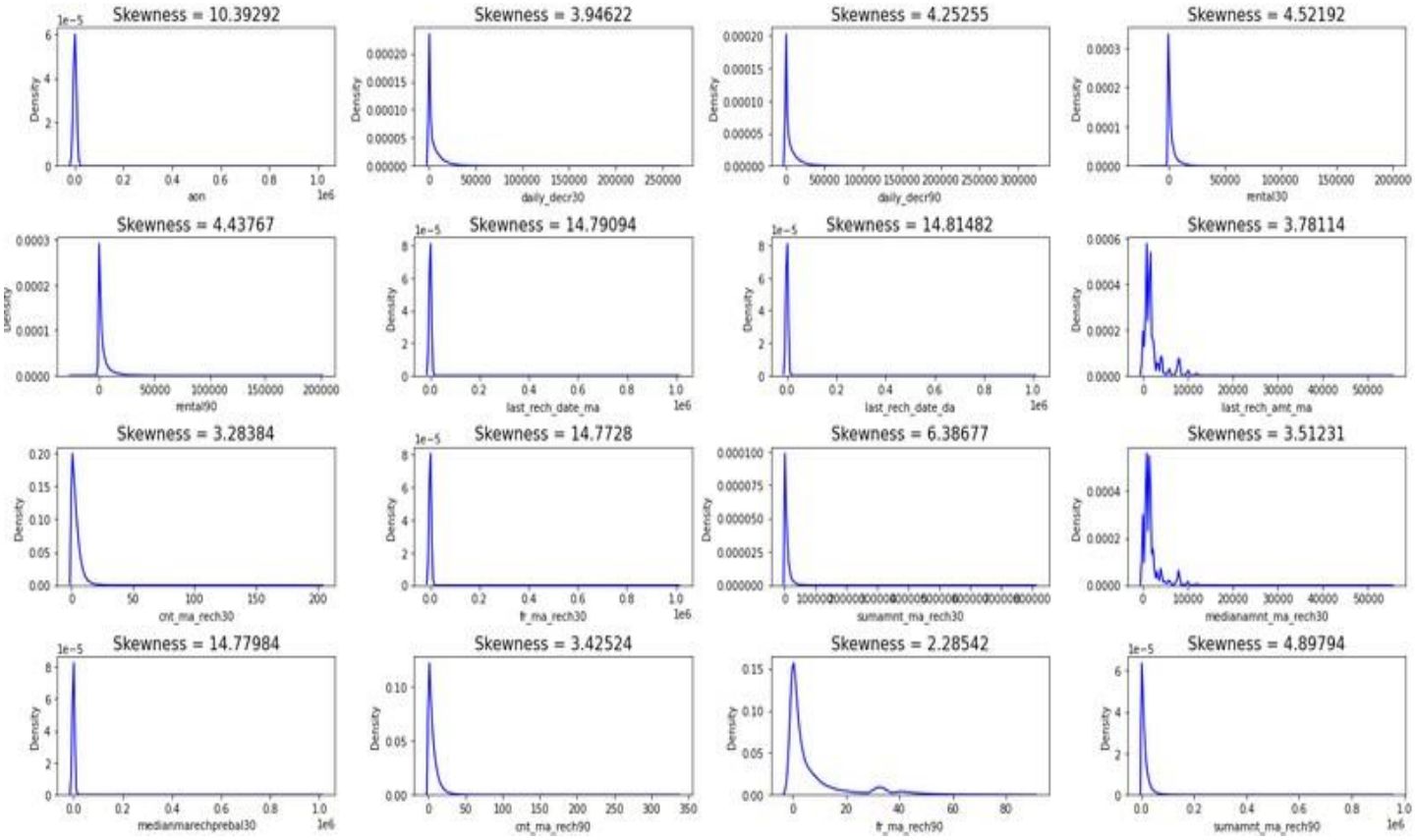
```
[5]: df= df.drop_duplicates()
```

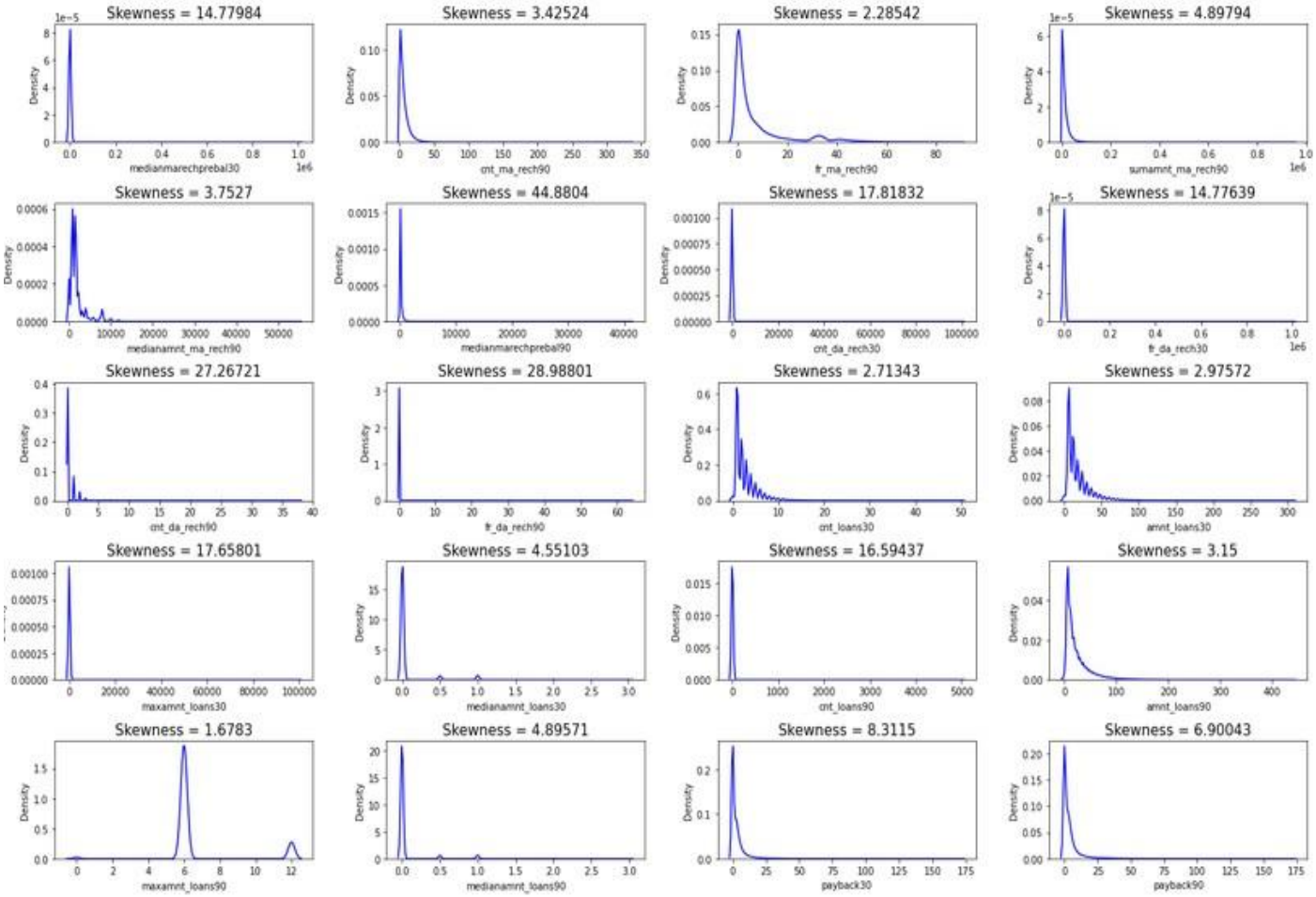
```
[7]: df.drop(["msisdn","pcircle"], axis =1, inplace = True)
```

From the given data we have observed that data is taken from year 2016 and mostly of 06th, 07th, 08th months and I have split the '**pdate'** columns into **'pdate_day'** and '**pdate_month'** to reduce the complexity and dropped **'pdate'**.

```
[9]: day = df.pdate.dt.day.to_frame()
month = df.pdate.dt.month.to_frame()

date= day.join(month, how='right', lsuffix='_day', rsuffix="_month")
df = df.join(date, how= "left")
df.drop("pdate", inplace = True, axis = 1)
df
```

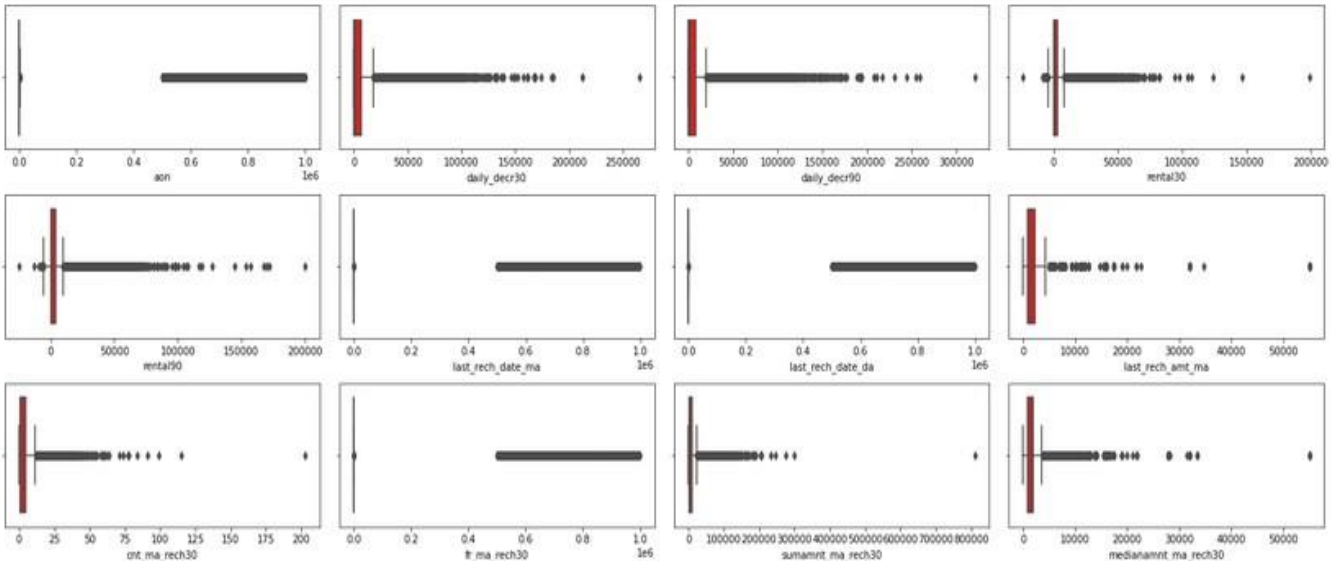
After cleansing the data for correcting the skewness and outliers I have plotted the following method.

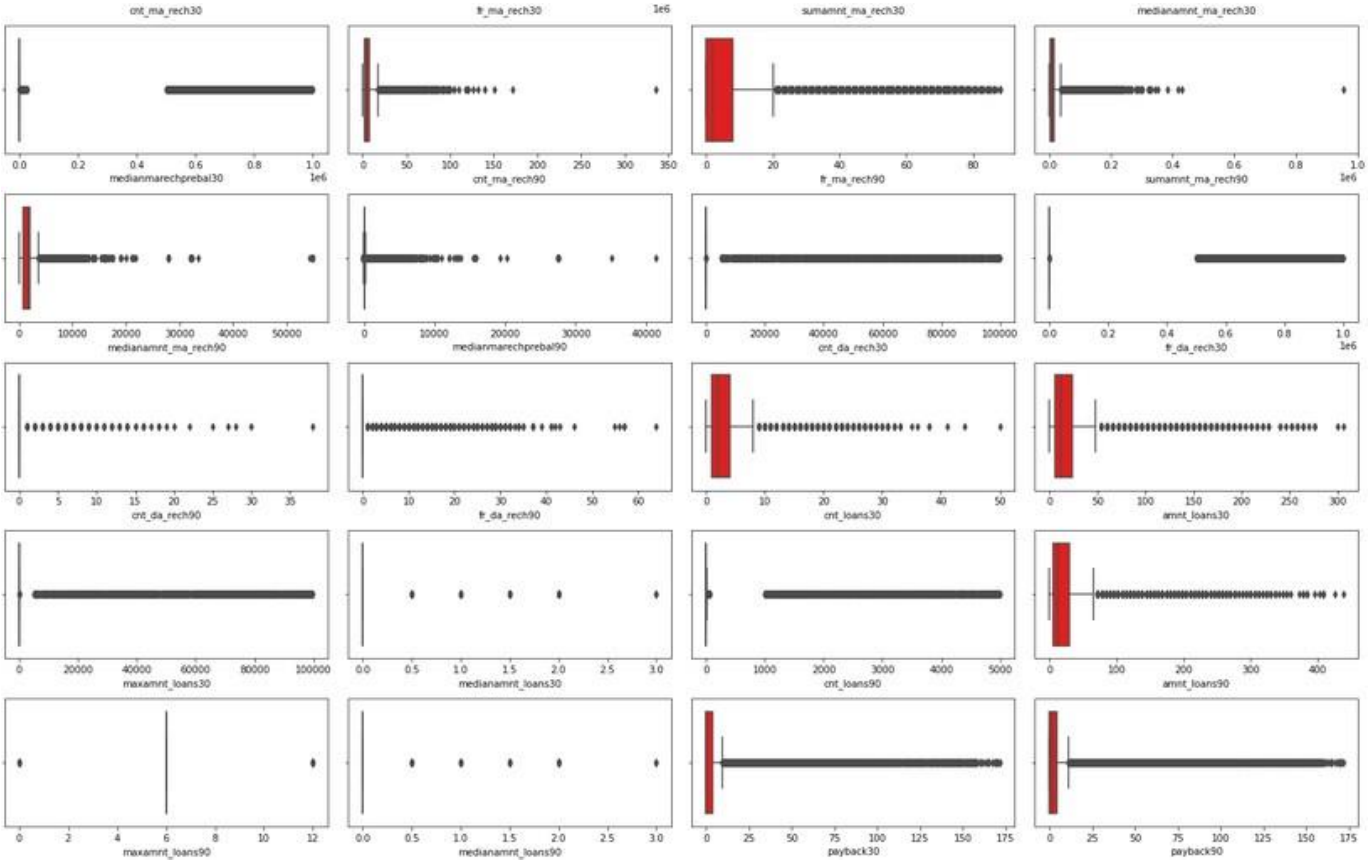




As observed the overall data is totally positive skewed before correcting the skewness, I tried identifying the Outliers and correcting the same without much loss in data.

[18]:
```
plt.figure(figsize=(20,20))
for i in range (0, len(collist)):
    plt.subplot(8,4,i+1)
    ax=sns.boxplot(df[collist[i]], color = "red")
    ax.set_facecolor("w")
    plt.tight_layout()
```





Observed data have too much outliers I tried Z-score method and Quantile method to rectify the same.

[19]:
```
from scipy.stats import zscore
z= np.abs(zscore(df))
threshold= 3
df_new = df[(z < 3).all(axis=1)]
```

[20]:
```
print(f"Orginal Data {df.shape}\n After Removing outliers {df_new.shape}\nThe percentage of data loss {((209592-161465),
```

```
Orginal Data (209592, 35)
 After Removing outliers (161465, 35)
The percentage of data loss 22.96223138287721%
```

The loss of data is greater than 22% in Z-Score removal methods, we cannot afford to loss that much data. Let's see about Quantile method.

[21]:
```
Q1=df.quantile(0.25)
Q3=df.quantile(0.75)
IQR=Q3-Q1

df_new1=df[~((df<(Q1-1.5*IQR))|(df>(Q3+1.5*IQR))).any(axis=1)]
```

[22]:
```
print(f"Orginal Data {df.shape}\nAfter Removing outliers {df_new1.shape}\nThe percentage of data loss {((209592-56628)/
```

```
Orginal Data (209592, 35)
After Removing outliers (56628, 35)
The percentage of data loss 72.981793198213 68%
```

The loss of data in Quantile method is 72% the data loss in Quantile method is also very high so we have to work with Outliers present in the data.

For correcting the skewness without deforming the data I have used Power transformation method (Yeo-Johnson) which as follows.

```
[23]: x_1=df.drop(["label"], axis = 1)
      y_1=df.label
```

```
[24]: from sklearn.preprocessing import power_transform
      x_1=power_transform(x_1,method="yeo-johnson")
      x_1
```

```
:[24]: array([[-0.17711202,  0.33215731,  0.29918259, ...,  1.78398404,
                0.71082238,  0.39752454],
              [ 0.03645225,  1.11601957,  1.04508281, ..., -1.01891435,
               -0.41014406,  1.48012963],
              [-0.03465955, -0.007506  , -0.02820064, ..., -1.01891435,
                0.60997236,  1.48012963],
              ...,
              [ 0.13457591,  1.10045362,  1.03342251, ...,  0.75295343,
                1.54675826,  0.39752454],
              [ 0.30910014,  1.13605548,  1.06837956, ...,  1.3378482 ,
                1.18938807,  0.39752454],
              [ 0.27706161,  0.52502202,  0.48715722, ..., -1.01891435,
               -0.81940564,  0.39752454]])
```

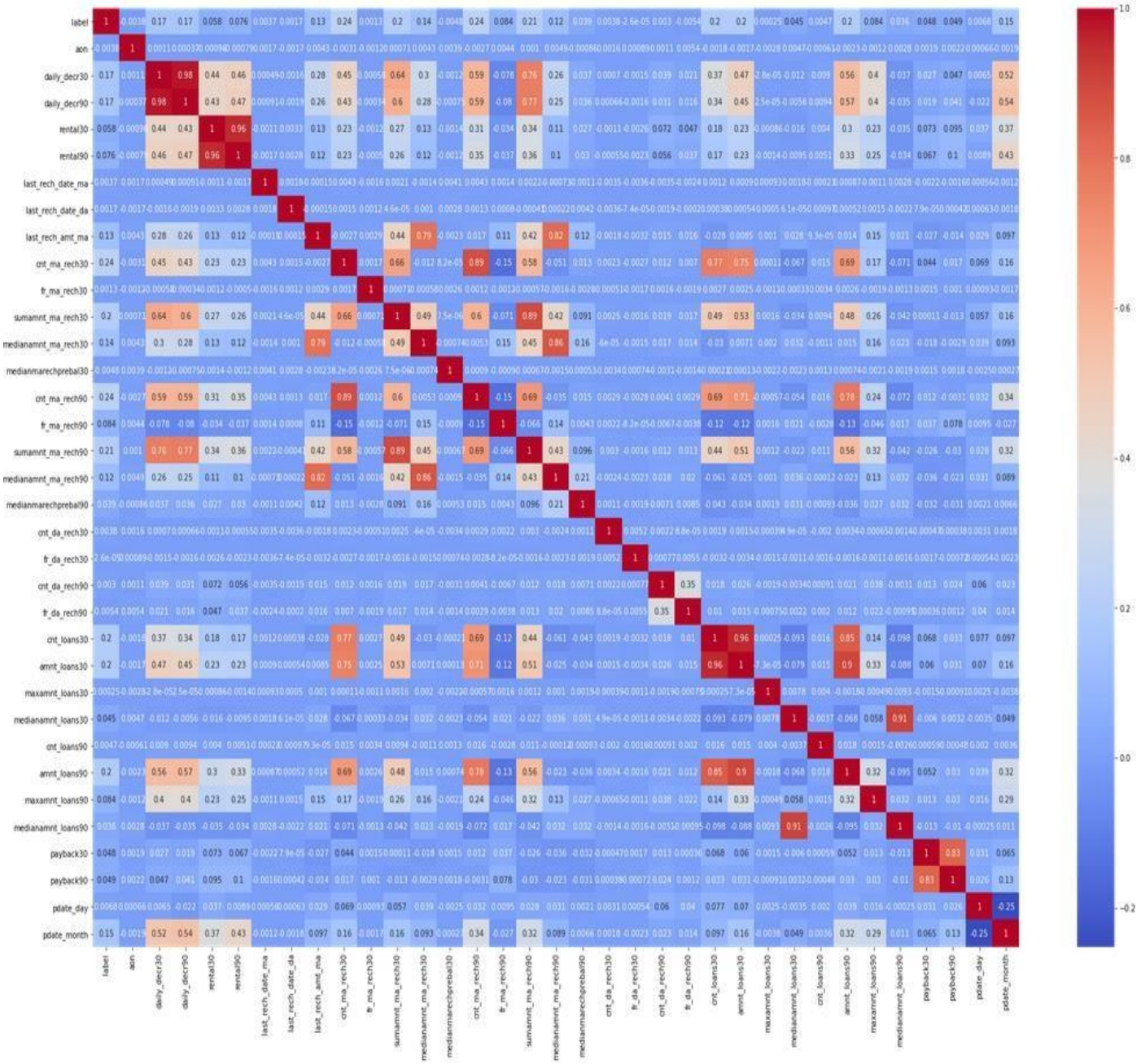
The Pre-Processing of the data is completed with the skewness removal, data is completely ready for split and train with the model, before which I have tried in understanding the logic of output based on the input and also the from the data and insights that is derived from the data and also with the correlation of all the data, I have tried in getting pre-assumed insight ideas on which the output is defined. We are seeing about these pre-assumed ideas with the tools used in deriving these ideas are as follows.

• Data Inputs- Logic- Output Relationships

From the Data we have already gained few insights which are as follows,

1. **daily_dec30, daily_dec90, rental30, rental90, cnt_da_rech30, sumamnt_ma_rech30, cnt_ma_rech90, sumamnt_ma_rech90, medianmarechprebal90, cnt_da_rech90, cnt_loans30, amnt_loans30, amnt_loans90,** the above columns have more contributions in 0 that is defaulter.
2. **aon, last_rech_date_ma, last_rech_date_da, fa_ma_rech30, medianmarechprebal30, medianamnt_ma_rech90, cnt_da_rech30, fr_da_rech30, maxamnt_loan30, cnt_loans90** have more contributions in 1.
3. **payback90, payback30, medianamnt_loans90, fr_da_reach90, medianamnt_ma_reach90, medianamnt_ma_reach30, last_rech_amt_ma** almost have equal contributions in 0 and 1.

Let's also see the correlation of the Data and let's see some insights in correlation.



Key Insights from Correlation:

- From above we can observe that "**cnt_ma_rech30", "cnt_ma_rech90", "sumamnt_ma_rech90", "sumamnt_ma_rech30"** have more correlation with the Target variable **"label".** Let's visualize the feature variable correlation with target variable separately.
- **'daily_decr30', 'daily_decr90', 'rental30', 'rental90', 'last_rech_amt_ma', 'cnt_ma_rech30', 'sumamnt_ma_rech90', 'medianamnt_ma_rech90', 'cnt_loans30', 'amnt_loans30', 'amnt_loans90', 'maxamnt_loans90',**

payback30, payback90, pdate_day, pdate_month have more correlation with each other, which is these variables are dependent on each other.

• Pre-assumptions of the Data.

With all the insights taken we have plotted few pre-assumptions which are as follows.

1. As like said above **Label** '1' indicates that the loan has been paid i.e., **Non- defaulter**, while, **Label '0'** indicates that the loan has not been paid i.e., **defaulter**.
2. **daily_dec30, daily_dec90, rental30, rental90, cnt_da_rech30, sumamnt_ma_rech30, cnt_ma_rech90, sumamnt_ma_rech90, medianmarechprebal90, cnt_da_rech90, cnt_loans30, amnt_loans30, amnt_loans90,** the above columns have more contributions in 0 that is defaulter, which means that customer have more contributions on above factors end up as defaulter.
3. **aon, last_rech_date_ma, last_rech_date_da, fa_ma_rech30, medianmarechprebal30, medianamnt_ma_rech90, cnt_da_rech30, fr_da_rech30, maxamnt_loan30, cnt_loans90** have more contributions in 1. which means that customer have more contributions on above factors end up as not a defaulter.
4. **payback90, payback30, medianamnt_loans90, fr_da_reach90, medianamnt_ma_reach90, medianamnt_ma_reach30, last_rech_amt_ma** almost have equal contributions in 0 and 1.

• Hardware and Software Requirements and Tools Used

- Python 3.9.2
- Anaconda (Juypter Note Book)
- Matplotlib
- Seaborn
- Scikit Learn
- Pandas
- Data Science
- Machine Learning
- NumPy

Model/s Development and Evaluation

- ## Identification of possible problem-solving approaches (methods)

From the data we observe that our Target Variable **'label'** has two variables 0 and 1, 0 – defines Defaulter and 1 – defines not a Defaulter. So, we understand that we have to train our model with classification algorithms. Before training our model, I have filtered the best random state parameter as following.

[25]:
```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report


accu = 0
for i in range(0,1000):
    x_train, x_test, y_train, y_test = train_test_split(x_1,y_1,test_size = .25, random_state = i,  stratify=y_1 )
    mod = GaussianNB()
    mod.fit(x_train,y_train)
    y_pred = mod.predict(x_test)
    tempacc = accuracy_score(y_test,y_pred)
    if tempacc> accu:
        accu= tempacc
        best_rstate=i

print(f"Best Accuracy {accu*100} found on randomstate {best_rstate}")
```

```
Best Accuracy 74.12878354135654 found on randomstate 156
```

[26]:
```
x_train, x_test, y_train, y_test = train_test_split(x_1,y_1,test_size = .25, random_state = best_rstate, stratify=y_1)
```

After finding best random state I have split the data into two parts train data and test data lets also see about the all the Classification algorithm on which am going to train the model.

- ## Testing of Identified Approaches (Algorithms)

I have trained our model with eight different Classification Algorithms with ensemble technique which are as follows.

- "Logistic Regression"
- "Naive Bayes Gaussian"
- "Random Forest",
- "Decision Tree",
- "Extra Tree",
- "Ada Boost",
- "Gradient Boosting",
- "XGBoost".

- ## Run and Evaluate selected models

As said, I have trained our model after splitting with eight different algorithms and I have shortlisted the best algorithm with the following classification metrics such as F1-Score, Accuracy, Pression, Recall. Let's see our evaluation as follows.

[27]:
```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_validate, cross_val_predict, StratifiedKFold
from sklearn.feature_selection import RFECV
from sklearn.linear_model import SGDClassifier, LogisticRegression
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier, ExtraTreeClassifier
from sklearn import metrics
import xgboost as xgb
```

Lets shortlist promising Classification models.

[28]:
```
def mod_sco(estimator, x_train, y_train, cv=5, verbose=True):

    scoring = {"accuracy": "accuracy",
               "precision": "precision_weighted",
               "recall": "recall_weighted",
               "f1": "f1_weighted"}
    scores = cross_validate(estimator, x_train, y_train, cv=cv, scoring=scoring)

    accuracy, accuracy_std = scores['test_accuracy'].mean(),scores['test_accuracy'].std()
    precision, precision_std = scores['test_precision'].mean(),scores['test_precision'].std()
    recall, recall_std = scores['test_recall'].mean(),scores['test_recall'].std()
    f1, f1_std = scores['test_f1'].mean(), scores['test_f1'].std()


    result = {
        "Accuracy": accuracy,
        "Accuracy std": accuracy_std,
        "Precision": precision,
        "Precision std": precision_std,
        "Recall": recall,
        "Recall std": recall_std,
        "f1": f1,
        "f1 std": f1_std,
    }

    if verbose:
        print(f"Accuracy: {accuracy} - (std: {accuracy_std})")
        print(f"Precision: {precision} - (std: {precision_std})")
        print(f"Recall: {recall} - (std: {recall_std})")
        print(f"f1: {f1} - (std: {f1_std})")

    return result
```

[29]:
```
models = [LogisticRegression(), GaussianNB(), RandomForestClassifier(random_state=42),
          DecisionTreeClassifier(random_state=42), ExtraTreeClassifier(random_state=42),
          AdaBoostClassifier(random_state=42), GradientBoostingClassifier(random_state=42),
          xgb.XGBClassifier(silent=True)]

model_names = ["LogisticRegression", "Naive Bayes Gaussian", "Random Forest",
               "Decision Tree", "Extra Tree", "Ada Boost",
               "Gradient Boosting", "XGBoost"]
```

[30]:
```
accuracy = []
precision = []
recall = []
f1 = []

for model in range(len(models)):
    print(f"\n\nStep {model+1} of {len(models)}")
    print(f".....running {model_names[model]}")

    clf_scores = mod_sco(models[model], x_train, y_train)

    accuracy.append(clf_scores["Accuracy"])
    precision.append(clf_scores["Precision"])
    recall.append(clf_scores["Recall"])
    f1.append(clf_scores["f1"])
```

Evaluation started with importing the required libraries and with help of cross validation I have filtered and shortlisted the best algorithm model and I have hyper tunes the same.

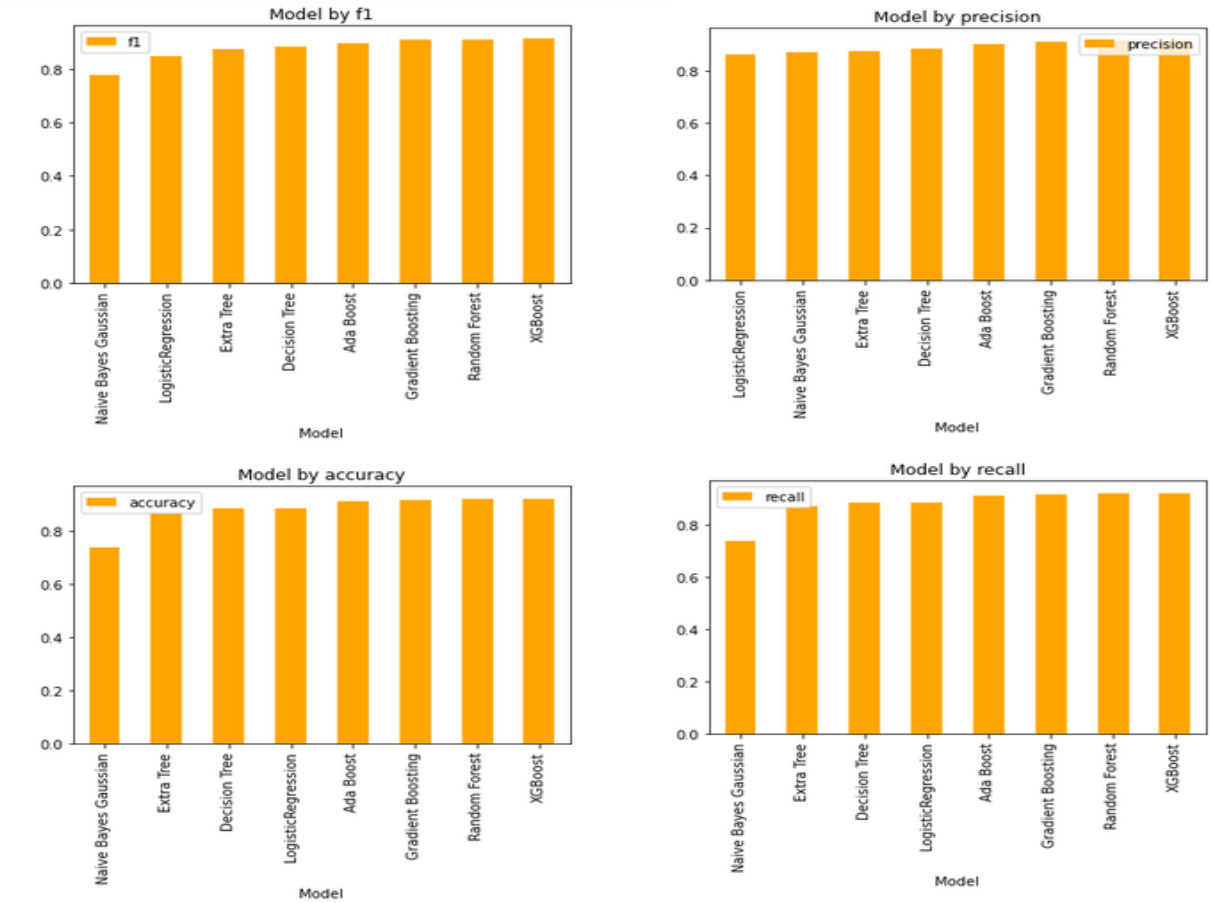
We will visualize the shortlisting of best performed model as follows.

[31]:
```
classification_result = pd.DataFrame({"Model": model_names,
                                       "accuracy": accuracy,
                                       "precision": precision,
                                       "recall": recall,
                                       "f1": f1})
classification_result.sort_values(by="f1", ascending=False)
```

:[31]:

	Model	accuracy	precision	recall	f1
7	XGBoost	0.922764	0.916827	0.922764	0.917606
2	Random Forest	0.920512	0.913824	0.920512	0.912892
6	Gradient Boosting	0.918699	0.911660	0.918699	0.910102
5	Ada Boost	0.910054	0.901830	0.910054	0.895978
3	Decision Tree	0.883157	0.885844	0.883157	0.884439
4	Extra Tree	0.873787	0.875459	0.873787	0.874604
0	LogisticRegression	0.883882	0.861792	0.883882	0.849212
1	Naive Bayes Gaussian	0.739227	0.871468	0.739227	0.780338

[32]:
```
metrics_list = ["f1", "accuracy", "precision", "recall"]

for metric in metrics_list:
    classification_result.sort_values(by=metric).plot.bar("Model", metric, color = "orange")
    plt.title(f"Model by {metric}")
    plt.show()
```



From above we see that Xg Boost model have top the chart at F1-Score of 91.7606% and accuracy of 92.2764%. I have further hyper tuned the same Xg Boost model for better performance.

- ## Key Metrics for success in solving problem under consideration

We saw that we have shortlisted the best model as XGBoost model, we will now see metrics for the same.

	Model	accuracy	precision	recall	f1
7	XGBoost	0.922764	0.916827	0.922764	0.917606
2	Random Forest	0.920512	0.913824	0.920512	0.912892
6	Gradient Boosting	0.918699	0.911660	0.918699	0.910102
5	Ada Boost	0.910054	0.901830	0.910054	0.895978
3	Decision Tree	0.883157	0.885844	0.883157	0.884439
4	Extra Tree	0.873787	0.875459	0.873787	0.874604
0	LogisticRegression	0.883882	0.861792	0.883882	0.849212
1	Naive Bayes Gaussian	0.739227	0.871468	0.739227	0.780338

[39]:
```
preds = cross_val_predict(clf_xgb, x_train, y_train, cv=5, n_jobs=-1)
pd.crosstab(y_train, preds, rownames = ['Real'], colnames =['Predicted'])
```

[39]:

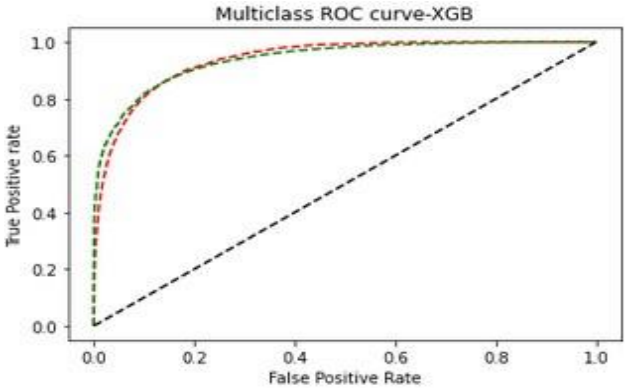
Real \ Predicted	0	1
0	10962	8659
1	3482	134091

[40]:
```
print(metrics.classification_report(y_train, preds, zero_division=0))
```

```
              precision    recall  f1-score   support

           0       0.76      0.56      0.64     19621
           1       0.94      0.97      0.96    137573

    accuracy                           0.92    157194
   macro avg       0.85      0.77      0.80    157194
weighted avg       0.92      0.92      0.92    157194
```

[36]:
```
fpr = {}
tpr = {}
thresh ={}

n_class = 2

for i in range(n_class):
    fpr[i], tpr[i], thresh[i] = roc_curve(y_test, y_pred_prob[:,i], pos_label=i)
```

[37]:
```
plt.plot(fpr[0], tpr[0], linestyle='--',color='red', label='Class 0 vs Rest')
plt.plot(fpr[1], tpr[1], linestyle='--',color='green', label='Class 1 vs Rest')
plt.title('Multiclass ROC curve-XGB')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive rate')
plt.plot([0,1],[0,1],'k--')
plt.show()
```

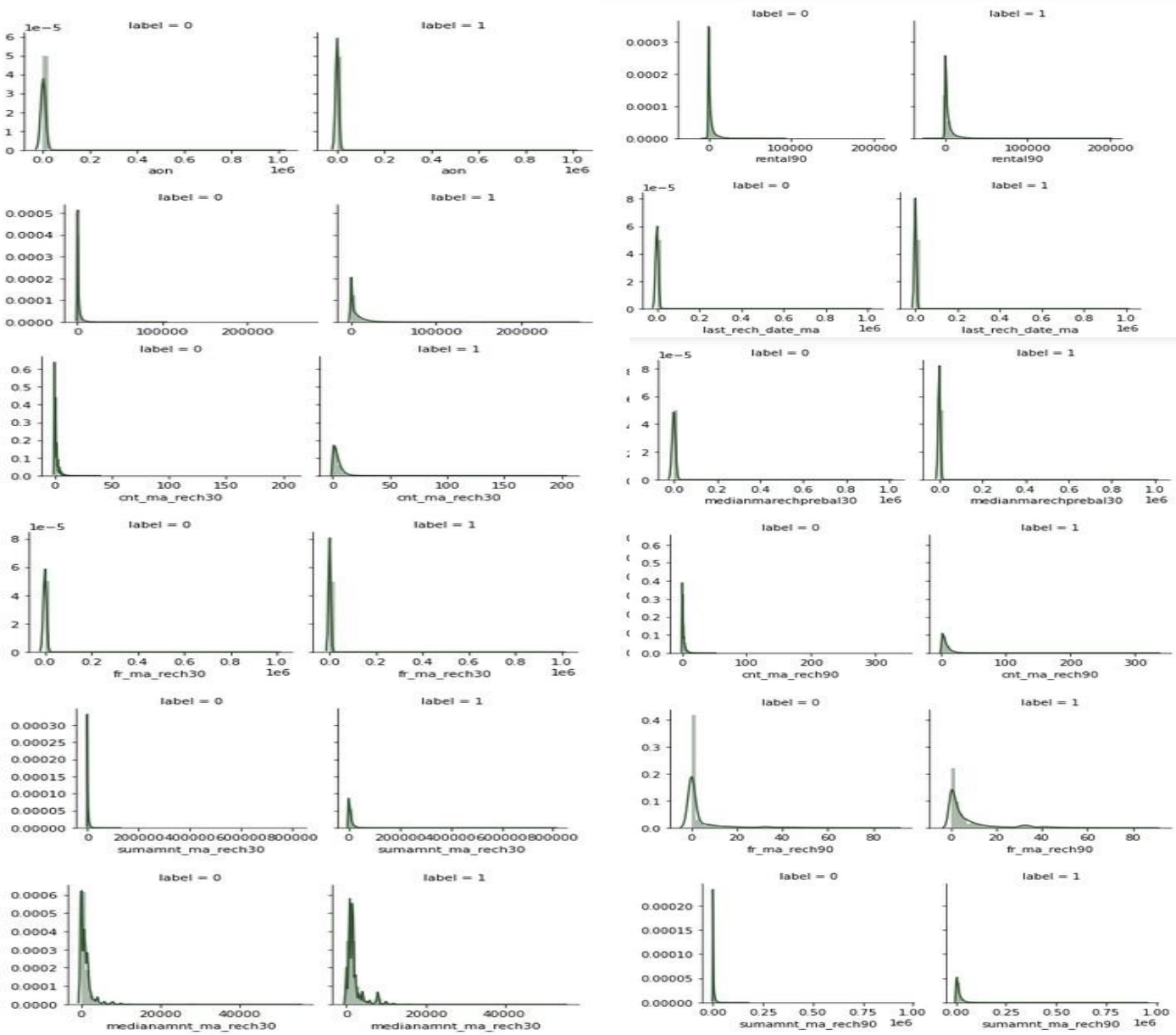


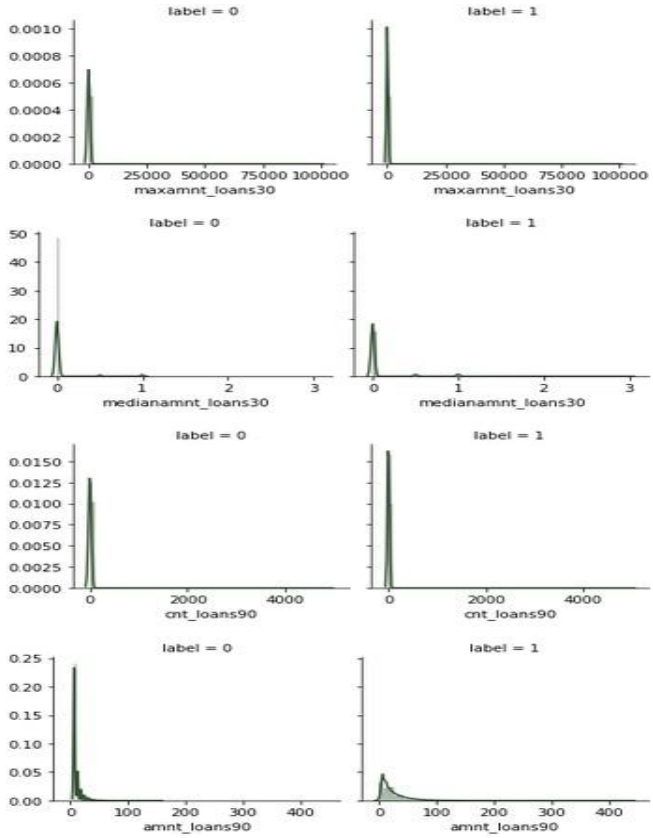
From above metrics we can understand how Xgboost model have performed and also overfitting in the model is very minimal since the model have accuracy of 92%. We also saw the ROC-AUC curve which clearly explains the performance of the model.

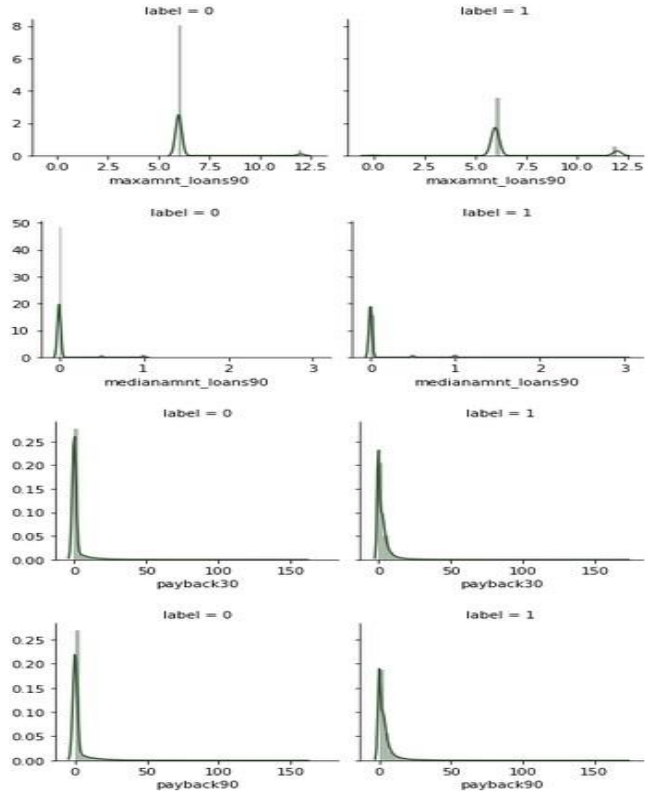
- Visualizations

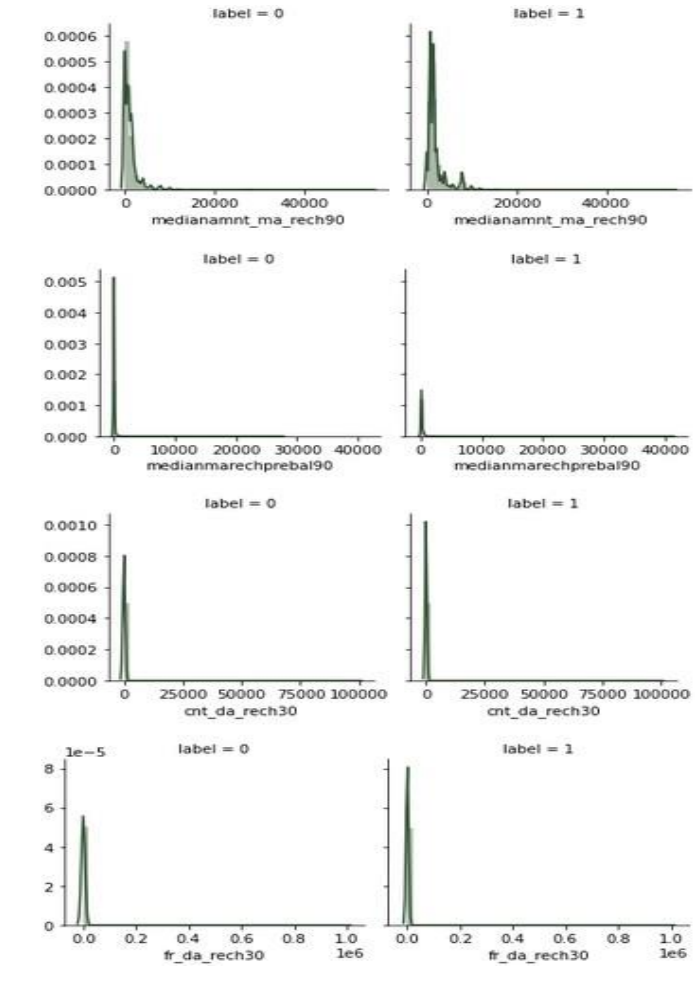
Data Insights

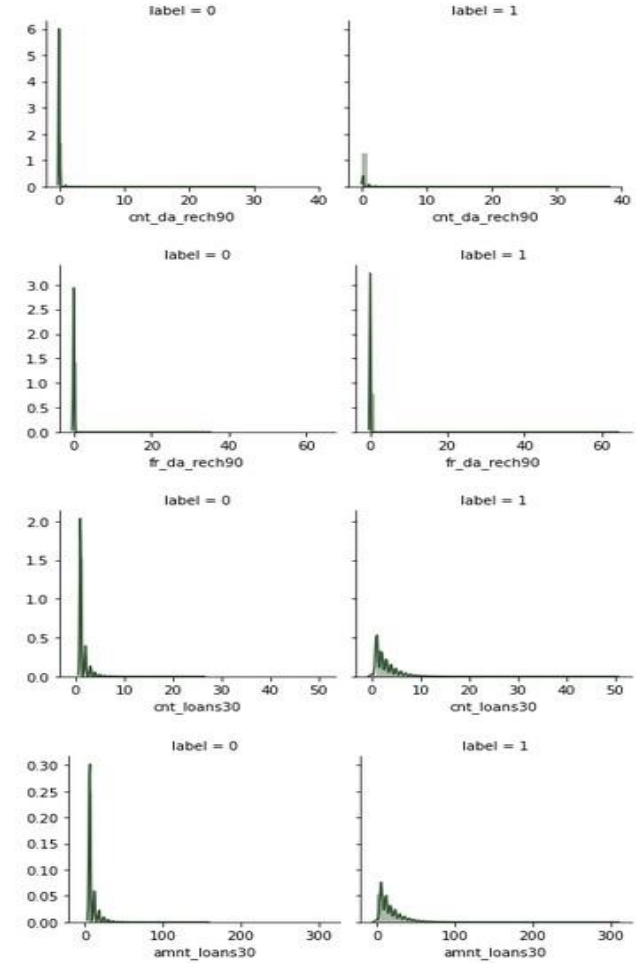
From all the Data available, we can bring out some neat insights or conclusions such as.




label = 0
label = 1
maxamnt_loans30
medianamnt_loans30
cnt_loans90
amnt_loans90


label = 0
label = 1
maxamnt_loans90
medianamnt_loans90
payback30
payback90


label = 0
label = 1
medianamnt_ma_rech90
medianmarechprebal90
cnt_da_rech30
fr_da_rech30


label = 0
label = 1
cnt_da_rech90
fr_da_rech90
cnt_loans30
amnt_loans30

Key observations:

1. As like said above *Label* '1' indicates that the loan has been paid i.e., *Non-defaulter*, while, *Label '0'* indicates that the loan has not been paid i.e., *defaulter*.
2. *daily_dec30, daily_dec90, rental30, rental90, cnt_da_rech30, sumamnt_ma_rech30, cnt_ma_rech90, sumamnt_ma_rech90, medianmarechprebal90, cnt_da_rech90, cnt_loans30, amnt_loans30, amnt_loans90,* the above columns have more contributions in 0 that is defaulter.
3. *aon, last_rech_date_ma, last_rech_date_da, fa_ma_rech30, medianmarechprebal30, medianamnt_ma_rech90, cnt_da_rech30, fr_da_rech30, maxamnt_loan30, cnt_loans90* have more contributions in 1.
4. *payback90, payback30, medianamnt_loans90, fr_da_reach90, medianamnt_ma_reach90, medianamnt_ma_reach30, last_rech_amt_ma* almost have equal contributions in 0 and 1.

• Interpretation of the Results

As per our pre- assumption and visualization that we have observed the 209593 records of data of Telecom customers and also as per our machine learning model we can understand how a customer is ending up being a defaulter and also how a customer are ending up being a non-defaulter from the given factors.

The customers who maintain lesser average balance in the account in 30days and 90 days and lesser number of loans taken by a customer ending up being not a defaulter. Customer who onboard with same network for longer period of time and high number of loans taken customer are not being a defaulter. But customer who does high recharge for data in 30 to 90 days, and customer who maintain high balance, who have taken lesser number of loans are ending up being a defaulter.

CONCLUSION

• Key Findings and Conclusions of the Study

From our above analysis we understand that which factors are responsible and using which a Microfinance Institution (MFI) can predict that a customer can be a defaulter or not a defaulter basis on which Microfinance Institution (MFI) can give a short-term loan and predictions that could help them in further investment and improvement in selection of customers.

• Limitations of this work and Scope for Future Work

Limitations of the study arise from the use of a single dataset obtained from one company. However, the large number of loans and customers considered and generic applicability of credit scoring for mobile credit suggests that the variables and models investigated are relevant for other business applications. Further work may include the use of demographic information which could be obtained from mobile network operators and consideration of additional pay-as-you-go mobile products.

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

As per our pre- assumption and visualization that we have observed the 209593 records of data of Telecom customers and also as per our machine learning model we can understand how a customer is ending up being a defaulter and also how a customer are ending up being a non-defaulter from the given factors.

The customers who maintain lesser average balance in the account in 30days and 90 days and lesser number of loans taken by a customer ending up being not a defaulter. Customer who onboard with same network for longer period of time and high number of loans taken customer are not being a defaulter. But customer who does high recharge for data in 30 to 90 days, and customer who maintain high balance, who have taken lesser number of loans are ending up being a defaulter.

Challenges faced in improving the performance of the training model are as follows we have shortlisted XGBoost model with the f1-Score: 0.917606 and also, further in process of increasing the performance I have done the hyper parameter tuning with Grid search CV as like follows.

HYPERPARAMETER TUNING....

[41]:
```
param_grid = [
    {'learning_rate' : [1e-3, 1e-1, 'log-uniform'],
     'n_estimators': [100, 500],
     'max_depth': [1, 10],
     'min_child_weight': [1, 6.],
     'gamma': [0, 0.5],
     'subsample':[0.5, 1.],
     'colsample_bytree': [0.5, 1.]
    }
]


clf_xgb =  xgb.XGBClassifier(silent=True)
```

[42]:
```
grid_search = GridSearchCV(clf_xgb, param_grid, cv=3,
                           scoring="f1_weighted", verbose=2, n_jobs=-1)
```

[43]:
```
grid_search.fit(x_train, y_train)
```

```
Fitting 3 folds for each of 192 candidates, totalling 576 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:  6.8min
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed: 33.3min
[Parallel(n_jobs=-1)]: Done 357 tasks      | elapsed: 95.2min
[Parallel(n_jobs=-1)]: Done 576 out of 576 | elapsed: 161.2min finished
[14:38:23] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:541:
Parameters: { silent } might not be used.
```

Post the tuning the model the performance are as follows.

- Accuracy: 0.923616667027628 - (std: 0.000970341942248066)
- Precision: 0.9177505278202464 - (std: 0.0011439892828334953)
- Recall: 0.923616667027628 - (std: 0.000970341942248066)
- f1: 0.9182938370683678 - (std: 0.0010890123712770127)

Post the hyperparameter tuning I have achieved model f1 - score increase from f1: 0.917606 to f1: 0.91829 with accuracy of 92.36%. Thus, we have finally achieved maximum .performed model for the prediction of Defaulters and non-Defaulters for mobile financial services (MFS).