FLIP ROBO

NAME OF THE PROJECT

MICRO CREDIT DEFAULTER MODEL

Submitted by:

REENA

ACKNOWLEDGEMENT

I would like to express my special gratitude to the “Flip Robo” team, who has given me this opportunity to deal with a beautiful dataset and it has helped me to improve my analyzing skills. Also, I want to express my huge gratitude to Ms. Khushboo Garg (SME FlipRobo), who has helped me overcome difficulties within this project and others.

CONTENTS

1. INTRODUCTION

1.1 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. 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.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. 1.2 Conceptual Background of the Domain Problem:Telecom Industries 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. They are collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed to be defaulter if he deviates from the path of paying back the loaned amount within the time duration of 5 days. For the loan amount of 5 (in Indonesian Rupiah), payback amount should be 6 (inIndonesian Rupiah), while, for the loan amount of 10 (in Indonesian Rupiah), the payback amount should be 12 (in Indonesian Rupiah). We have to build a model 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 payed i.e. Nondefaulter, while, Label ‘0’ indicates that the loan has not been payed i.e. defaulter.

1.3 Review of Literature

Microfinance in India has seen incredible growth in the last two decades in terms of the number of Microfinance customers and institutions offering microfinance. An attempt has been made in this report to review the available literature in microfinance. Approaches to microfinance, issues related to measuring social impact versus profitability of MFIs, issue of sustainability, variables impacting sustainability, effect of regulations of profitability and impact assessment of MFIs have been summarized in the below report. We hope that the below report of literature will provide a platform for further research and help the industry to combine theory and practice to take microfinance forward and contribute to alleviating the poor from poverty. The findings and recommendations of existing research work can help in identifying appropriate ML algorithms for credit assessment specifically for rural borrowers. This study can also help fintech startups, banking and non-banking financial institutions in India to develop more financially inclusive products.

1.4 Motivation for the Problem Undertaken:

I have to model the micro credit defaulters with the available independent variables. This model will then be used by the management to understand how the customer is considered as defaulter or non-defaulter based on the independent variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns.

Further, the model will be a good way for the management to understand whether the customer will be paying back the loaned amount within 5 days of insurance of loan. The relationship between predicting defaulter and the economy is an important motivating factor for predicting micro credit defaulter model. Furthermore, studies have indicated that analysis of social media data can also help in building predictive models for rural settings where there has been a penetration of mobile applications and online banking applications.
Lastly, some research methodologies have also explored deep learning models such as artificial neural networks (ANN) for micro-credit score prediction.

2. Analytical Problem

Framing

2.1 Mathematical/Analytical Modeling of the Problem

In this particular problem I had label as my target column and it was having two classes Label '1' indicates that the loan has been payed i.e., non-defaulter, while Label '0' indicates that the loan has not been payed i.e., defaulter. So clearly it is a binary classification problem, and I must use all classification algorithms while building the model. There were no null value in the dataset. Also, I observed some unnecessary entries in some of the columns like in some columns I found more than 90% zero values, so I decided to drop those columns. If I keep those columns as it is, it will create high skewness in the model. To get better insight on the features I have used plotting like distribution plot, bar plot and count plot. With these plotting I was able to understand the relation between the features in better manner. Also, I found outliers and skewness in the dataset, so I removed outliers using percentile method

and I removed skewness using yeo-Johnson method. I have used all the classification algorithms while building model then tunned the best model and saved the best model. At
last, I have predicted the label using saved model.
2.2 Data Sources and their formats:The data was collected for my internship company – Flip Robo technologies in excel format. The sample data is provided to us from our client database. It is hereby given to us for this
exercise. To improve the selection of customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers. Also, my dataset was having 209593 rows and 36 columns including target. In this particular dataset, I have object, float, and integer types of data. The information about features is as follows
Features Information:
1. label : Flag indicating whether the user paid back the credit amount within 5 days of
issuing the loan{1:success, 0:failure}
2. msisdn : mobile number of user
3. aon : age on cellular network in days
4. daily_decr30 : Daily amount spent from main account, averaged over last 30 days (in
Indonesian Rupiah)
5. daily_decr90 : Daily amount spent from main account, averaged over last 90 days (in
Indonesian Rupiah)
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 (in Indonesian Rupiah)
11. cnt_ma_rech30 : Number of times main account got recharged in last 30 days
12. fr_ma_rech30 : Frequency of main account recharged in last 30 days
13. sumamnt_ma_rech30 : Total amount of recharge in main account over last 30 days (in
Indonesian Rupiah)
14. medianamnt_ma_rech30 : Median of amount of recharges done in main account over
last 30 days at user level (in Indonesian Rupiah)
15. medianmarechprebal30 : Median of main account balance just before recharge in last
30 days at user level (in Indonesian Rupiah)
16. cnt_ma_rech90 : Number of times main account got recharged in last 90 days
17. fr_ma_rech90 : Frequency of main account recharged in last 90 days
18. sumamnt_ma_rech90 : Total amount of recharge in main account over last 90 days
19. medianamnt_ma_rech90 : Median of amount of recharges done in main account over
last 90 days at user level (in Indonasian Rupiah)
20. medianmarechprebal90 : Median of main account balance just before recharge in last
90 days at user level (in Indonasian Rupiah)
21. cnt_da_rech30 : Number of times data account got recharged in last 30 days

22. fr_da_rech30: Frequency of data account recharged in last 30 days
23. cnt_da_rech90 : Number of times data account got recharged in last 90 days
24. fr_da_rech90 : Frequency of data account recharged 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 : maximum amount of loan taken by the user 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

2.3 Data Processing done

* Imported necessary Libraries and loaded the dataset.
* Statistical Analysis done like Shape, info, nunique, value_counts.
* Dropped columns with more than 90% of Zero values.
* No Null values in the dataset.
* Dropped column- Unnamed:0, msisdn and pcircle which had all unique or 1 unique value.
* Feature extraction done in pdate and extracted pday, pmonth and pyear.
* Converted few columns with negative values to positive values.
* Checked Correlation between Variables and Feature
* Outliers removed, Encoded categorical columns.
* Skewness removed, removed columns with Multicollinearity issue.
* Using Min-Max Scaler, normalized the data.

2.4 Data Inputs- Logic- Output Relationships

• Since I had all numerical columns, I have plotted dist plot to see the distribution of
each column data.

• I have used box plot for each pair of categorical features that shows the relation
between label and independent features. Also we can observe whether the person
pays back the loan within the date based on features.

• In maximum features relation with target, I observed Non-defaulter count is high
compared to defaulters.

<u>3 Hardware and Software Requirements and Tools Used</u>

* Hardware: Processor- Core i5 and above, RAM- 8GB or above, SSD- 250 or above

* Software: Anaconda

* Libraries Used mentioned before.

There are 3 sets of libraries used.

● Basic Libraries for Data Analysis and Visualization

● Libraries for Data Cleaning and Feature Engineering (Data preprocessing)

● Libraries for Building the ML Models

2. MODEL DEVELOPMENT AND EVALUATION

3.1 Identification of possible problem-solving approaches

To remove outliers, I have used percentile method. And to remove skewness I have used yeo-Johnson method. We have dropped all the unnecessary columns in the dataset according to our understanding. Use of Pearson's correlation coefficient to check the correlation between dependent and independent features. Also, I have used Normalization to scale the data. After scaling we must balance the target column using oversampling.

Then followed by model building with all Classification algorithms. I have used oversampling (SMOTE) to get rid of data imbalancing. The balanced output looks like this.

3.2 Testing of Identified Approaches (Algorithms)

Since label was my target and it was a classification column with 0-defaulter and 1-Nondefaulter, so this particular problem was Classification problem. And I have used all Classification algorithms to build my model. By looking into the difference of accuracy score and cross validation score I found RandomForestClassifier as a best model with least difference. Also to get the best model we have to run through multiple models and to avoid the confusion of overfitting we have go through cross validation. Below are the list of classification algorithms I have used in my project.

- Decision Tree Model
- Random Forest Classifier
- KNN Classifier
- XGB Classifier

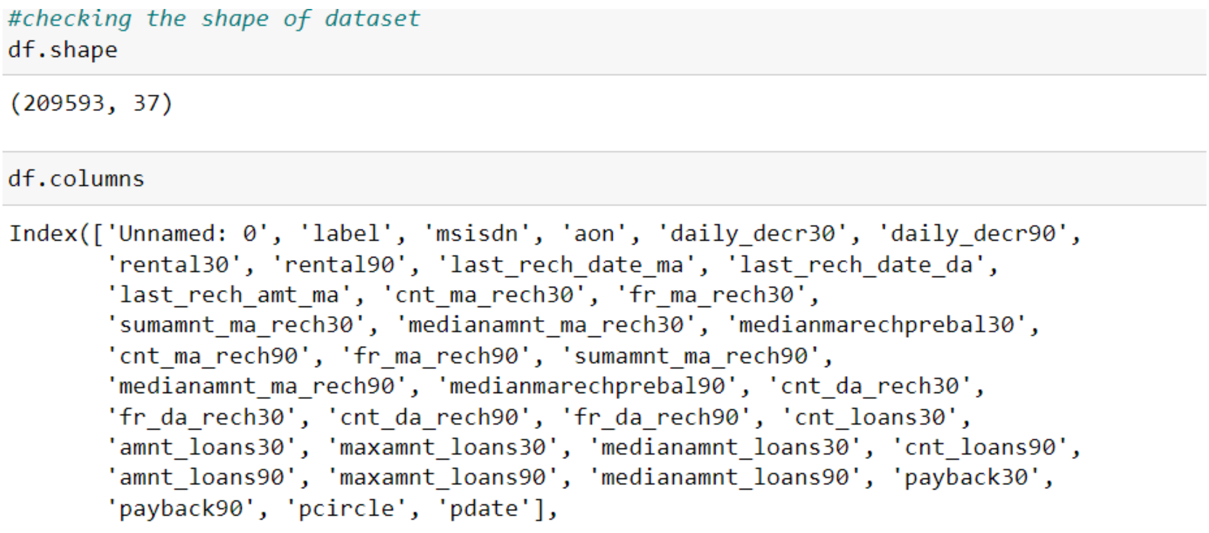
3.3 Run and evaluate selected models

A. Decision Tree Classifier Model:

I found the best random state which yields the best score and then fit the model. Accuracy score for Decision Tree Model was 91.57%. I have tuned with different parameters and the score has decreased to 80.86%. The CV Score of Decision tree model without tuned parameters are 91.14%. The Confusion Matrix of actual and predicted values using Decision Tree model is:

B. Random Forest Regressor (Best Model):

I first found the best random state and fit the model. I got a score of 95.26 % and a CV Score of 94.92%. Hence, I selected the random forest as the best model and saved the model. While performing the fitting of the final model, my score was improved, and it became 95.59%.

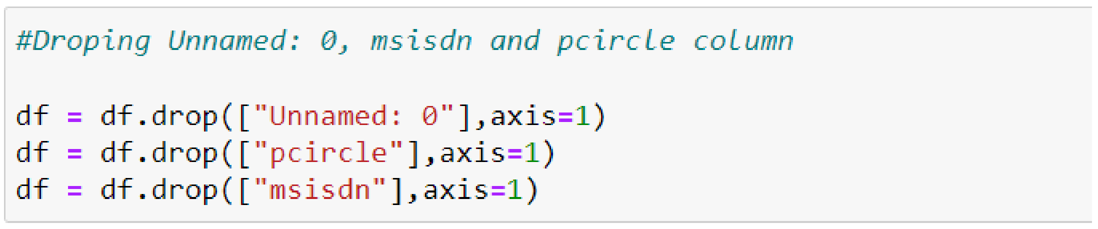


```
#checking the shape of dataset
df.shape
```

```
(209593, 37)
```

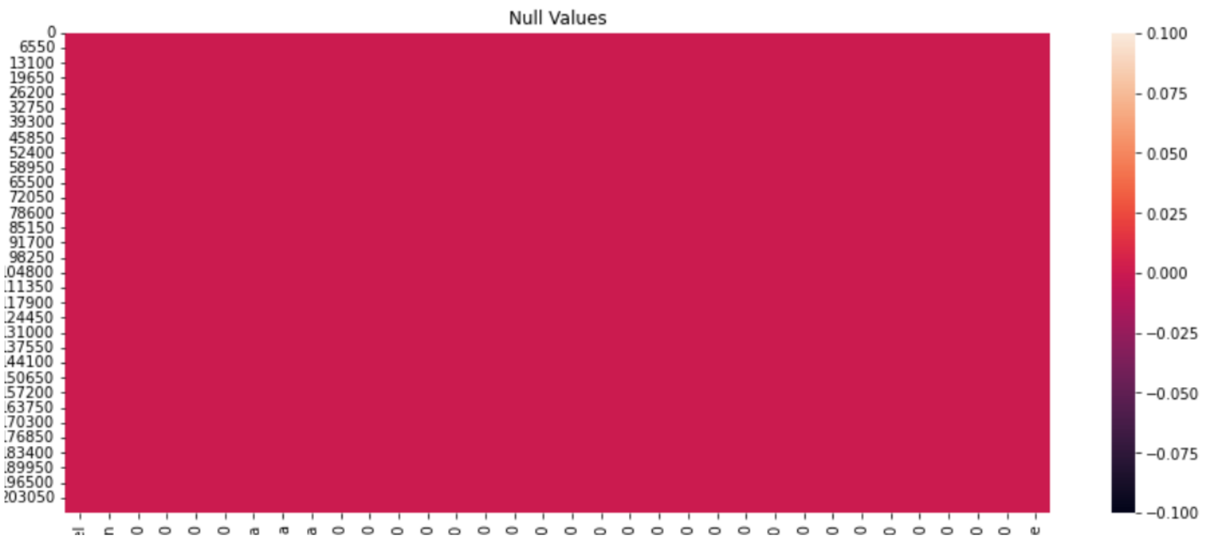
```
df.columns
```

```
Index(['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_ma_rech30',
       'sumamnt_ma_rech30', 'medianamnt_ma_rech30', 'medianmarechprebal30',
       'cnt_ma_rech90', 'fr_ma_rech90', 'sumamnt_ma_rech90',
       'medianamnt_ma_rech90', 'medianmarechprebal90', 'cnt_da_rech30',
       'fr_da_rech30', 'cnt_da_rech90', 'fr_da_rech90', 'cnt_loans30',
       'amnt_loans30', 'maxamnt_loans30', 'medianamnt_loans30', 'cnt_loans90',
       'amnt_loans90', 'maxamnt_loans90', 'medianamnt_loans90', 'payback30',
       'payback90', 'pcircle', 'pdate'],
```



```
#Droping Unnamed: 0, msisdn and pcircle column

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

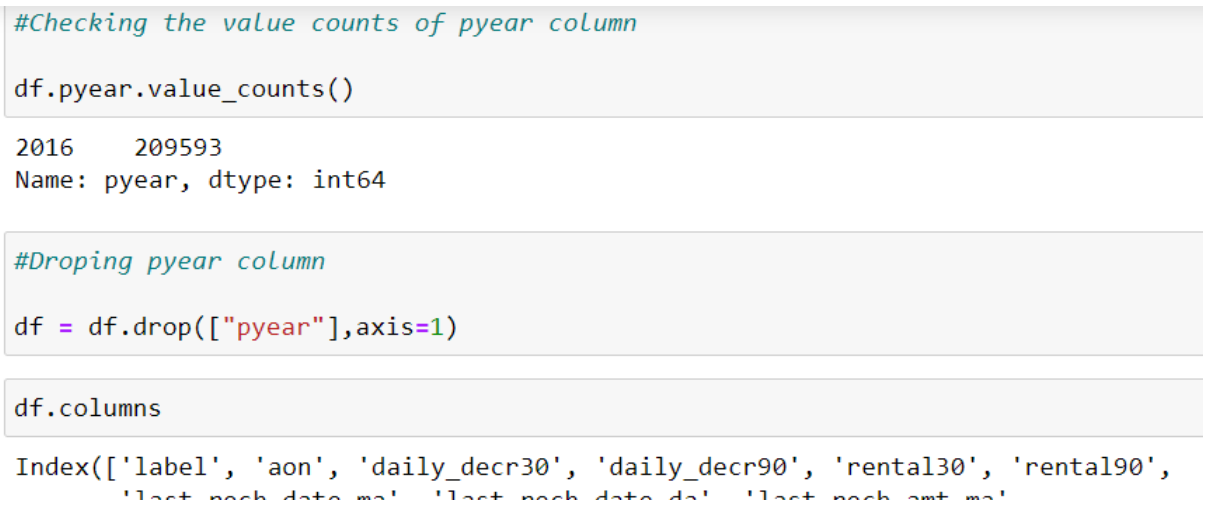
```
#Droping Unnamed: 0, msisdn and pcircle column

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



2. Feature Engineering

```
#Converting object data type to datetime

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

```
#Extracting paid year,month and day from pdate

#Extracting year
df["pyear"]=pd.to_datetime(df.pdate, format="%d/%m/%Y").dt.year
```



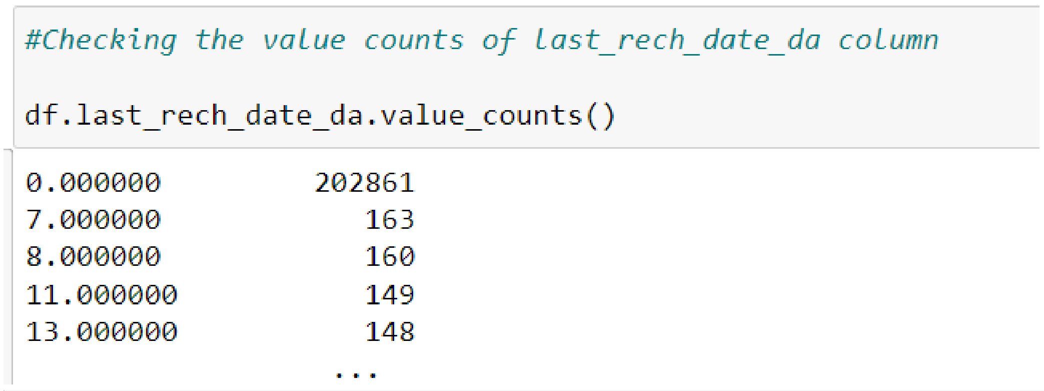
```
#Checking the value counts of pyear column

df.pyear.value_counts()
```

```
2016    209593
Name: pyear, dtype: int64
```

```
#Droping pyear column

df = df.drop(["pyear"],axis=1)
```

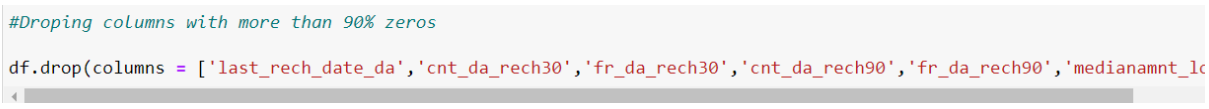
```
df.columns
```

```
Index(['label', 'aon', 'daily_decr30', 'daily_decr90', 'rental30', 'rental90',
       'last_rech_date_ma', 'last_rech_date_da', 'last_rech_amt_ma',
```

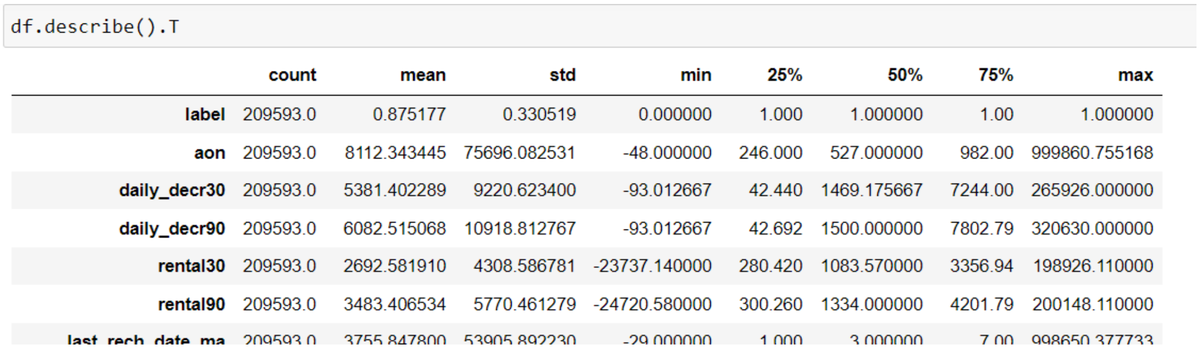


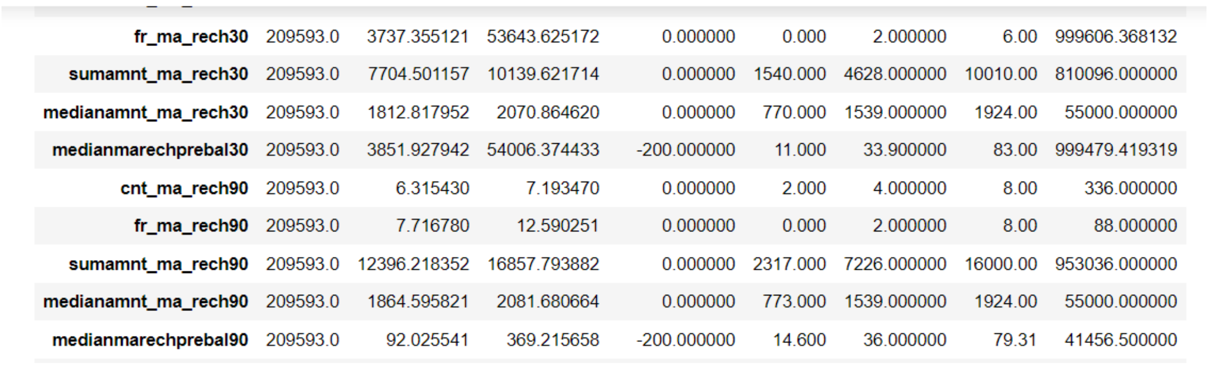
```
#Checking the value counts of last_rech_date_da column

df.last_rech_date_da.value_counts()
```

```
0.000000        202861
7.000000           163
8.000000           160
11.000000          149
13.000000          148
                  ...
```



```
#Droping columns with more than 90% zeros

df.drop(columns = ['last_rech_date_da','cnt_da_rech30','fr_da_rech30','cnt_da_rech90','fr_da_rech90','medianamnt_lo
```





```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
label	209593.0	0.875177	0.330519	0.000000	1.000	1.000000	1.00	1.000000
aon	209593.0	8112.343445	75696.082531	-48.000000	246.000	527.000000	982.00	999860.755168
daily_decr30	209593.0	5381.402289	9220.623400	-93.012667	42.440	1469.175667	7244.00	265926.000000
daily_decr90	209593.0	6082.515068	10918.812767	-93.012667	42.692	1500.000000	7802.79	320630.000000
rental30	209593.0	2692.581910	4308.586781	-23737.140000	280.420	1083.570000	3356.94	198926.110000
rental90	209593.0	3483.406534	5770.461279	-24720.580000	300.260	1334.000000	4201.79	200148.110000
last_rech_date_ma	209593.0	3755.847800	53905.892230	-29.000000	1.000	3.000000	7.00	998650.377733
fr_ma_rech30	209593.0	3737.355121	53643.625172	0.000000	0.000	2.000000	6.00	999606.368132
sumamnt_ma_rech30	209593.0	7704.501157	10139.621714	0.000000	1540.000	4628.000000	10010.00	810096.000000
medianamnt_ma_rech30	209593.0	1812.817952	2070.864620	0.000000	770.000	1539.000000	1924.00	55000.000000
medianmarechprebal30	209593.0	3851.927942	54006.374433	-200.000000	11.000	33.900000	83.00	999479.419319
cnt_ma_rech90	209593.0	6.315430	7.193470	0.000000	2.000	4.000000	8.00	336.000000
fr_ma_rech90	209593.0	7.716780	12.590251	0.000000	0.000	2.000000	8.00	88.000000
sumamnt_ma_rech90	209593.0	12396.218352	16857.793882	0.000000	2317.000	7226.000000	16000.00	953036.000000
medianamnt_ma_rech90	209593.0	1864.595821	2081.680664	0.000000	773.000	1539.000000	1924.00	55000.000000
medianmarechprebal90	209593.0	92.025541	369.215658	-200.000000	14.600	36.000000	79.31	41456.500000

**Observations:

elow columns have negative values: aon : age on cellular network in days daily_decr30 : Daily amount spent from main account, averag (in Indonesian Rupiah) daily_decr90 : Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah) rental3 account balance over last 30 days rental90 : Average main account balance over last 90 days last_rech_date_ma : Number of days till l account medianmarechprebal30 : Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupi medianmarechprebal90 : Median of main account balance just before recharge in last 90 days at user level (in Indonesian Rupiah) The be negative as they are age, account balance and number of days. So we will change them to positive.

Most of the columns have 0 as the minimum value. Mean and Median is different for almost all columns. There is a high chance of skew dataset in almost all columns.

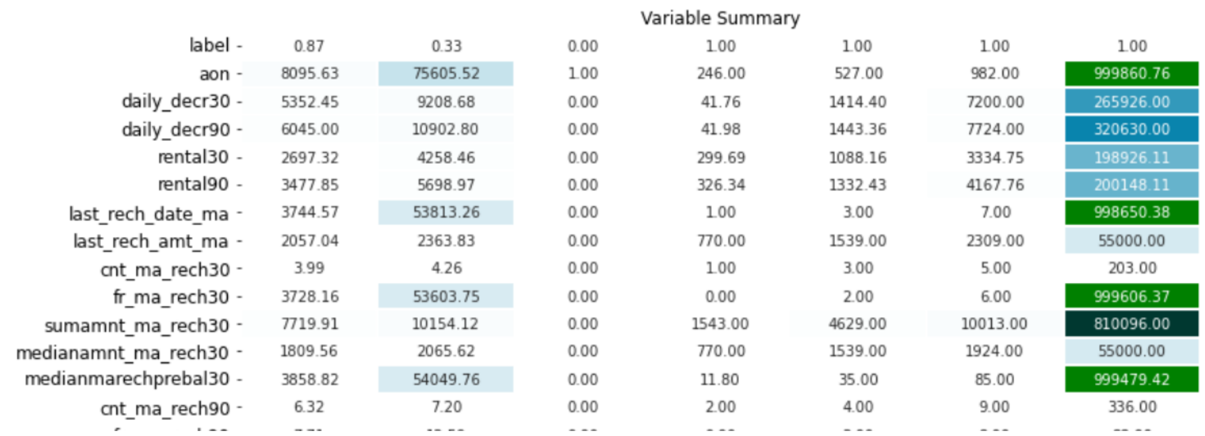
```
#Converting all negative values to positive values in above columns

df['aon']=abs(df['aon'])
df['daily_decr30']=abs(df['daily_decr30'])
df['daily_decr90']=abs(df['daily_decr90'])
df['rental30']=abs(df['rental30'])
df['rental90']=abs(df['rental90'])
df['last_rech_date_ma']=abs(df['last_rech_date_ma'])
df['medianmarechprebal30']=abs(df['medianmarechprebal30'])
df['medianmarechprebal90']=abs(df['medianmarechprebal90'])
```

I have successfully converted all negative values to positive values.

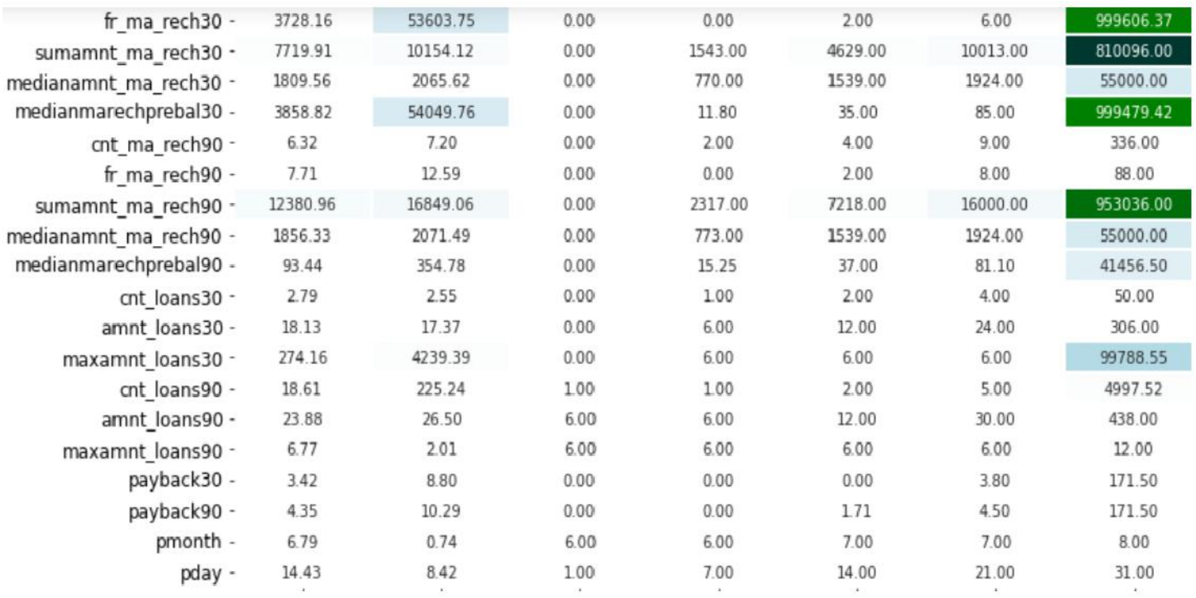
```
#droping the people who haven't taken any loan

df.drop(df[df['amnt_loans90']==0].index, inplace = True)
```

Droping people who haven't taken any loans as we don't have any use from them.

```
#Heatmap of the Describe function

plt.figure(figsize=(15,10))
sns.heatmap(round(df.describe()[1:].transpose(),2),linewidth=2,annot=True,fmt='.2f',cmap='ocean_r')
plt.xticks(fontsize=16)
plt.yticks(fontsize=12)
plt.title('Variable Summary')
plt.show()
```


Variable Summary

label	0.87	0.33	0.00	1.00	1.00	1.00	1.00
aon	8095.63	75605.52	1.00	246.00	527.00	982.00	999860.76
daily_decr30	5352.45	9208.68	0.00	41.76	1414.40	7200.00	265926.00
daily_decr90	6045.00	10902.80	0.00	41.98	1443.36	7724.00	320630.00
rental30	2697.32	4258.46	0.00	299.69	1088.16	3334.75	198926.11
rental90	3477.85	5698.97	0.00	326.34	1332.43	4167.76	200148.11
last_rech_date_ma	3744.57	53813.26	0.00	1.00	3.00	7.00	998650.38
last_rech_amt_ma	2057.04	2363.83	0.00	770.00	1539.00	2309.00	55000.00
cnt_ma_rech30	3.99	4.26	0.00	1.00	3.00	5.00	203.00
fr_ma_rech30	3728.16	53603.75	0.00	0.00	2.00	6.00	999606.37
sumamnt_ma_rech30	7719.91	10154.12	0.00	1543.00	4629.00	10013.00	810096.00
medianamnt_ma_rech30	1809.56	2065.62	0.00	770.00	1539.00	1924.00	55000.00
medianmarechprebal30	3858.82	54049.76	0.00	11.80	35.00	85.00	999479.42
cnt_ma_rech90	6.32	7.20	0.00	2.00	4.00	9.00	336.00



fr_ma_rech30	3728.16	53603.75	0.00	0.00	2.00	6.00	999606.37
sumamnt_ma_rech30	7719.91	10154.12	0.00	1543.00	4629.00	10013.00	810096.00
medianamnt_ma_rech30	1809.56	2065.62	0.00	770.00	1539.00	1924.00	55000.00
medianmarechprebal30	3858.82	54049.76	0.00	11.80	35.00	85.00	999479.42
cnt_ma_rech90	6.32	7.20	0.00	2.00	4.00	9.00	336.00
fr_ma_rech90	7.71	12.59	0.00	0.00	2.00	8.00	88.00
sumamnt_ma_rech90	12380.96	16849.06	0.00	2317.00	7218.00	16000.00	953036.00
medianamnt_ma_rech90	1856.33	2071.49	0.00	773.00	1539.00	1924.00	55000.00
medianmarechprebal90	93.44	354.78	0.00	15.25	37.00	81.10	41456.50
cnt_loans30	2.79	2.55	0.00	1.00	2.00	4.00	50.00
amnt_loans30	18.13	17.37	0.00	6.00	12.00	24.00	306.00
maxamnt_loans30	274.16	4239.39	0.00	6.00	6.00	6.00	99788.55
cnt_loans90	18.61	225.24	1.00	1.00	2.00	5.00	4997.52
amnt_loans90	23.88	26.50	6.00	6.00	12.00	30.00	438.00
maxamnt_loans90	6.77	2.01	6.00	6.00	6.00	6.00	12.00
payback30	3.42	8.80	0.00	0.00	0.00	3.80	171.50
payback90	4.35	10.29	0.00	0.00	1.71	4.50	171.50
pmonth	6.79	0.74	6.00	6.00	7.00	7.00	8.00
pday	14.43	8.42	1.00	7.00	14.00	21.00	31.00

3. Data Visualization

3.1 Univariate Analysis

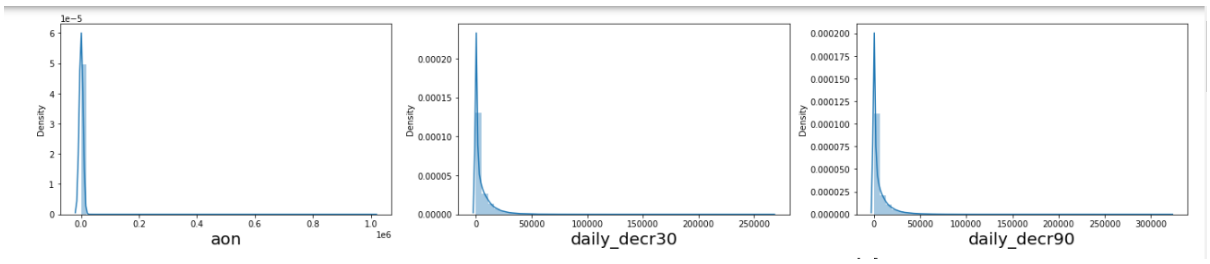
```
# checking for categorical columns

cat_cols=[]
for i in df.dtypes.index:
    if df.dtypes[i]=='object':
        cat_cols.append(i)
print(cat_cols)
```

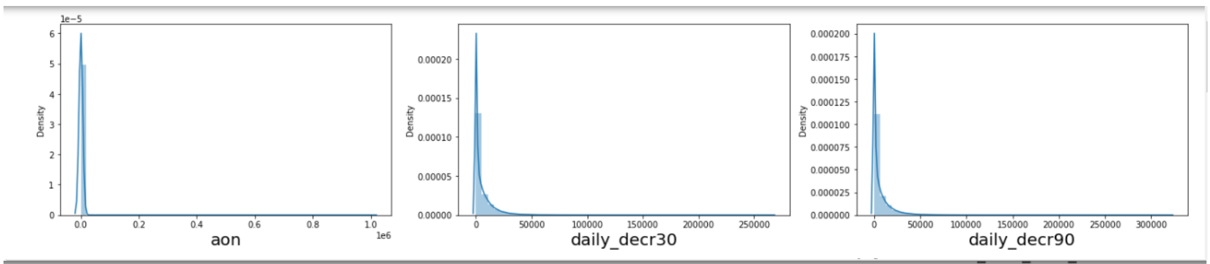
```
[]
```

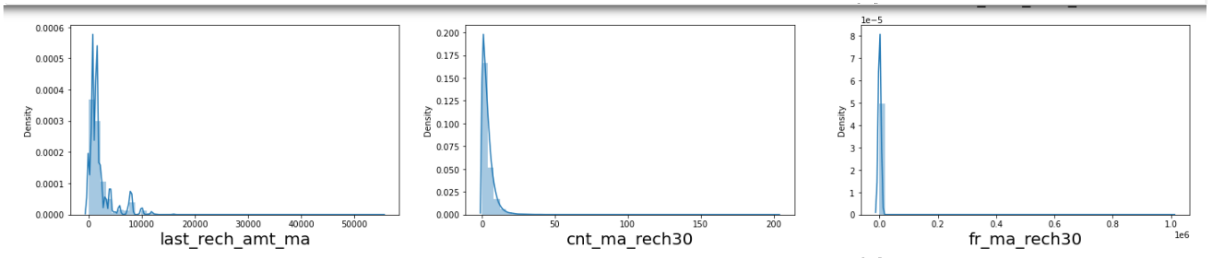
There is no categorical data in the given dataset.

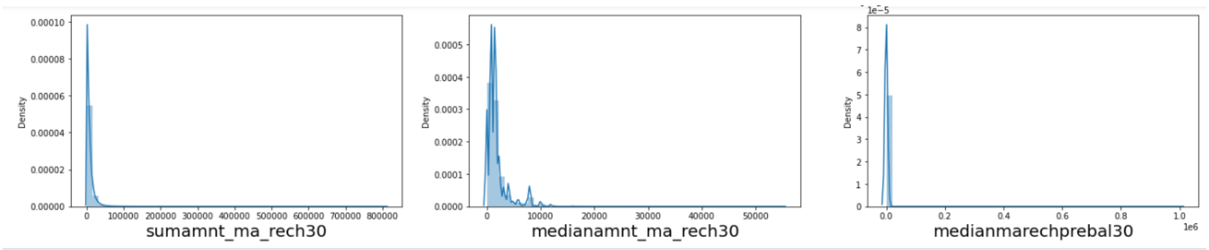
```
# Now checking for numerical columns

num_cols=['aon', 'daily_decr30', 'daily_decr90', 'rental30', 'rental90', 'last_rech_date_ma', 'last_rech_amt_
```

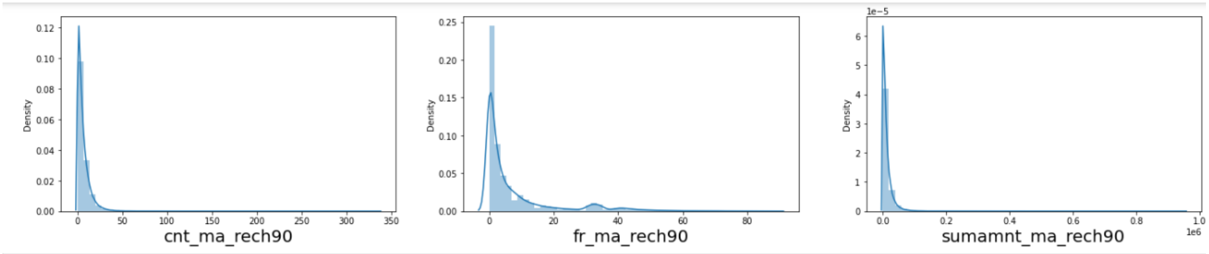
```
#The distribution of data for numerical columns

plt.figure(figsize = (20,40))
plotnumber = 1
for column in df[num_cols]:
    if plotnumber <=30:
        ax = plt.subplot(10,3,plotnumber)
        sns.distplot(df[column])
        plt.xlabel(column,fontsize = 20)
    plotnumber+=1
plt.tight_layout()
```

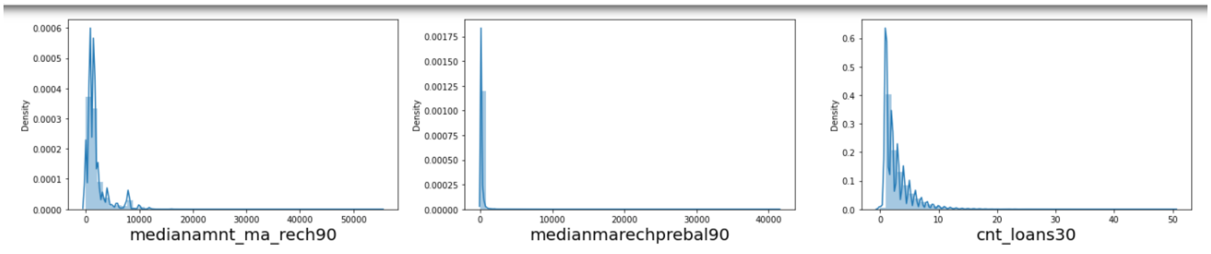

aon
daily_decr30
daily_decr90
Density

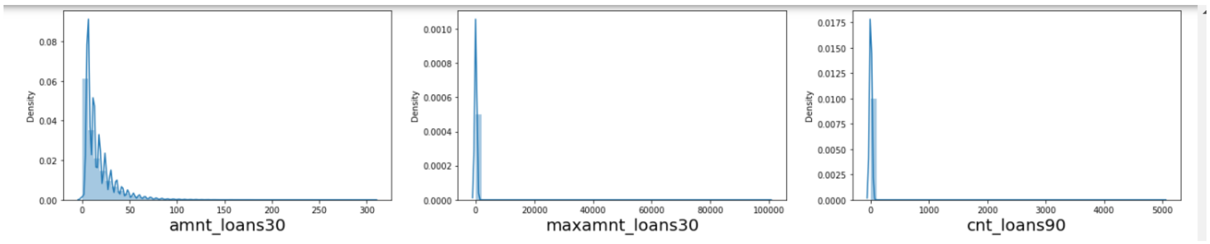

aon
daily_decr30
daily_decr90
Density

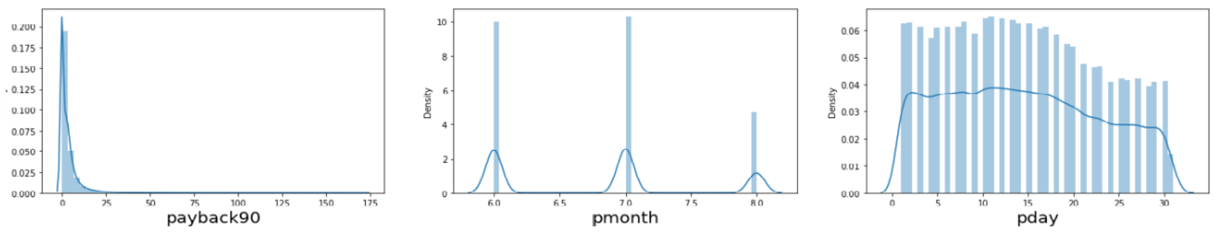

last_rech_amt_ma
cnt_ma_rech30
fr_ma_rech30
Density


sumamnt_ma_rech30
medianamnt_ma_rech30
medianmarechprebal30
Density


cnt_ma_rech90
fr_ma_rech90
sumamnt_ma_rech90
Density


medianamnt_ma_rech90
medianmarechprebal90
cnt_loans30
Density


amnt_loans30
maxamnt_loans30
cnt_loans90
Density

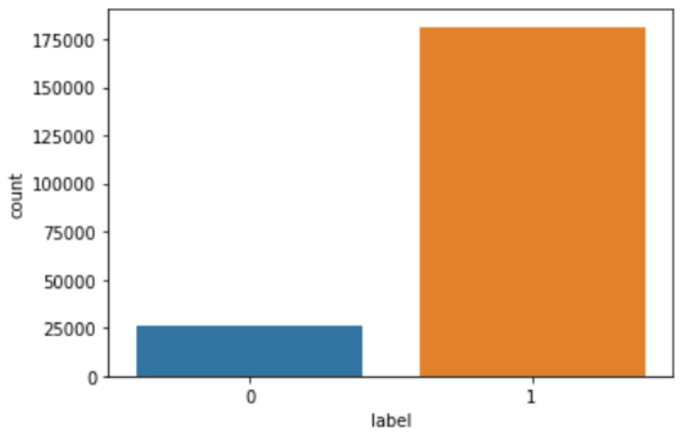


OBSERVATIONS:

Almost all columns are right skewed. We have symmetrical distribution for the pday and pmonth columns.

```
#count plot for target column

sns.countplot(df['label'])
```

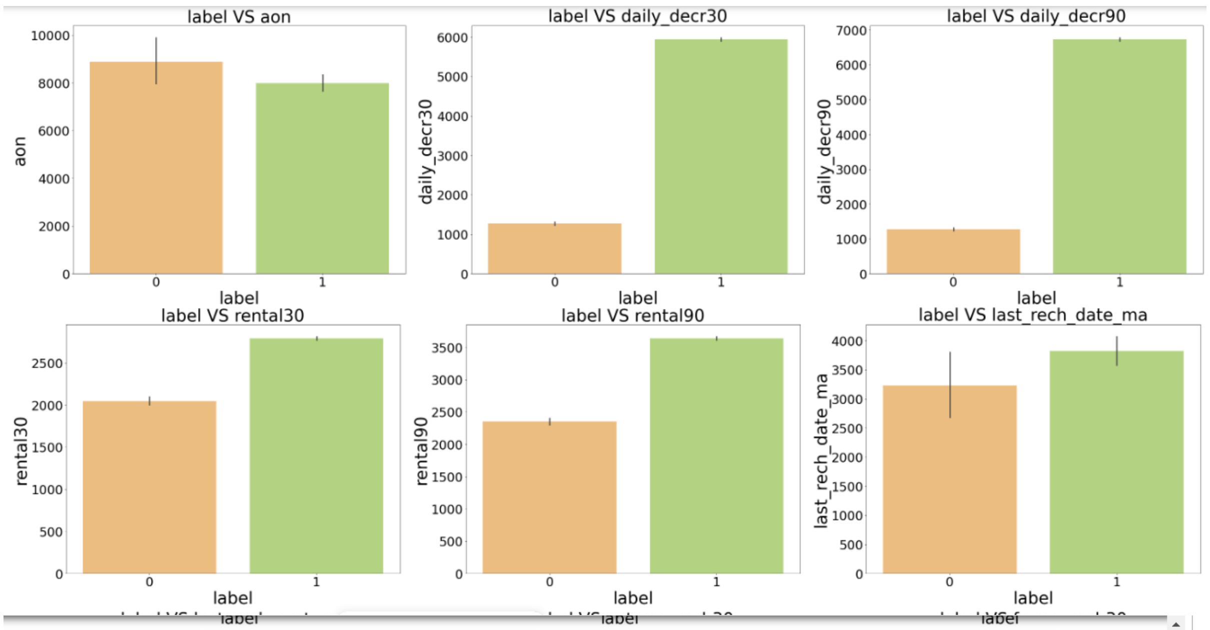
```
<AxesSubplot:xlabel='label', ylabel='count'>
```

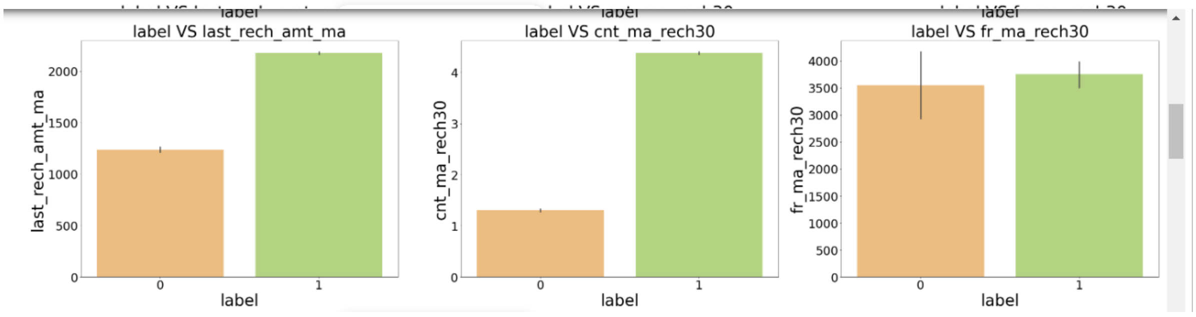


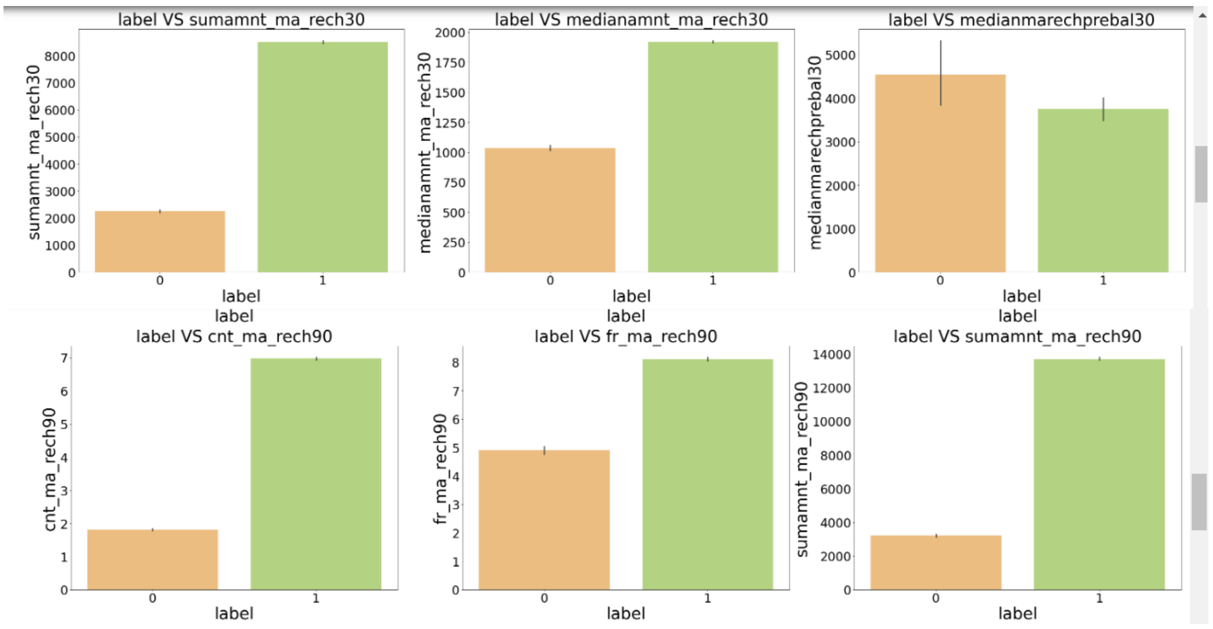
There is a data imbalancing issue so we have to treat this by using SMOTE Technique.

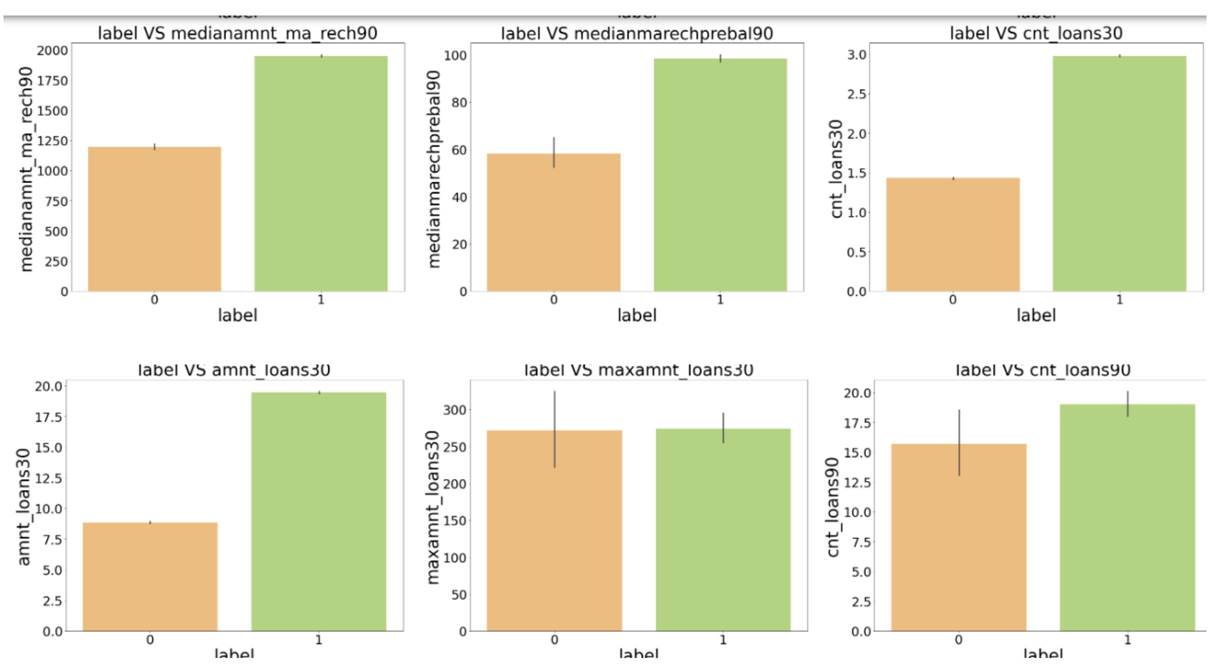
Bivariate Analysis:

```
#barplot for numerical columns

plt.figure(figsize=(40,100))
for i in range(len(num_cols)):
    plt.subplot(10,3,i+1)
    sns.barplot(x=df['label'], y=df[num_cols[i]], palette="RdYlGn")
    plt.title(f"label VS {num_cols[i]}",fontsize=40)
    plt.xticks(fontsize=30)
    plt.yticks(fontsize=30)
    plt.xlabel('label',fontsize = 40)
    plt.ylabel(num_cols[i],fontsize = 40)
    plt.tight_layout()
```


label VS aon
aon
label
label VS daily_decr30
daily_decr30
label
label VS daily_decr90
daily_decr90
label
label VS rental30
rental30
label
label VS rental90
rental90
label
label VS last_rech_date_ma
last_rech_date_ma
label


label VS last_rech_amt_ma
last_rech_amt_ma
label
label VS cnt_ma_rech30
cnt_ma_rech30
label
label VS fr_ma_rech30
fr_ma_rech30
label


label VS sumamnt_ma_rech30
sumamnt_ma_rech30
label
label VS medianamnt_ma_rech30
medianamnt_ma_rech30
label
label VS medianmarechprebal30
medianmarechprebal30
label
label VS cnt_ma_rech90
cnt_ma_rech90
label
label VS fr_ma_rech90
fr_ma_rech90
label
label VS sumamnt_ma_rech90
sumamnt_ma_rech90
label



Z-Score Method

```
#Columns having outliers

out_cols=df[['aon', 'daily_decr30', 'daily_decr90', 'rental30', 'rental90', 'last_rech_date_ma', 'last_rech_amt_ma', 'cnt_ma_rech
```

Above are the list of columns with outliers in the dataset.

```
from scipy.stats import zscore

z=np.abs(zscore(out_cols))
df_new=df[(z<3).all(axis=1)]
df_new
```

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_amt_ma	cnt_ma_rech30	fr_ma_rech30	sumamnt_ma_rech30
0	0	272.0	3055.050000	3065.150000	220.13	260.13	2.0	1539	2	21.0	3078.0
1	1	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	5787	1	0.0	5787.0
2	1	535.0	1398.000000	1398.000000	900.13	900.13	3.0	1539	1	0.0	1539.0
3	1	241.0	21.228000	21.228000	159.42	159.42	41.0	947	0	0.0	0.0
4	1	947.0	150.619333	150.619333	1098.90	1098.90	4.0	2309	7	2.0	20029.0
...	...	...	...	...	...	...	...	...	...	...	...
209588	1	404.0	151.872333	151.872333	1089.19	1089.19	1.0	4048	3	2.0	10404.0
209589	1	1075.0	36.936000	36.936000	1728.36	1728.36	4.0	773	4	1.0	3092.0
209590	1	1013.0	11843.111667	11904.350000	5861.83	8893.20	3.0	1539	5	8.0	9334.0
209591	1	1732.0	12488.228333	12574.370000	411.83	984.58	2.0	773	5	4.0	12154.0
209592	1	1581.0	4489.362000	4534.820000	483.92	631.20	13.0	7526	2	1.0	9065.0

```
print('old data shape:',df.shape)
print('new data shape:',df_new.shape)
```

```
old data shape: (207550, 28)
new data shape: (171073, 28)
```

```
#checking the loss percentage

loss_percentage=(((207550-171073)/207550)*100)
loss_percentage
```

```
17.575042158516023
```

Here, The data loss percentage is too high. As mentioned in the Problem statement we cannot accept this high loss percentage. We will have to try other method of outlier removal

IQR method:

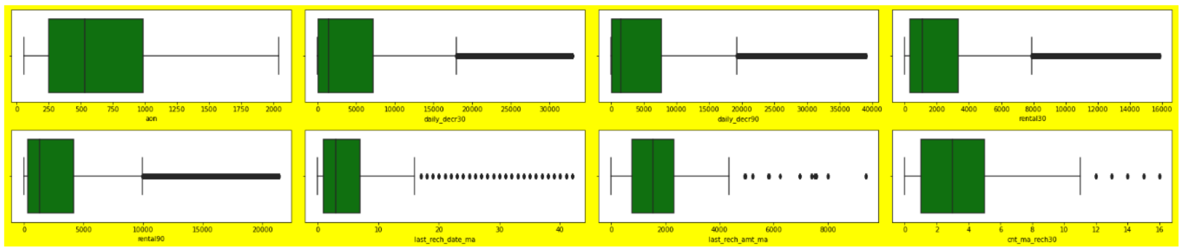
```
# 1st quantile

Q1=out_cols.quantile(0.25)

# 3rd quantile

Q3=out_cols.quantile(0.75)

# IQR

IQR=Q3 - Q1

df_1=df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
```

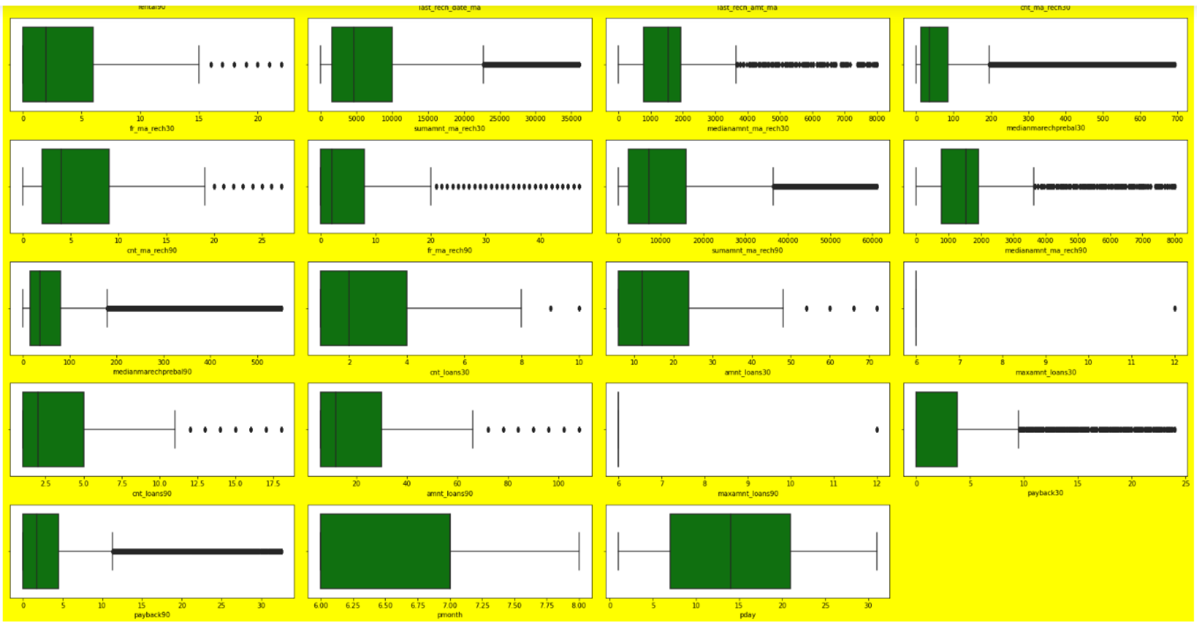
Percentile Method

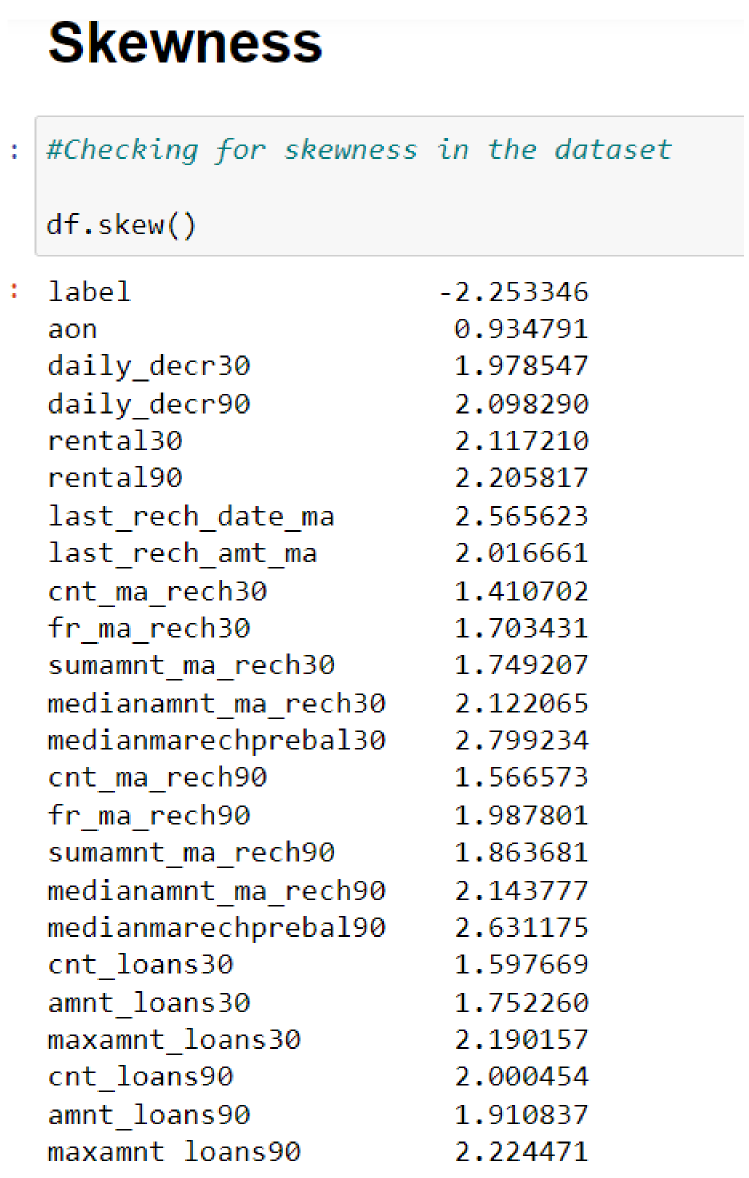
```
#Removing outliers using percentile method

for column in out_cols:
    if df[column].dtypes != 'object':
        percentile = df[column].quantile([0.01,0.98]).values
        df[column][df[column]<=percentile[0]]=percentile[0]
        df[column][df[column]>=percentile[1]]=percentile[1]
```

I have successfully removed outliers in the dataset using percentile method.

```
# Checking if the outliers is reduced or not

plt.figure(figsize=(25,20),facecolor='yellow')
graph=1

for column in num_cols:

    plt.subplot(8,4,graph)

    sns.boxplot(df[column],color='green',orient='h')

    plt.xlabel(column,fontsize=10)

    graph+=1

    plt.tight_layout()
```



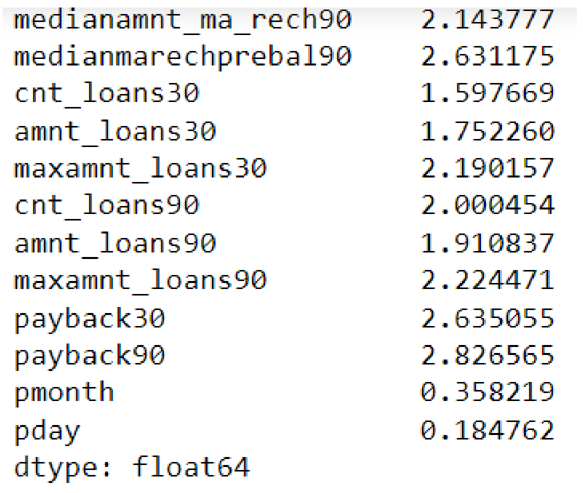




Skewness

```
#Checking for skewness in the dataset

df.skew()
```

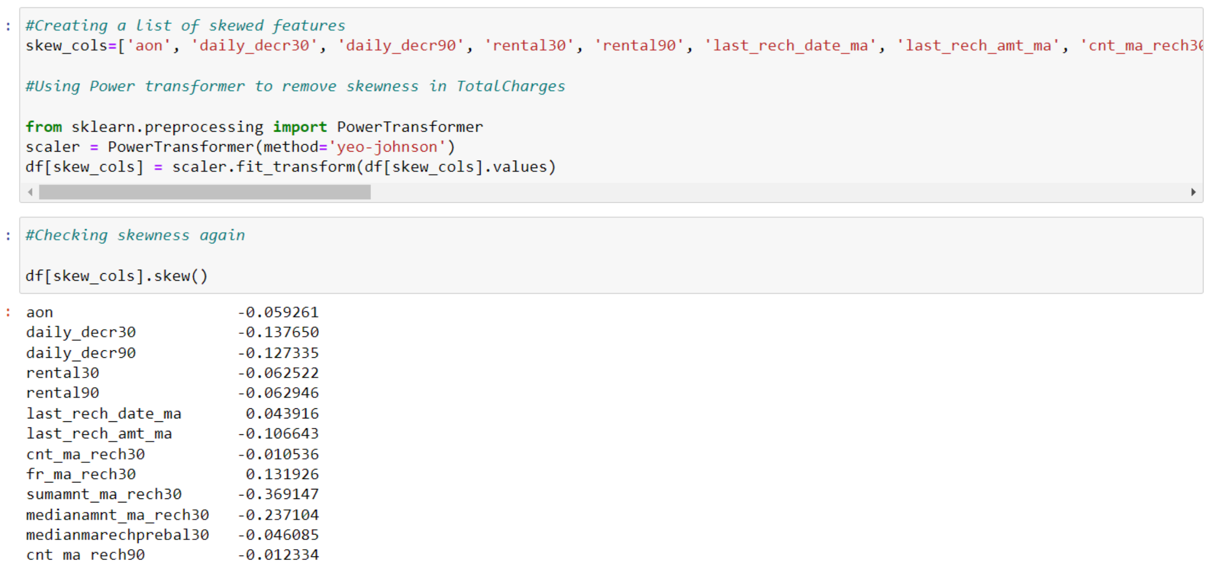
```
label                  -2.253346
aon                     0.934791
daily_decr30            1.978547
daily_decr90            2.098290
rental30                2.117210
rental90                2.205817
last_rech_date_ma       2.565623
last_rech_amt_ma        2.016661
cnt_ma_rech30           1.410702
fr_ma_rech30            1.703431
sumamnt_ma_rech30       1.749207
medianamnt_ma_rech30    2.122065
medianmarechprebal30    2.799234
cnt_ma_rech90           1.566573
fr_ma_rech90            1.987801
sumamnt_ma_rech90       1.863681
medianamnt_ma_rech90    2.143777
medianmarechprebal90    2.631175
cnt_loans30             1.597669
amnt_loans30            1.752260
maxamnt_loans30         2.190157
cnt_loans90             2.000454
amnt_loans90            1.910837
maxamnt_loans90         2.224471
```



```
medianamnt_ma_rech90    2.143777
medianmarechprebal90    2.631175
cnt_loans30             1.597669
amnt_loans30            1.752260
maxamnt_loans30         2.190157
cnt_loans90             2.000454
amnt_loans90            1.910837
maxamnt_loans90         2.224471
```

```
payback30               2.635055
payback90               2.826565
pmonth                  0.358219
pday                    0.184762
dtype: float64
```

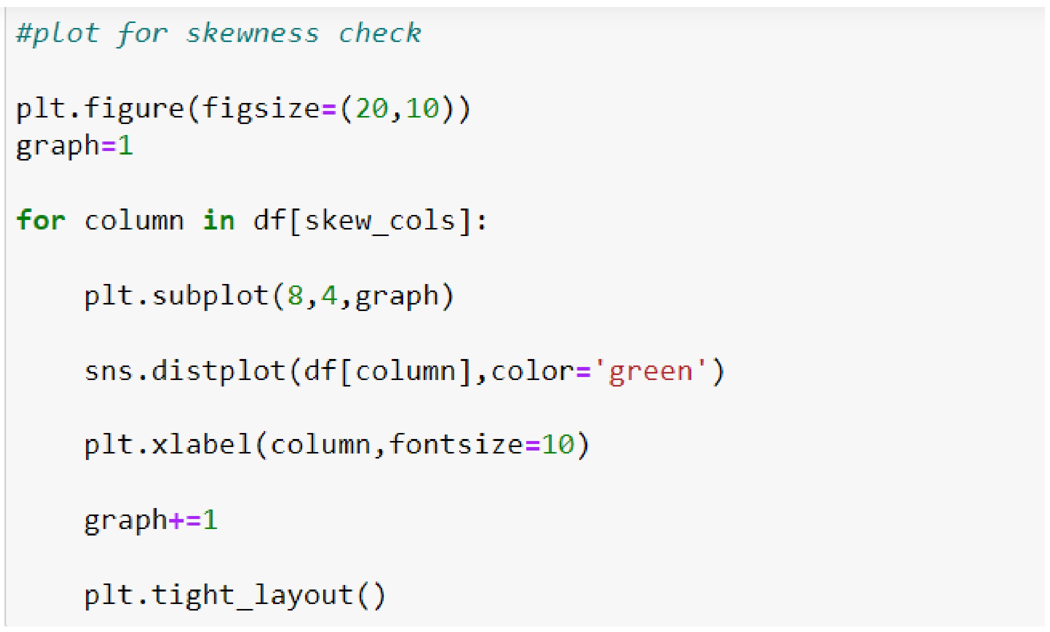
Removing skewness using yeo-johnson method:



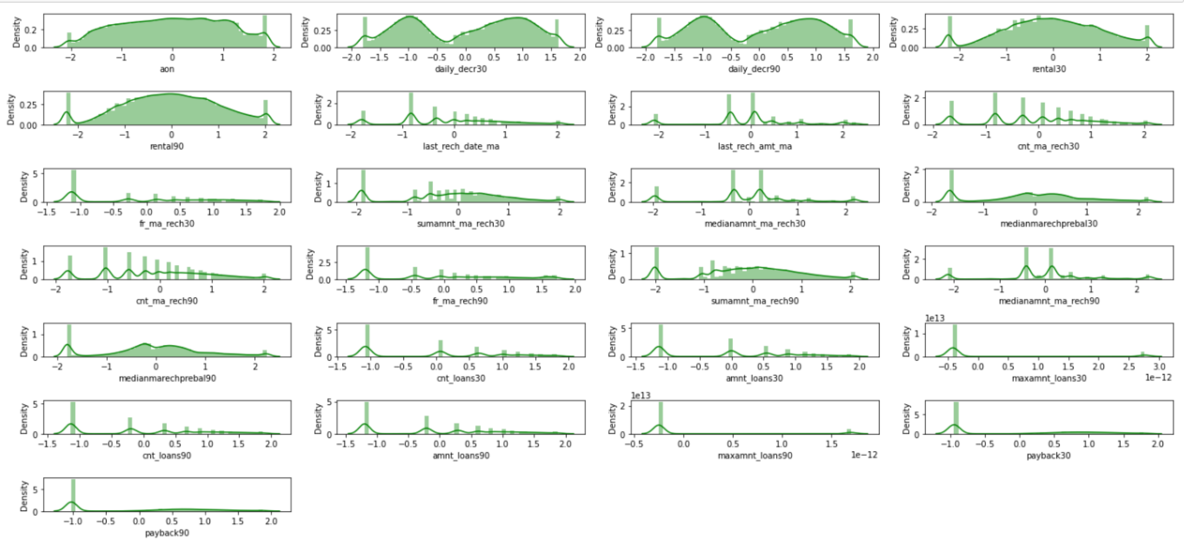
```
#Creating a list of skewed features
skew_cols=['aon', 'daily_decr30', 'daily_decr90', 'rental30', 'rental90', 'last_rech_date_ma', 'last_rech_amt_ma', 'cnt_ma_rech3

#Using Power transformer to remove skewness in TotalCharges

from sklearn.preprocessing import PowerTransformer
scaler = PowerTransformer(method='yeo-johnson')
df[skew_cols] = scaler.fit_transform(df[skew_cols].values)
```

```
#Checking skewness again

df[skew_cols].skew()
```

```
aon                    -0.059261
daily_decr30           -0.137650
daily_decr90           -0.127335
rental30               -0.062522
rental90               -0.062946
last_rech_date_ma       0.043916
last_rech_amt_ma       -0.106643
cnt_ma_rech30          -0.010536
fr_ma_rech30            0.131926
sumamnt_ma_rech30      -0.369147
medianamnt_ma_rech30   -0.237104
medianmarechprebal30   -0.046085
cnt_ma_rech90          -0.012334
```



```
#plot for skewness check

plt.figure(figsize=(20,10))
graph=1

for column in df[skew_cols]:

    plt.subplot(8,4,graph)

    sns.distplot(df[column],color='green')

    plt.xlabel(column,fontsize=10)

    graph+=1

    plt.tight_layout()
```

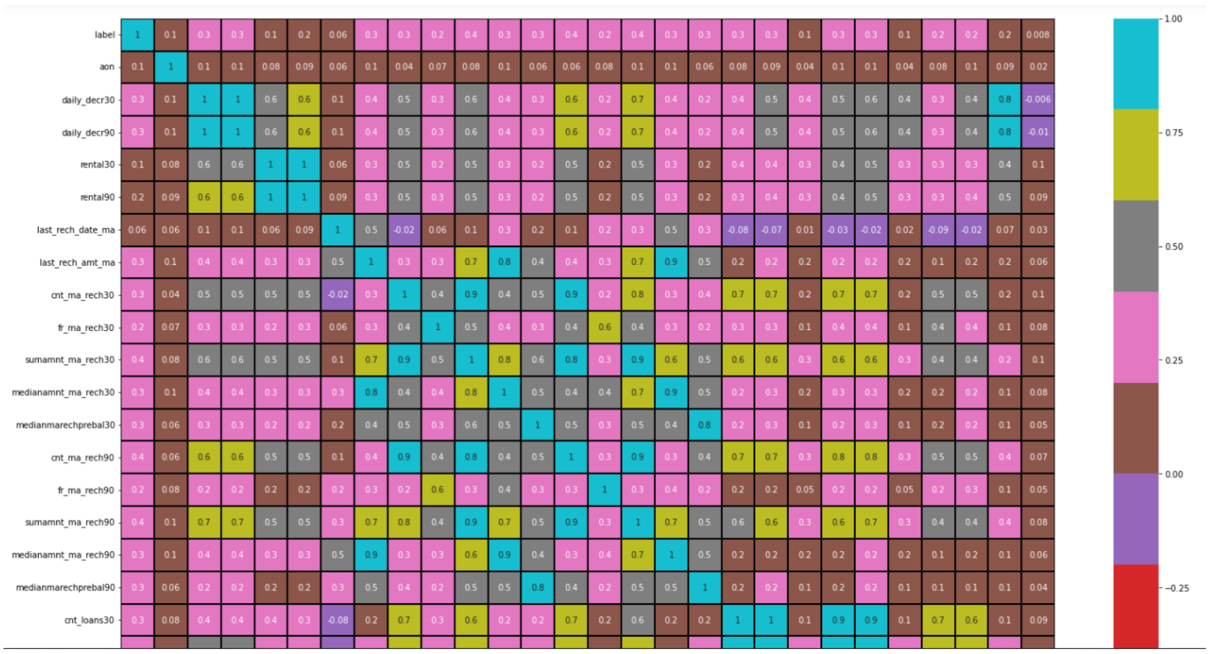


4.3 Correlation

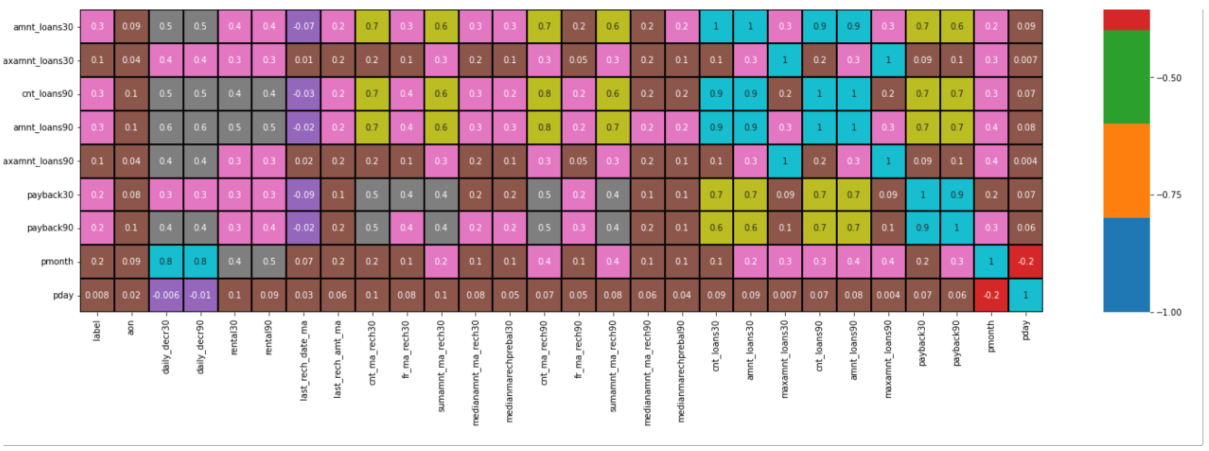
```
#           ck the correlation

(           orr()
corr
```

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_amt_ma	cnt_ma_rech30	fr_ma_rech30
label	1.000000	0.097208	0.266444	0.268775	0.142205	0.155661	0.064305	0.262083	0.347162	0.243564
aon	0.097208	1.000000	0.117096	0.117840	0.084111	0.092045	0.062278	0.097625	0.038139	0.070579
daily_decr30	0.266444	0.117096	1.000000	0.998435	0.585638	0.641529	0.123072	0.375006	0.498882	0.297504
daily_decr90	0.268775	0.117840	0.998435	1.000000	0.586036	0.646414	0.126662	0.374533	0.494780	0.296841
rental30	0.142205	0.084111	0.585638	0.586036	1.000000	0.974866	0.055640	0.284950	0.457289	0.248717
rental90	0.155661	0.092045	0.641529	0.646414	0.974866	1.000000	0.086931	0.294793	0.450131	0.251783
last_rech_date_ma	0.064305	0.062278	0.123072	0.126662	0.055640	0.086931	1.000000	0.518141	-0.016311	0.062349
last_rech_amt_ma	0.262083	0.097625	0.375006	0.374533	0.284950	0.294793	0.518141	1.000000	0.324761	0.274736
cnt_ma_rech30	0.347162	0.038139	0.498882	0.494780	0.457289	0.450131	-0.016311	0.324761	1.000000	0.416578
fr_ma_rech30	0.243564	0.070579	0.297504	0.296841	0.248717	0.251783	0.062349	0.274736	0.416578	1.000000
sumamnt_ma_rech30	0.361711	0.081316	0.556595	0.551778	0.483215	0.473789	0.120058	0.656182	0.857169	0.459433
medianamnt_ma_rech30	0.279565	0.097596	0.388856	0.386170	0.321685	0.314687	0.330315	0.815169	0.414204	0.381396
medianmarechprebal30	0.266752	0.057144	0.274683	0.273615	0.224043	0.221703	0.193080	0.428939	0.483347	0.304847
cnt_ma_rech90	0.363878	0.059595	0.629258	0.632905	0.513308	0.541062	0.097029	0.375053	0.921898	0.409782
fr_ma_rech90	0.221068	0.075794	0.219986	0.221059	0.162761	0.176180	0.220031	0.337799	0.221113	0.616698
sumamnt_ma_rech90	0.370102	0.098734	0.664160	0.667208	0.520826	0.546615	0.250383	0.702025	0.780681	0.422094
medianamnt_ma_rech90	0.252702	0.102863	0.364479	0.364683	0.273283	0.282589	0.507557	0.902231	0.289736	0.294093
medianmarechprebal90	0.257523	0.057758	0.243450	0.244228	0.180335	0.189007	0.338033	0.490820	0.390532	0.225942
cnt_loans30	0.276082	0.081993	0.398639	0.394111	0.351445	0.344863	-0.076541	0.184620	0.732750	0.339864
amnt_loans30	0.291492	0.087453	0.469091	0.465628	0.391547	0.391728	-0.067983	0.221013	0.738661	0.349147
maxamnt_loans30	0.097844	0.036202	0.394698	0.397026	0.270246	0.295450	0.014493	0.181203	0.189267	0.099354
cnt_loans90	0.293548	0.116259	0.530528	0.532191	0.426924	0.447225	-0.027212	0.216948	0.710656	0.351095
amnt_loans90	0.309039	0.118862	0.585352	0.587372	0.457818	0.482003	-0.022283	0.246442	0.723829	0.362987
maxamnt_loans90	0.101247	0.038690	0.406822	0.409956	0.279351	0.306454	0.016073	0.185654	0.194797	0.102128
payback30	0.236554	0.080812	0.336835	0.334597	0.313625	0.311970	-0.094731	0.147133	0.546307	0.415946
payback90	0.244596	0.111218	0.413587	0.414701	0.349548	0.368359	-0.020555	0.184311	0.488019	0.394390
pmonth	0.151680	0.088821	0.819261	0.832069	0.420241	0.505436	0.070518	0.152274	0.199803	0.135763
pday	0.008241	0.016277	-0.005543	-0.012356	0.104843	0.088389	0.032959	0.060133	0.097776	0.081552

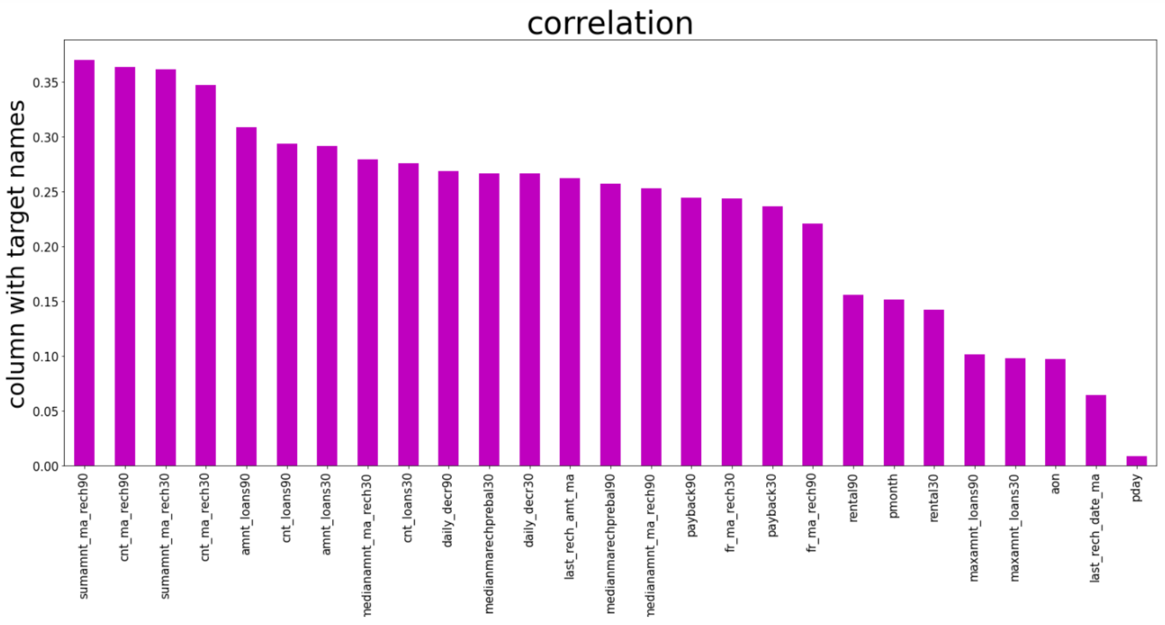
Above are the correlations of all the pair of features To get better visualization on the correlation of features let me plot it using heat map



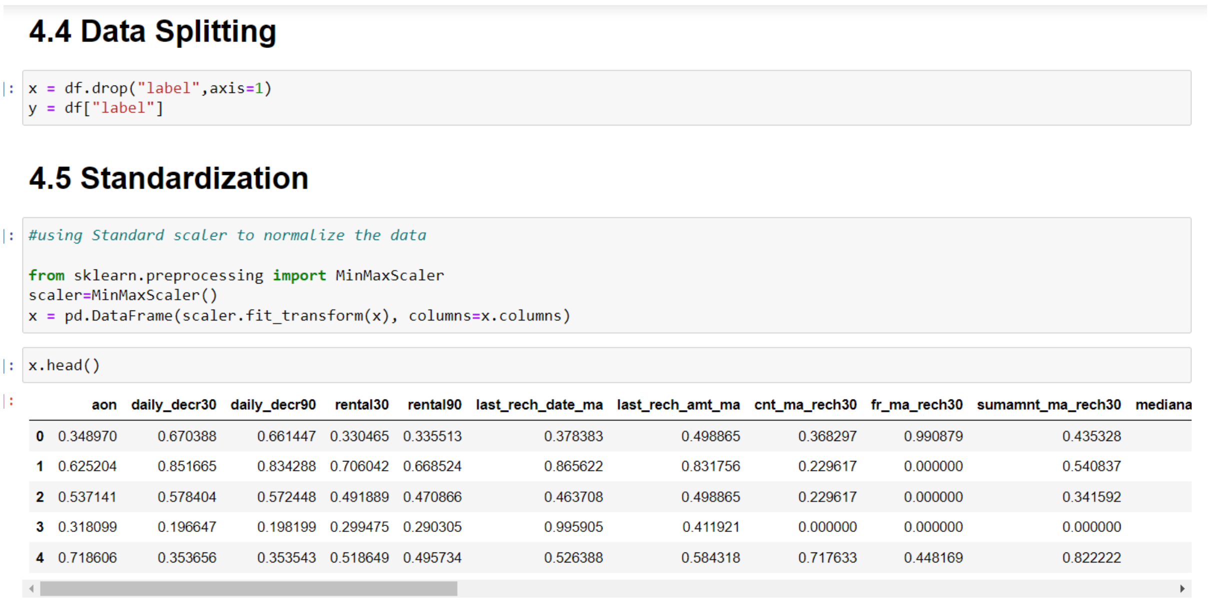
label
aon
daily_decr30
daily_decr90
rental30
rental90
last_rech_date_ma
last_rech_amt_ma
cnt_ma_rech30
fr_ma_rech30
sumamnt_ma_rech30
medianamnt_ma_rech30
medianmarechprebal30
cnt_ma_rech90
fr_ma_rech90
sumamnt_ma_rech90
medianamnt_ma_rech90
medianmarechprebal90
cnt_loans30
1.00
0.75
0.50
0.25
0.00
-0.25



amnt_loans30
maxamnt_loans30
cnt_loans90
amnt_loans90
maxamnt_loans90
payback30
payback90
pmonth
pday
label
aon
daily_decr30
daily_decr90
rental30
rental90
last_rech_date_ma
last_rech_amt_ma
cnt_ma_rech30
fr_ma_rech30
sumamnt_ma_rech30
medianamnt_ma_rech30
medianmarechprebal30
cnt_ma_rech90
fr_ma_rech90
sumamnt_ma_rech90
medianamnt_ma_rech90
medianmarechprebal90
cnt_loans30
amnt_loans30
maxamnt_loans30
cnt_loans90
amnt_loans90
maxamnt_loans90
payback30
payback90
pmonth
pday
-0.50
-0.75
-1.00



correlation
column with target names
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
sumamnt_ma_rech90
cnt_ma_rech90
sumamnt_ma_rech30
cnt_ma_rech30
amnt_loans90
cnt_loans90
amnt_loans30
medianamnt_ma_rech30
cnt_loans30
daily_decr90
medianmarechprebal30
daily_decr30
last_rech_amt_ma
medianmarechprebal90
medianamnt_ma_rech90
payback90
fr_ma_rech30
payback30
fr_ma_rech90
rental90
pmonth
rental30
maxamnt_loans90
maxamnt_loans30
aon
last_rech_date_ma
pday



4.4 Data Splitting

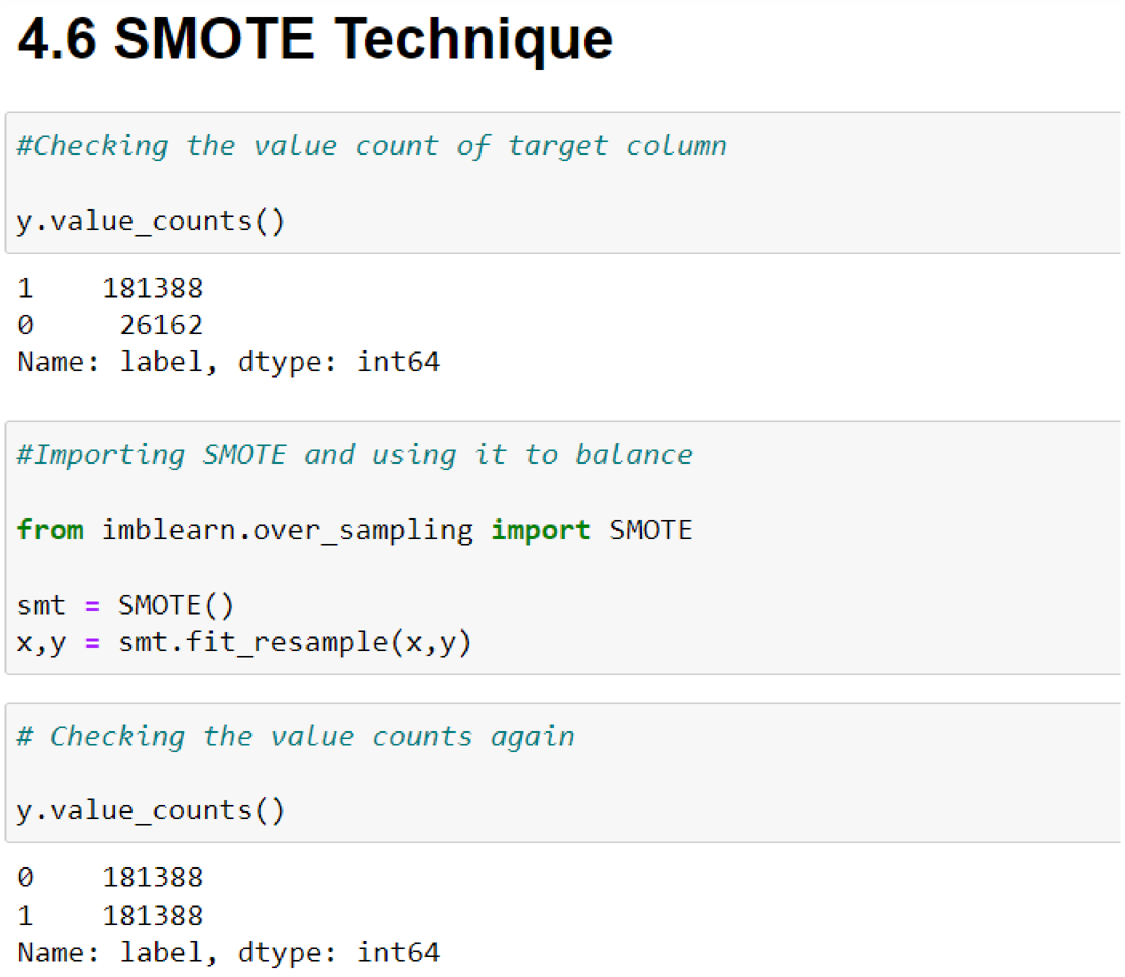
```
x = df.drop("label",axis=1)
y = df["label"]
```

4.5 Standardization

```
#using Standard scaler to normalize the data

from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
x = pd.DataFrame(scaler.fit_transform(x), columns=x.columns)
```

```
x.head()
```

	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_amt_ma	cnt_ma_rech30	fr_ma_rech30	sumamnt_ma_rech30	mediana
0	0.348970	0.670388	0.661447	0.330465	0.335513	0.378383	0.498865	0.368297	0.990879	0.435328	
1	0.625204	0.851665	0.834288	0.706042	0.668524	0.865622	0.831756	0.229617	0.000000	0.540837	
2	0.537141	0.578404	0.572448	0.491889	0.470866	0.463708	0.498865	0.229617	0.000000	0.341592	
3	0.318099	0.196647	0.198199	0.299475	0.290305	0.995905	0.411921	0.000000	0.000000	0.000000	
4	0.718606	0.353656	0.353543	0.518649	0.495734	0.526388	0.584318	0.717633	0.448169	0.822222	



4.6 SMOTE Technique

```
#Checking the value count of target column

y.value_counts()
```

```
1     181388
0      26162
Name: label, dtype: int64
```

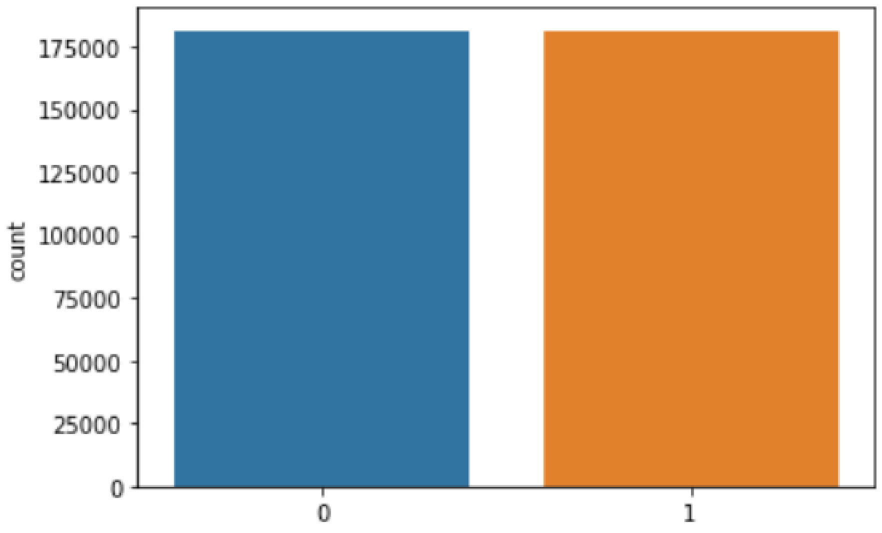
```
#Importing SMOTE and using it to balance

from imblearn.over_sampling import SMOTE

smt = SMOTE()
x,y = smt.fit_resample(x,y)
```

```
# Checking the value counts again

y.value_counts()
```

```
0     181388
1     181388
Name: label, dtype: int64
```

```
# Visualizing the target data after oversampling

sns.countplot(y)
```

```
<AxesSubplot:xlabel='label', ylabel='count'>
```



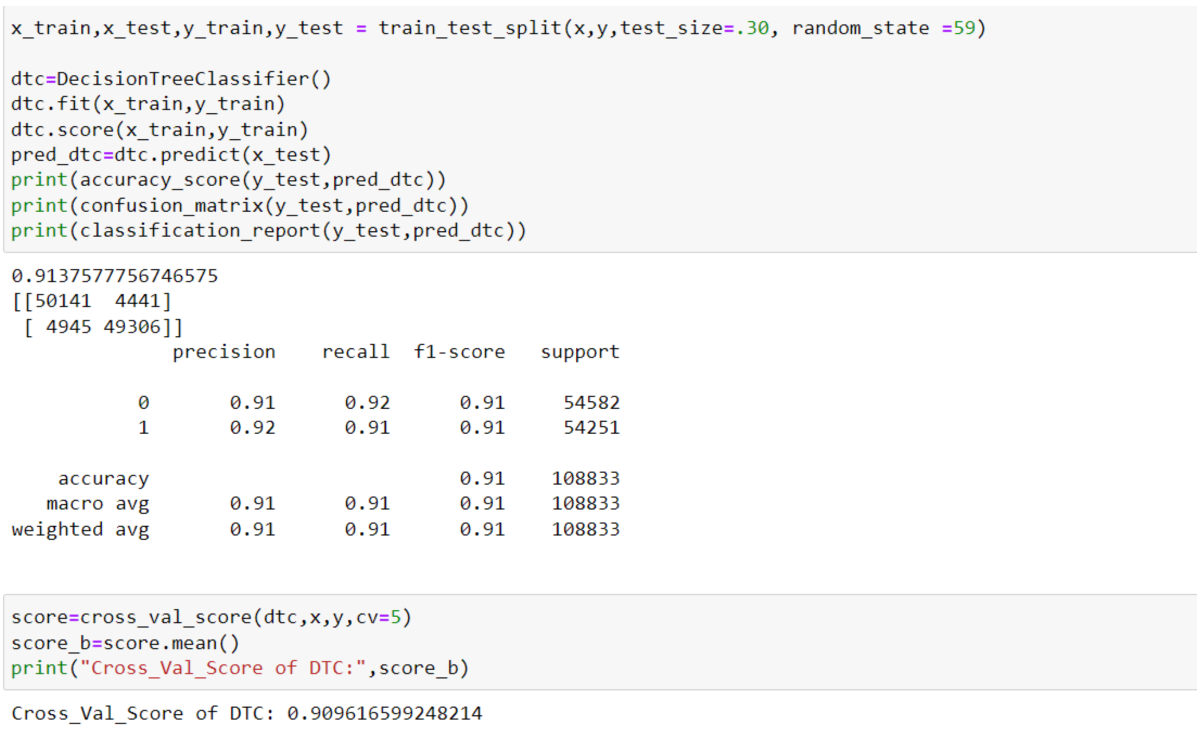
BUILDING THE MODEL

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, roc_curve, roc_auc_score, accuracy_score
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import make_scorer, recall_score, confusion_matrix,precision_score, f1_score, accuracy_score, classificatio
```

1. Decision Tree Classifier

```
#Finding the best Randomstate:

Accu=0
RS=0
for i in range(1,100):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.30, random_state =i)
    dt = DecisionTreeClassifier()
    dt.fit(x_train, y_train)
    pred = dt.predict(x_test)
    acc=accuracy_score(y_test, pred)
    if acc>Accu:
        Accu=acc
        RS=i
print("Best accuracy is ",Accu," on Random_state ",RS)
```

```
Best accuracy is  0.9138037176224123  on Random_state  59
```

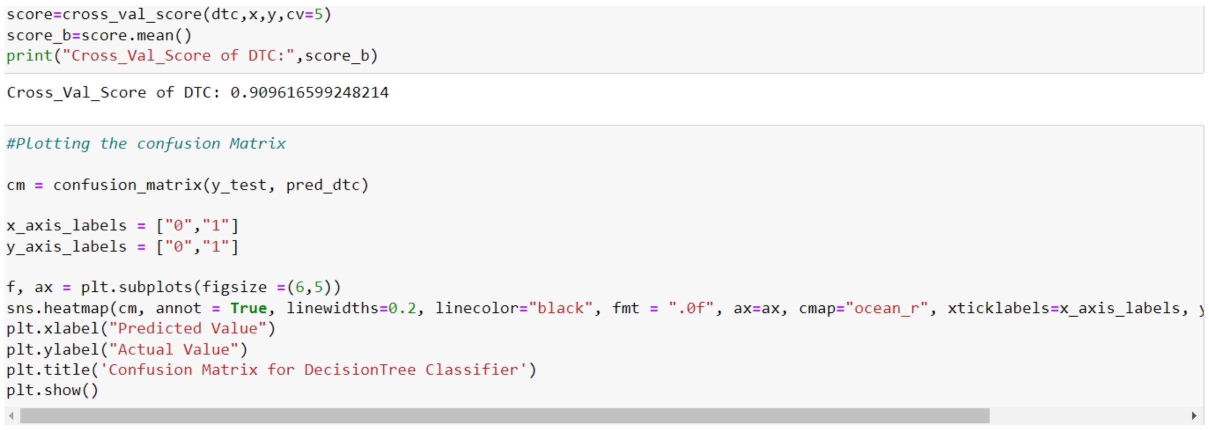


```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.30, random_state =59)

dtc=DecisionTreeClassifier()
dtc.fit(x_train,y_train)
dtc.score(x_train,y_train)
pred_dtc=dtc.predict(x_test)
print(accuracy_score(y_test,pred_dtc))
print(confusion_matrix(y_test,pred_dtc))
print(classification_report(y_test,pred_dtc))
```

```
0.9137577756746575
[[50141  4441]
 [ 4945 49306]]
              precision    recall  f1-score   support

           0       0.91      0.92      0.91     54582
           1       0.92      0.91      0.91     54251

    accuracy                           0.91    108833
   macro avg       0.91      0.91      0.91    108833
weighted avg       0.91      0.91      0.91    108833
```

```
score=cross_val_score(dtc,x,y,cv=5)
score_b=score.mean()
print("Cross_Val_Score of DTC:",score_b)
```

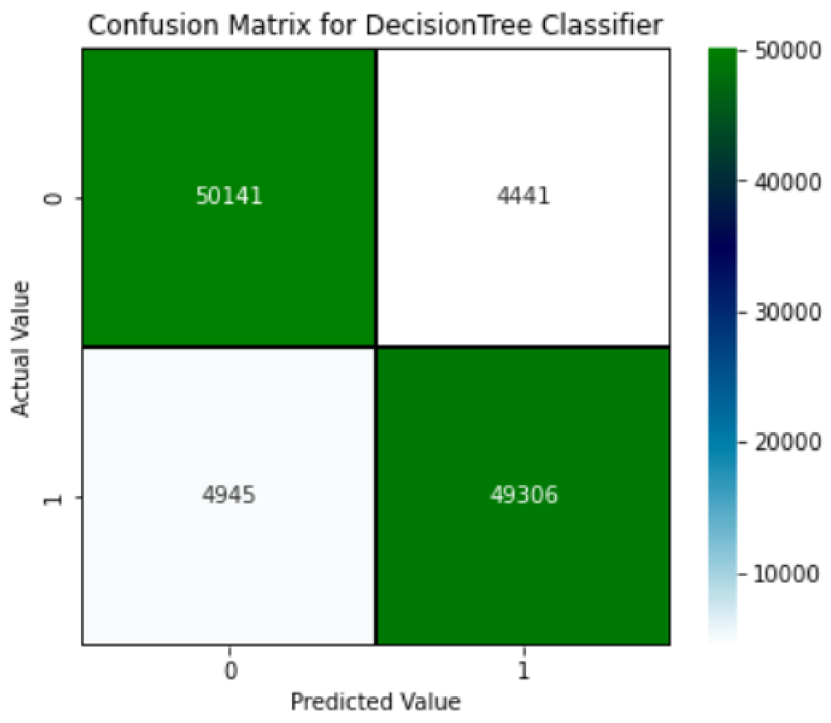
```
Cross_Val_Score of DTC: 0.909616599248214
```



```
score=cross_val_score(dtc,x,y,cv=5)
score_b=score.mean()
print("Cross_Val_Score of DTC:",score_b)
```

```
Cross_Val_Score of DTC: 0.909616599248214
```

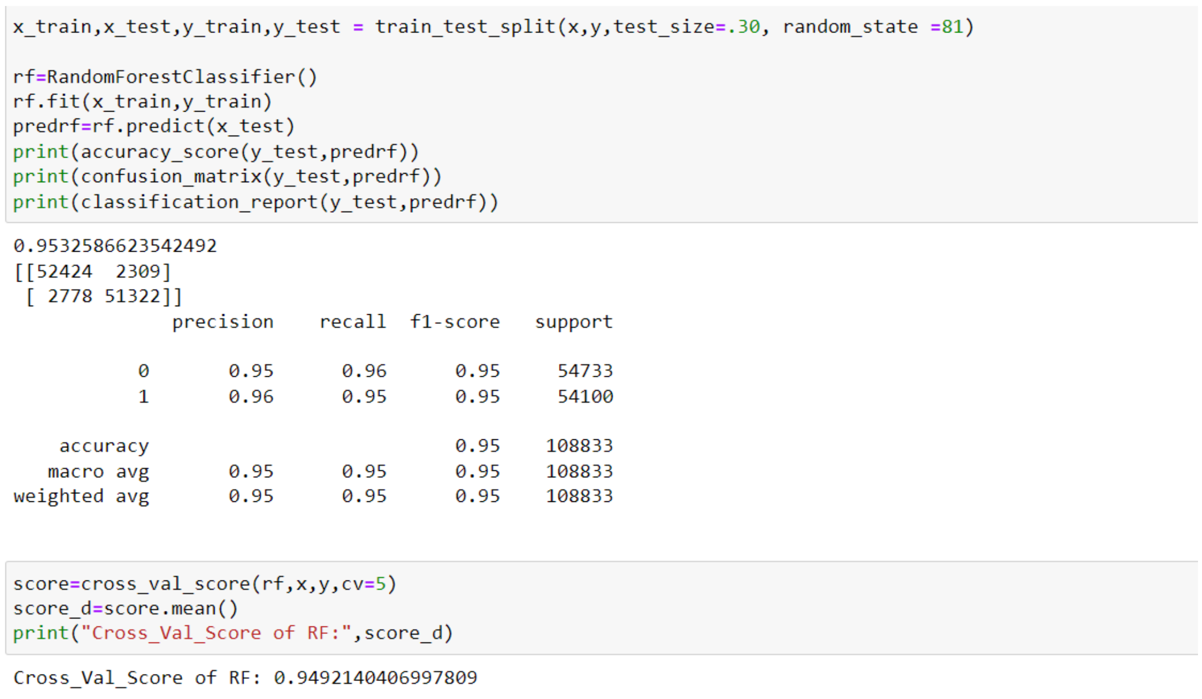
```
#Plotting the confusion Matrix

cm = confusion_matrix(y_test, pred_dtc)

x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f, ax = plt.subplots(figsize =(6,5))
sns.heatmap(cm, annot = True, linewidths=0.2, linecolor="black", fmt = ".0f", ax=ax, cmap="ocean_r", xticklabels=x_axis_labels, y
plt.xlabel("Predicted Value")
plt.ylabel("Actual Value")
plt.title('Confusion Matrix for DecisionTree Classifier')
plt.show()
```



2. Random Forest Classifier

```
#Finding the best random state

Accu=0
RS=0
for i in range(1,100):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.30, random_state =i)
    rf=RandomForestClassifier()
    rf.fit(x_train, y_train)
    pred = rf.predict(x_test)
    acc=accuracy_score(y_test, pred)
    if acc>Accu:
        Accu=acc
        RS=i
print("Best accuracy is ",Accu," on Random_state ",RS)
```

```
Best accuracy is  0.9533137926915549  on Random_state  81
```

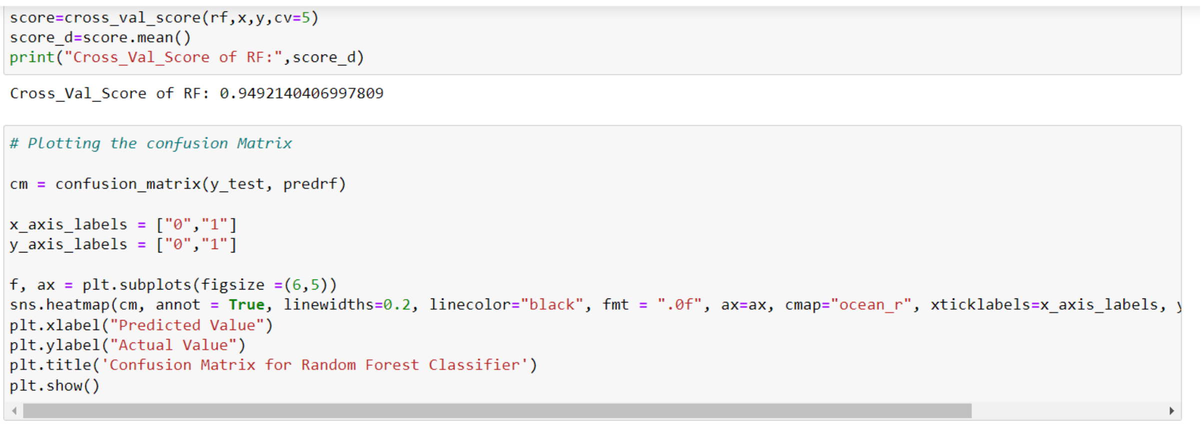


```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.30, random_state =81)

rf=RandomForestClassifier()
rf.fit(x_train,y_train)
predrf=rf.predict(x_test)
print(accuracy_score(y_test,predrf))
print(confusion_matrix(y_test,predrf))
print(classification_report(y_test,predrf))
```

```
0.9532586623542492
[[52424  2309]
 [ 2778 51322]]
              precision    recall  f1-score   support

           0       0.95      0.96      0.95     54733
           1       0.96      0.95      0.95     54100

    accuracy                           0.95    108833
   macro avg       0.95      0.95      0.95    108833
weighted avg       0.95      0.95      0.95    108833
```

```
score=cross_val_score(rf,x,y,cv=5)
score_d=score.mean()
print("Cross_Val_Score of RF:",score_d)
```

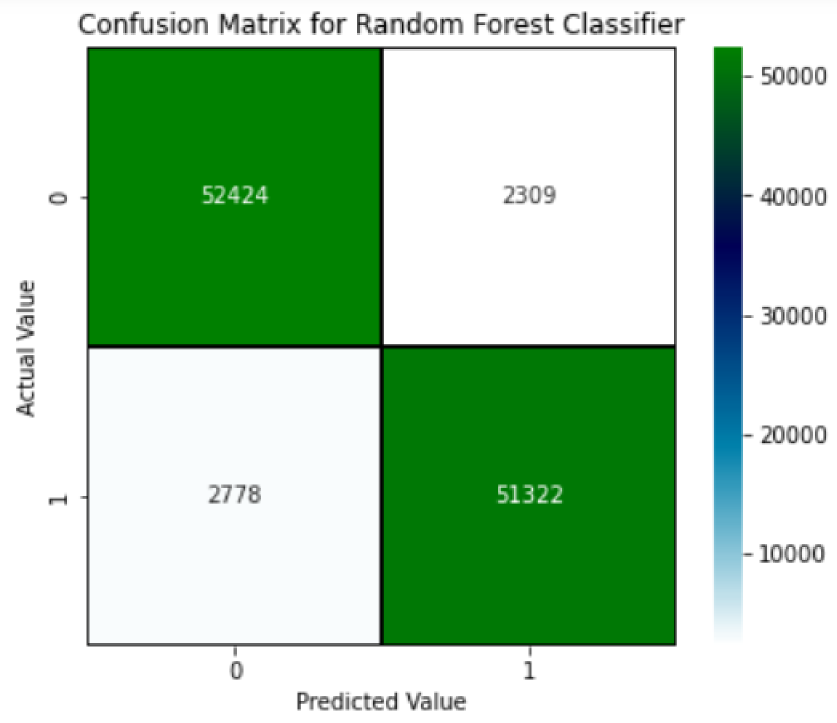
```
Cross_Val_Score of RF: 0.9492140406997809
```



```
score=cross_val_score(rf,x,y,cv=5)
score_d=score.mean()
print("Cross_Val_Score of RF:",score_d)
```

```
Cross_Val_Score of RF: 0.9492140406997809
```

```
# Plotting the confusion Matrix

cm = confusion_matrix(y_test, predrf)

x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f, ax = plt.subplots(figsize =(6,5))
sns.heatmap(cm, annot = True, linewidths=0.2, linecolor="black", fmt = ".0f", ax=ax, cmap="ocean_r", xticklabels=x_axis_labels, y
plt.xlabel("Predicted Value")
plt.ylabel("Actual Value")
plt.title('Confusion Matrix for Random Forest Classifier')
plt.show()
```



3. KNN Classifier

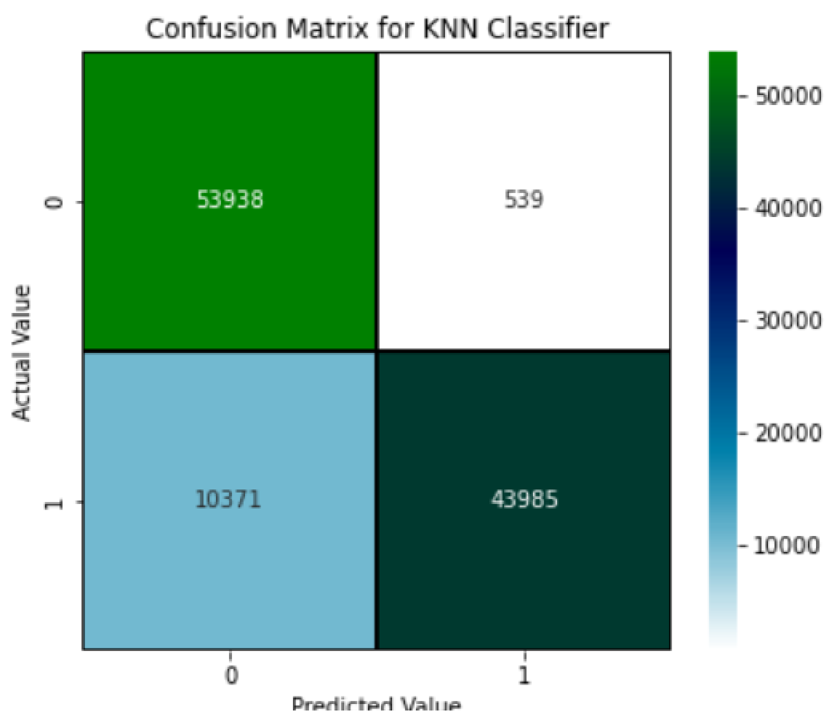
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.30, random_state =65)

knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
knn.score(x_train,y_train)
pred_knn=knn.predict(x_test)
print(accuracy_score(y_test,pred_knn))
print(confusion_matrix(y_test,pred_knn))
print(classification_report(y_test,pred_knn))
```

```
0.8997546699989892
[[53938   539]
 [10371 43985]]
              precision    recall  f1-score   support

           0       0.84      0.99      0.91     54477
           1       0.99      0.81      0.89     54356

    accuracy                           0.90    108833
   macro avg       0.91      0.90      0.90    108833
weighted avg       0.91      0.90      0.90    108833
```

```
score=cross_val_score(knn,x,y,cv=5)
score_e=score.mean()
print("Cross_Val_Score of KNN:",score_e)
```

```
Cross_Val_Score of KNN: 0.9046078070640867
```

```
#Plotting the confusion Matrix

cm = confusion_matrix(y_test, pred_knn)

x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f, ax = plt.subplots(figsize =(6,5))
sns.heatmap(cm, annot = True, linewidths=0.2, linecolor="black", fmt = ".0f", ax=ax, cmap="ocean_r", xticklabels=x_axis_labels, y
plt.xlabel("Predicted Value")
plt.ylabel("Actual Value")
plt.title('Confusion Matrix for KNN Classifier')
plt.show()
```



4. XGB Classifier

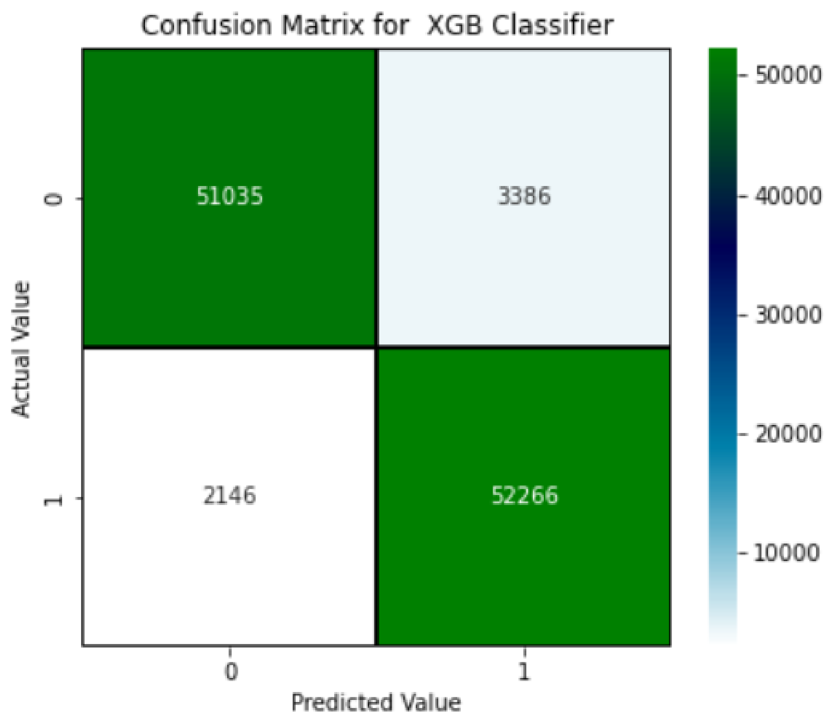
```
from xgboost import XGBClassifier

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.30, random_state =41)

# Checking accuracy for XGBClassifier
XGB=XGBClassifier(verbosity=0)
XGB.fit(x_train,y_train)

# Prediction
predxgb = XGB.predict(x_test)

print(accuracy_score(y_test, predxgb))
print(confusion_matrix(y_test, predxgb))
print(classification_report(y_test,predxgb))
```

```
0.9491698290040704
[[51035  3386]
 [ 2146 52266]]
              precision    recall  f1-score   support

           0       0.96      0.94      0.95     54421
           1       0.94      0.96      0.95     54412

    accuracy                           0.95    108833
   macro avg       0.95      0.95      0.95    108833
weighted avg       0.95      0.95      0.95    108833
```

```
score=cross_val_score(XGB,x,y,cv=5)
score_g=score.mean()
print("Cross_Val_Score of XGB Classifier:",score_g)
```

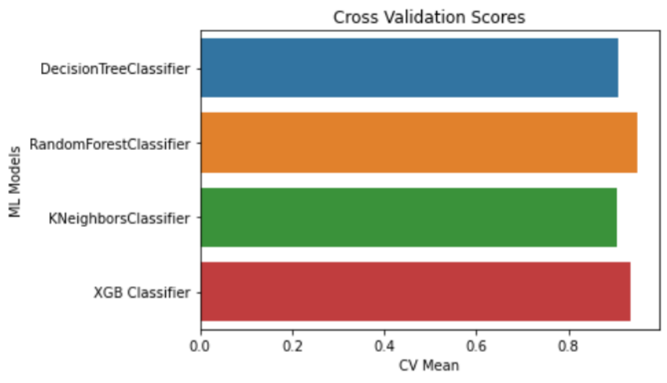
```
Cross_Val_Score of XGB Classifier: 0.935638391567679
```

```
# Lets plot confusion matrix for  XGBClassifier
cm = confusion_matrix(y_test,predxgb)

x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f, ax = plt.subplots(figsize =(6,5))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="ocean_r",xticklabels=x_axis_labels,ytick

plt.xlabel("Predicted Value")
plt.ylabel("Actual Value")
plt.title('Confusion Matrix for  XGB Classifier')
plt.show()
```



```
cv_result=[score_b,score_d,score_e,score_g]
cv_results = pd.DataFrame({"Cross Validation Means":cv_result, "ML Models":["DecisionTreeClassifier",
                                                                              "RandomForestClassifier",
                                                                              "KNeighborsClassifier",
                                                                              "XGB Classifier"]})

g = sns.barplot("Cross Validation Means", "ML Models", data = cv_results)
g.set_xlabel("CV Mean")
g.set_title("Cross Validation Scores")
```

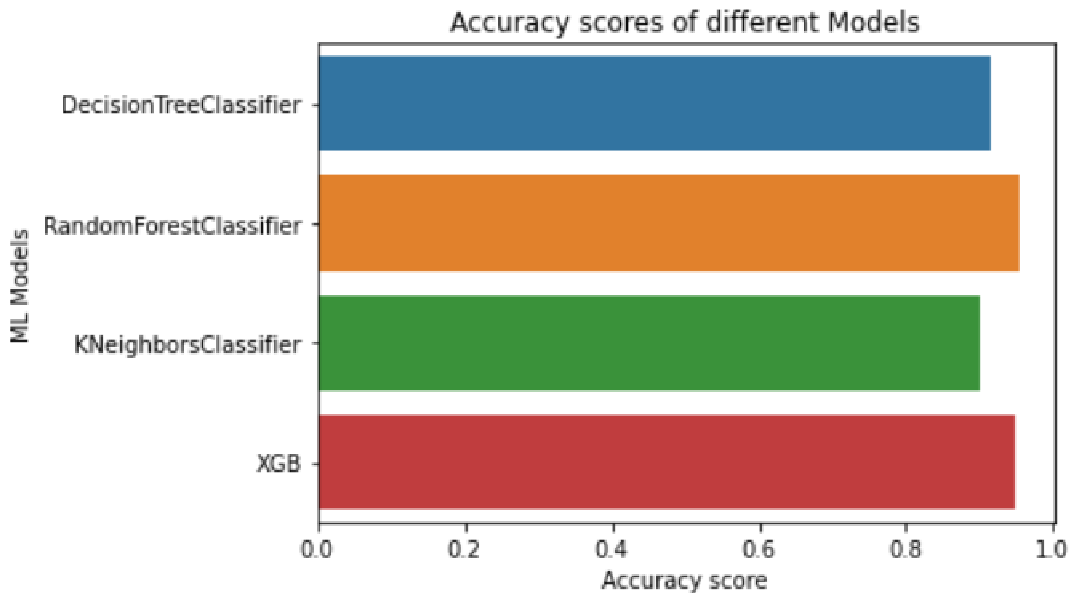
```
Text(0.5, 1.0, 'Cross Validation Scores')
```



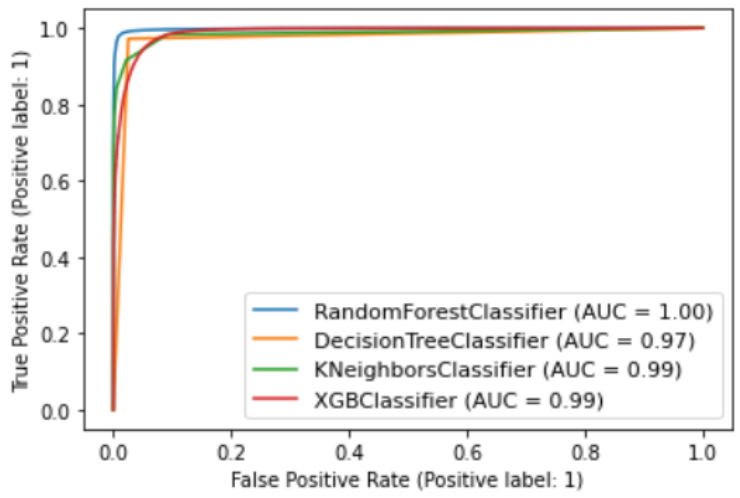
Random forest is the best

```
score_dtc=0.9155587000266463
score_rf=0.9531667784587395
score_knn=0.8999660029586615
score_xgb=0.9495006110279052

acc_result=[score_dtc,score_rf,score_knn,score_xgb]
acc_results = pd.DataFrame({"Accuracy Scores":acc_result, "ML Models":[ "DecisionTreeClassifier",
                                                                          "RandomForestClassifier",
                                                                          "KNeighborsClassifier","XGB"]})

g = sns.barplot("Accuracy Scores", "ML Models", data = acc_results)
g.set_xlabel("Accuracy score")
g.set_title("Accuracy scores of different Models")
```

```
Text(0.5, 1.0, 'Accuracy scores of different Models')
```



```python
# Plotting AUC-ROC Curve for all the models used here

from sklearn.metrics import plot_roc_curve

disp = plot_roc_curve(rf,x_test,y_test)
plot_roc_curve(dtc,x_test, y_test, ax=disp.ax_)
plot_roc_curve(knn, x_test, y_test, ax=disp.ax_)
plot_roc_curve(XGB, x_test, y_test, ax=disp.ax_)

plt.legend(prop={'size':11}, loc='lower right')
plt.show()
```



Final Model

```python
#Final model as Random Forest
#{'bootstrap': False, 'criterion': 'gini', 'max_features': 1, 'min_samples_leaf': 1, 'min_samples_split': 3, 'n_estimators': 300

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.30, random_state =81)

Final_Model=RandomForestClassifier(n_estimators=300,bootstrap=False,criterion='gini',
                          max_features=1,min_samples_leaf=1,min_samples_split=3)
Final_Model.fit(x_train,y_train)
predrf=Final_Model.predict(x_test)
print(accuracy_score(y_test,predrf))
print(confusion_matrix(y_test,predrf))
print(classification_report(y_test,predrf))
```

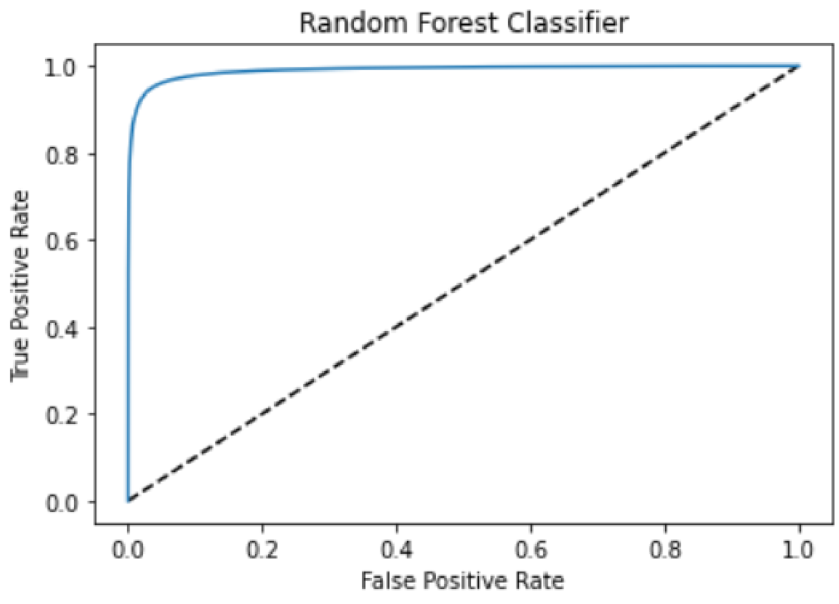
```
0.9562816425165162
[[52261  2472]
 [ 2286 51814]]
              precision    recall  f1-score   support

           0       0.96      0.95      0.96     54733
           1       0.95      0.96      0.96     54100

    accuracy                           0.96    108833
   macro avg       0.96      0.96      0.96    108833
weighted avg       0.96      0.96      0.96    108833
```

AUC_ROC Curve

```
: from sklearn.metrics import roc_curve
  from sklearn.metrics import roc_auc_score

  y_pred_prob=Final_Model.predict_proba(x_test)[:,1]
  y_pred_prob
```

```
: array([0.86333333, 0.97666667, 0.12333333, ..., 0.00333333, 0.905     ,
         0.01833333])
```

```
plt.plot([0,1],[0,1],'k--')
plt.plot(fpr,tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Random Forest Classifier')
plt.show()
```



```
auc_score=roc_auc_score(y_test,Final_Model.predict(x_test))
print((auc_score)*100)
```

```
95.62901040720178
```

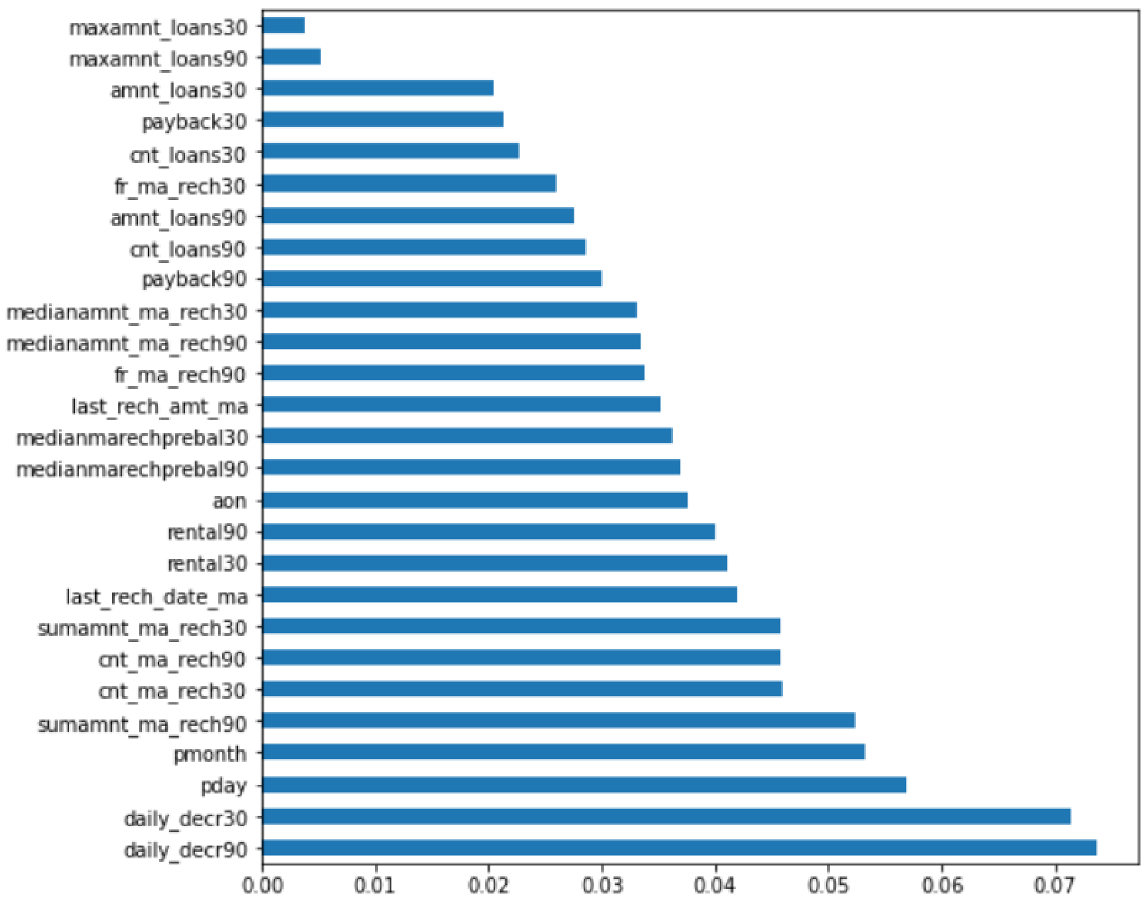
Saving the Model

```
: import joblib
  joblib.dump(Final_Model,'Micro Credit RF Model.pkl')
```

```
: ['Micro Credit RF Model.pkl']
```

```
: #Checking Feature Importance with Random Forest Model

  feat_importances = pd.Series(Final_Model.feature_importances_, index=x.columns)
  feat_importances.nlargest(100).plot(kind='barh',figsize=(8,8))
```

```
: <AxesSubplot:>
```



Predictions

```
# Loading the saved model
model=joblib.load("Micro Credit RF Model.pkl")

#Prediction
prediction = model.predict(x_test)
prediction
```

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

```
#saving as dataframe

base = pd.DataFrame()
base["actual"] = y_test
base["predictions"] = prediction
base
```

	actual	predictions
163460	1	1
8928	1	1
248038	0	0
137081	0	0
310285	0	0
...	...	...
245188	0	0
168885	1	1
219511	0	0
189838	1	1
350006	0	0

108833 rows × 2 columns

```
#Adding another column of thier difference.

base['difference']=base['actual']-base['predictions']

#If 0 then actual and predicted are same. else its different

print(base['difference'].value_counts())
```

```
 0    104075
-1      2472
 1      2286
Name: difference, dtype: int64
```

```
a=(4790/104043)*100
a
```

```
4.603865709370164
```

As per our Model, only 4% of the result is wrong and rest all are true values.