FLIP ROBO

# Micro Credit Project

**Submitted by:**

**Surjit Singh**

# ACKNOWLEDGMENT

- First of all, I would like to thank all my mentors in Data Trained and FlipRobo Technologies for this opportunity.
- Most of the concepts used to predict the Micro Credit are learned from Data Trained Institute.
- Here I would be thankful that I got this chance to do the project, this gave me good knowledge about the data collection and model building ie., prediction of the data.

# 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.
- 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.
- They understand the importance of communication and how it affects a person's life, thus, focusing on providing their services and products to low-income families and poor customers that can help them in the need of hour.
- They are collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed to be defaulter if he deviates from the path of paying back the loaned amount within the time duration of 5 days. For the loan amount of 5 (in Indonesian Rupiah), payback amount should be 6 (in Indonesian Rupiah), while, for the loan amount of 10 (in Indonesian Rupiah), the payback amount should be 12 (in Indonesian Rupiah)

# Analytical Problem Framing

- ❖ Here our dataset has 209593 rows and 37 columns, using this dataset we will be building the model followed by training the data and then finally the model is tested by using 67% of the training data and 33% of the testing data.
- ❖ Since we have no null values from the dataset during the data collection stage, we can expect outliers and un-realistic values for certain variables.

| | A | B | C |
|---|---|---|---|
| 1 | Variable | Definition | Comment |
| 2 | label | Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure} | |
| 3 | msisdn | mobile number of user | |
| 4 | aon | age on cellular network in days | |
| 5 | daily_decr30 | Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah) | |
| 6 | daily_decr90 | Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah) | |
| 7 | rental30 | Average main account balance over last 30 days | Unsure of given definition |
| 8 | rental90 | Average main account balance over last 90 days | Unsure of given definition |
| 9 | last_rech_date_ma | Number of days till last recharge of main account | |
| 10 | last_rech_date_da | Number of days till last recharge of data account | |
| 11 | last_rech_amt_ma | Amount of last recharge of main account (in Indonesian Rupiah) | |
| 12 | cnt_ma_rech30 | Number of times main account got recharged in last 30 days | |
| 13 | fr_ma_rech30 | Frequency of main account recharged in last 30 days | Unsure of given definition |
| 14 | sumamnt_ma_rech30 | Total amount of recharge in main account over last 30 days (in Indonesian Rupiah) | |
| 15 | medianamnt_ma_rech30 | Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah) | |
| 16 | medianmarechprebal30 | Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah) | |
| 17 | cnt_ma_rech90 | Number of times main account got recharged in last 90 days | |
| 18 | fr_ma_rech90 | Frequency of main account recharged in last 90 days | Unsure of given definition |
| 19 | sumamnt_ma_rech90 | Total amount of recharge in main account over last 90 days (in Indonasian Rupiah) | |
| 20 | medianamnt_ma_rech90 | Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah) | |
| 21 | medianmarechprebal90 | Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah) | |
| 22 | cnt_da_rech30 | Number of times data account got recharged in last 30 days | |
| 23 | fr_da_rech30 | Frequency of data account recharged in last 30 days | |
| 24 | cnt_da_rech90 | Number of times data account got recharged in last 90 days | |
| 25 | fr_da_rech90 | Frequency of data account recharged in last 90 days | |
| 26 | cnt_loans30 | Number of loans taken by user in last 30 days | |
| 27 | amnt_loans30 | Total amount of loans taken by user in last 30 days | |
| 28 | maxamnt_loans30 | maximum amount of loan taken by the user in last 30 days | There are only two options: 5 & 10 Rs., for which the user needs to pay back 6 & 12 Rs. respectively |
| 29 | medianamnt_loans30 | Median of amounts of loan taken by the user in last 30 days | |
| 30 | cnt_loans90 | Number of loans taken by user in last 90 days | |
| 31 | amnt_loans90 | Total amount of loans taken by user in last 90 days | |
| 32 | maxamnt_loans90 | maximum amount of loan taken by the user in last 90 days | |
| 33 | medianamnt_loans90 | Median of amounts of loan taken by the user in last 90 days | |
| 34 | payback30 | Average payback time in days over last 30 days | |
| 35 | payback90 | Average payback time in days over last 90 days | |
| 36 | pcircle | telecom circle | |
| 37 | pdate | date | |

# Analysis

## Importing the Required libraries :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

## Data Collection:

```
In [2]: data = pd.read_csv("Micro credit project.csv")
        data
```

Out[2]:

| | Unnamed: 0 | label | msisdn | aon | daily_decr30 | daily_decr90 | rental30 | rental90 | last_rech_date_ma | last_rech_date_da | ... | maxamnt_loans30 | mec |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 21408I70789 | 272.0 | 3055.050000 | 3065.150000 | 220.13 | 260.13 | 2.0 | 0.0 | ... | 6.0 | |
| **1** | 2 | 1 | 76462I70374 | 712.0 | 12122.000000 | 12124.750000 | 3691.26 | 3691.26 | 20.0 | 0.0 | ... | 12.0 | |
| **2** | 3 | 1 | 17943I70372 | 535.0 | 1398.000000 | 1398.000000 | 900.13 | 900.13 | 3.0 | 0.0 | ... | 6.0 | |
| **3** | 4 | 1 | 55773I70781 | 241.0 | 21.228000 | 21.228000 | 159.42 | 159.42 | 41.0 | 0.0 | ... | 6.0 | |
| **4** | 5 | 1 | 03813I82730 | 947.0 | 150.619333 | 150.619333 | 1098.90 | 1098.90 | 4.0 | 0.0 | ... | 6.0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **209588** | 209589 | 1 | 22758I85348 | 404.0 | 151.872333 | 151.872333 | 1089.19 | 1089.19 | 1.0 | 0.0 | ... | 6.0 | |
| **209589** | 209590 | 1 | 95583I84455 | 1075.0 | 36.936000 | 36.936000 | 1728.36 | 1728.36 | 4.0 | 0.0 | ... | 6.0 | |
| **209590** | 209591 | 1 | 28556I85350 | 1013.0 | 11843.111670 | 11904.350000 | 5861.83 | 8893.20 | 3.0 | 0.0 | ... | 12.0 | |
| **209591** | 209592 | 1 | 59712I82733 | 1732.0 | 12488.228330 | 12574.370000 | 411.83 | 984.58 | 2.0 | 38.0 | ... | 12.0 | |
| **209592** | 209593 | 1 | 65061I85339 | 1581.0 | 4489.362000 | 4534.820000 | 483.92 | 631.20 | 13.0 | 0.0 | ... | 12.0 | |

209593 rows × 37 columns

```
In [3]: data.columns
```

```
Out[3]: 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'],
              dtype='object')
```

```
In [4]: data.shape
```

```
Out[4]: (209593, 37)
```

```
In [5]: data.dtypes
```

```
Out[5]: Unnamed: 0                int64
        label                     int64
        msisdn                   object
        aon                     float64
        daily_decr30            float64
        daily_decr90            float64
        rental30                float64
        rental90                float64
        last_rech_date_ma       float64
        last_rech_date_da       float64
        last_rech_amt_ma          int64
        cnt_ma_rech30             int64
        fr_ma_rech30            float64
        sumamnt_ma_rech30       float64
        medianamnt_ma_rech30    float64
        medianmarechprebal30    float64
        cnt_ma_rech90             int64
        fr_ma_rech90              int64
        sumamnt_ma_rech90         int64
        medianamnt_ma_rech90    float64
        medianmarechprebal90    float64
        cnt_da_rech30           float64
        fr_da_rech30            float64
        cnt_da_rech90             int64
```

- Here, as I can see that I, have the features with “Float and int” datatypes except the feature “pdate” which is “object” datatype.

# Statistical Analysis of the data

```
data.describe()
```

| | Unnamed: 0 | label | aon | daily_decr30 | daily_decr90 | rental30 | rental90 | last_rech_date_ma | last_rech_date_da | last_r |
|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 209593.000000 | 209593.000000 | 209593.000000 | 209593.000000 | 209593.000000 | 209593.000000 | 209593.000000 | 209593.00000 | 209593.000000 | 20 |
| **mean** | 104797.000000 | 0.875177 | 8112.343445 | 5381.402289 | 6082.515068 | 2692.581910 | 3483.406534 | 3755.84780 | 3712.202921 | |
| **std** | 60504.431823 | 0.330519 | 75696.082531 | 9220.623400 | 10918.812767 | 4308.586781 | 5770.461279 | 53905.89223 | 53374.833430 | |
| **min** | 1.000000 | 0.000000 | -48.000000 | -93.012667 | -93.012667 | -23737.140000 | -24720.580000 | -29.00000 | -29.000000 | |
| **25%** | 52399.000000 | 1.000000 | 246.000000 | 42.440000 | 42.692000 | 280.420000 | 300.260000 | 1.00000 | 0.000000 | |
| **50%** | 104797.000000 | 1.000000 | 527.000000 | 1469.175667 | 1500.000000 | 1083.570000 | 1334.000000 | 3.00000 | 0.000000 | |
| **75%** | 157195.000000 | 1.000000 | 982.000000 | 7244.000000 | 7802.790000 | 3356.940000 | 4201.790000 | 7.00000 | 0.000000 | |
| **max** | 209593.000000 | 1.000000 | 999860.755200 | 265926.000000 | 320630.000000 | 198926.110000 | 200148.110000 | 998650.37770 | 999171.809400 | 5 |

8 rows × 34 columns

```
data.describe(include = "O")
```

| | msisdn | pcircle | pdate |
|---|---|---|---|
| **count** | 209593 | 209593 | 209593 |
| **unique** | 186243 | 1 | 82 |
| **top** | 04581I85330 | UPW | 04-07-2016 |
| **freq** | 7 | 209593 | 3150 |

## Documentation:-

*Data has few columns in which the difference between mean and the standard deviation is more and, in few columns, it is less and is appropriate that few columns has mean value higher than standard deviation and also there are few columns in which standard deviation is higher than the mean value and also we can see that statistical analysis of the object datatype columns also in which the unique values of the data are mentioned and also we get more information regarding the frequent values present in the data of the columns.*

### Dropping the unrequired columns:

```
data = data.drop(columns = ["Unnamed: 0","msisdn"])
```

### Checking the count for our label column "label":

```
label_column_count = pd.DataFrame(data["label"].value_counts())
label_column_count
```

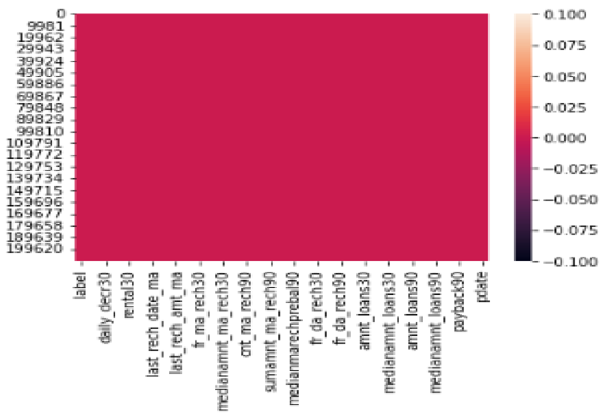
| | label |
|---|---|
| 1 | 183431 |
| 0 | 26162 |

I can see that the label column has imbalanced data in it and I have to balance the data.

### Plotting the heatmap for null-values :-

```
sns.heatmap(data.isnull())
```

<AxesSubplot:>




## Pre-processing:

```
pcircle = pd.DataFrame(data["pcircle"].value_counts())
pcircle
```

| | pcircle |
|---|---|
| UPW | 209593 |

```
data = data.drop(columns = ["pcircle"])
```

## Pre-processing of the column "pdate" :

```
data["Pdate"] = pd.to_datetime(data.pdate,format = "%d-%m-%Y").dt.day
```

```
data["Pmonth"] = pd.to_datetime(data.pdate,format = "%d-%m-%Y").dt.month
```

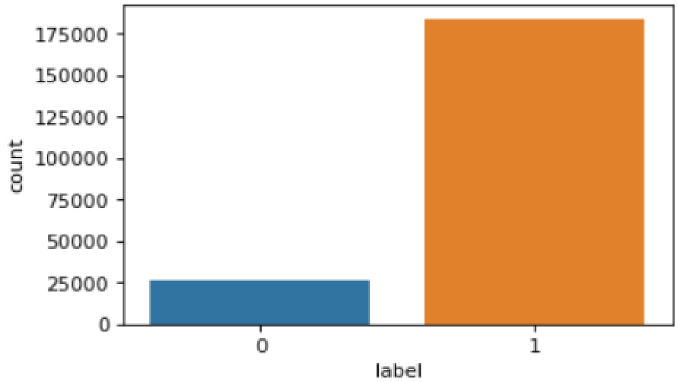
```
data["Pyear"] = pd.to_datetime(data.pdate,format = "%d-%m-%Y").dt.year
```

*I can see that from the column "pdate", multiple columns are extracted with the help of "pd.to_datetime".*

# Visualization

## Label:

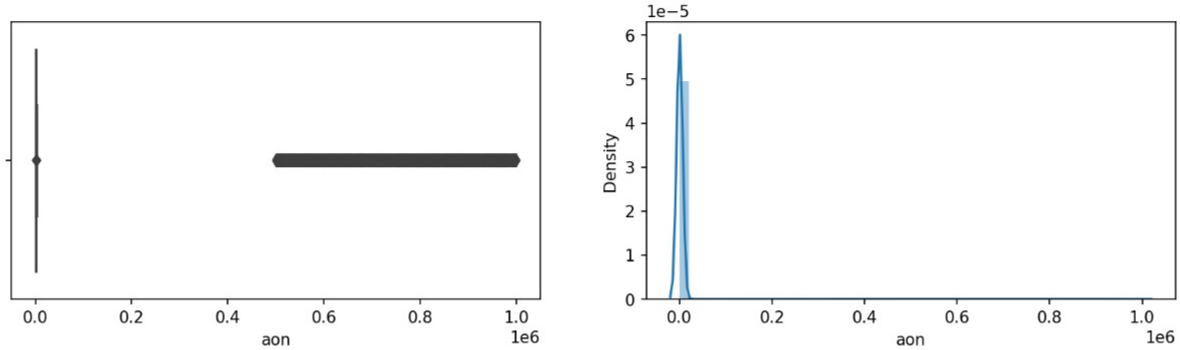
```
plt.figure(figsize=(5,3),dpi=80)
sns.countplot(data.label);
```




*I can see that the column has an attribute(non-defaulter) with very high count than the other attribute (defaulter)*

## Aon :

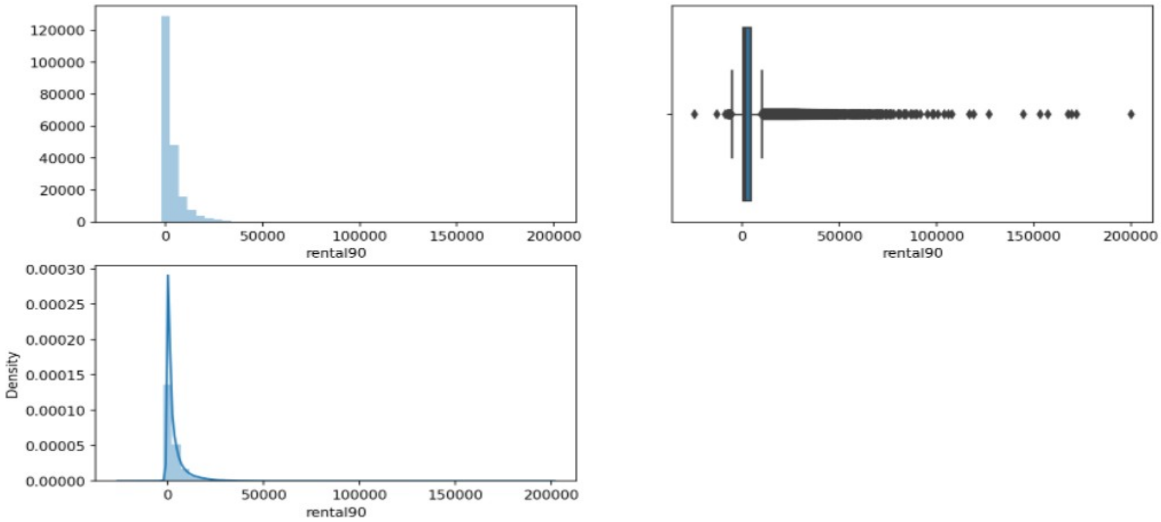
```
plt.figure(figsize=(13,7),dpi=150)
plt.subplot(2,2,1)
sns.boxplot(data['aon']);
plt.subplot(2,2,2)
sns.distplot(data['aon']);
```




I can see that the column has many number of outliers present and also there are dense in nature and the distribution peak is also very narrow.

**Rental90 :**

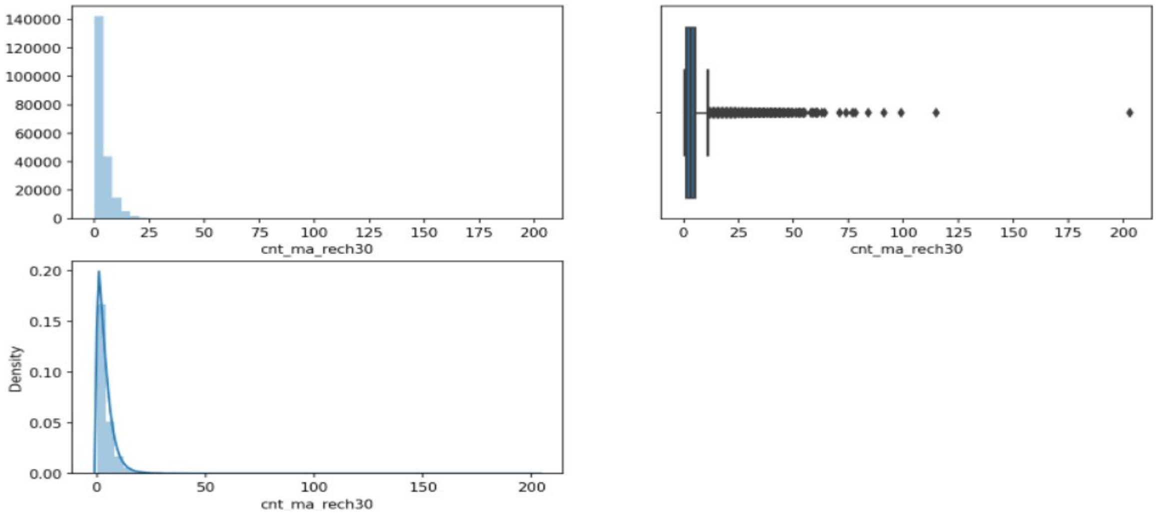
```
plt.figure(figsize=(13,7),dpi=150)
plt.subplot(2,2,1)
sns.distplot(data['rental90'], kde=False);
plt.subplot(2,2,2)
sns.boxplot(data['rental90']);
plt.subplot(2,2,3)
sns.distplot(data['rental90']);
```




I can see that the column has many outliers and the distribution curve has the narrow peak and also has skewness

**Cnt_ma_rech30:**

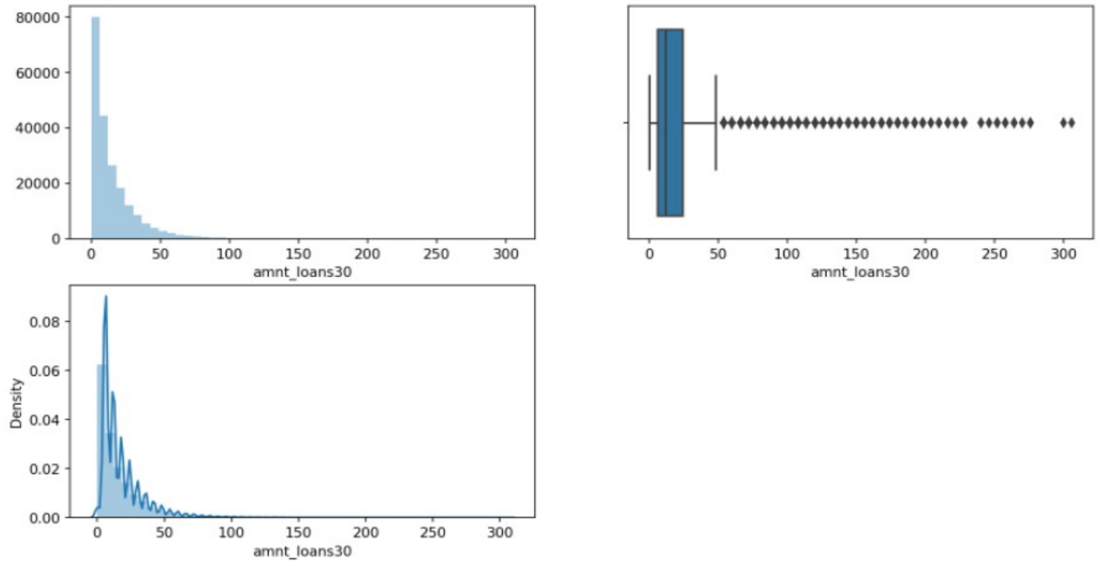
```
plt.figure(figsize=(13,7),dpi=150)
plt.subplot(2,2,1)
sns.distplot(data['cnt_ma_rech30'], kde=False);
plt.subplot(2,2,2)
sns.boxplot(data['cnt_ma_rech30']);
plt.subplot(2,2,3)
sns.distplot(data['cnt_ma_rech30']);
```




I can see that the column has many outliers and the distribution curve has the narrow peak and also has skewness.

**Amnt_loans30 :**

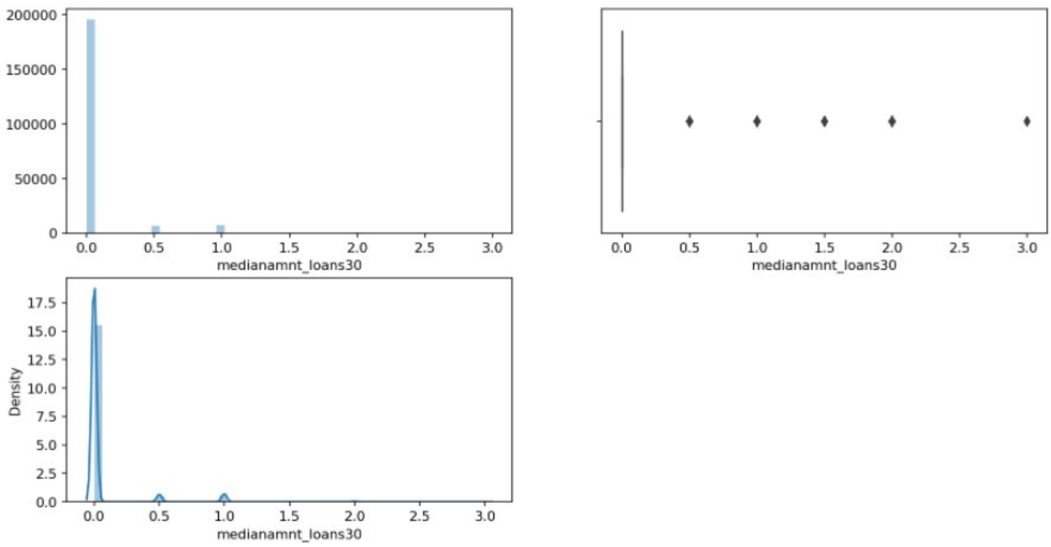
```
plt.figure(figsize=(13,7),dpi=150)
plt.subplot(2,2,1)
sns.distplot(data['amnt_loans30'], kde=False);
plt.subplot(2,2,2)
sns.boxplot(data['amnt_loans30']);
plt.subplot(2,2,3)
sns.distplot(data['amnt_loans30']);
```




I can see that the column has a large number of outliers which are dense in nature and the distribution curve has multiple peaks and also is with skewness.

**Medianamnt_loans :**

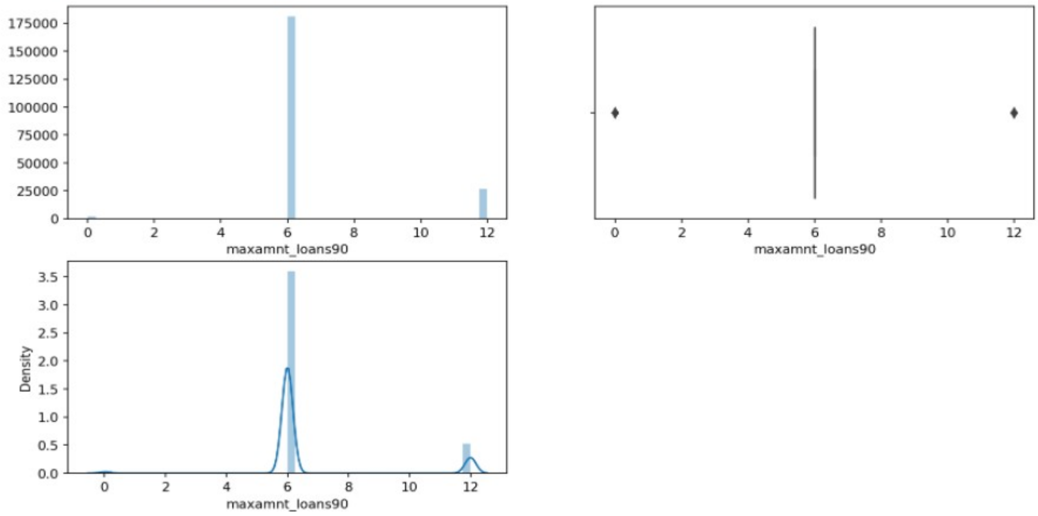
```
plt.figure(figsize=(13,7),dpi=150)
plt.subplot(2,2,1)
sns.distplot(data['medianamnt_loans30'], kde=False);
plt.subplot(2,2,2)
sns.boxplot(data['medianamnt_loans30']);
plt.subplot(2,2,3)
sns.distplot(data['medianamnt_loans30']);
```




I can see that the column has very few outliers which are far away and the distribution curve is with narrow peak and also has more peaks where the data has skewness.

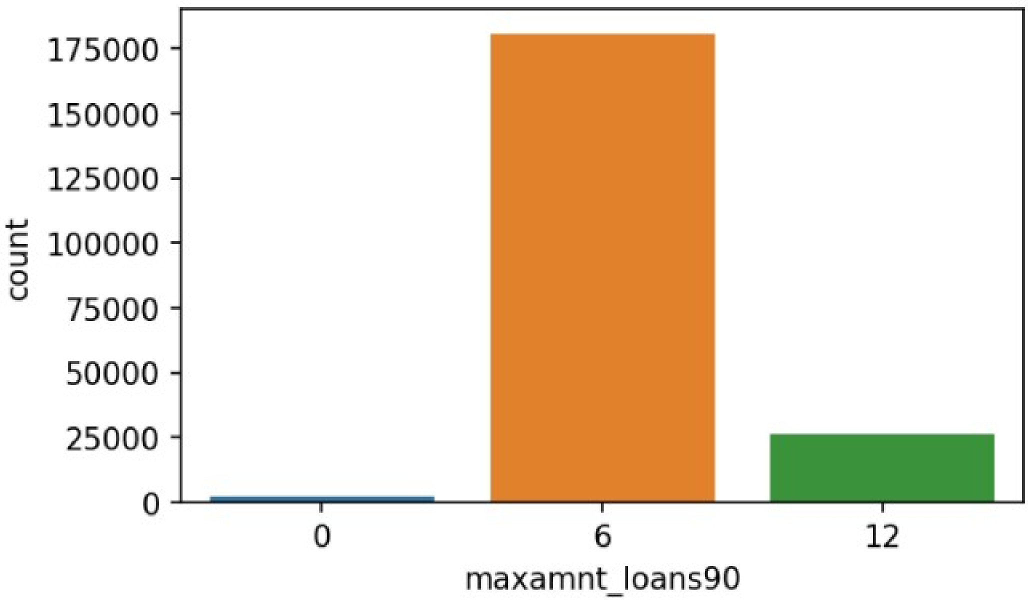
**Maxamnt_loans90 :**

```
plt.figure(figsize=(13,7),dpi=150)
plt.subplot(2,2,1)
sns.distplot(data['maxamnt_loans90'], kde=False);
plt.subplot(2,2,2)
sns.boxplot(data['maxamnt_loans90']);
plt.subplot(2,2,3)
sns.distplot(data['maxamnt_loans90']);
```




I can see that the column has the outliers at the very rare end and is also very far from the quartle which can be negligible.

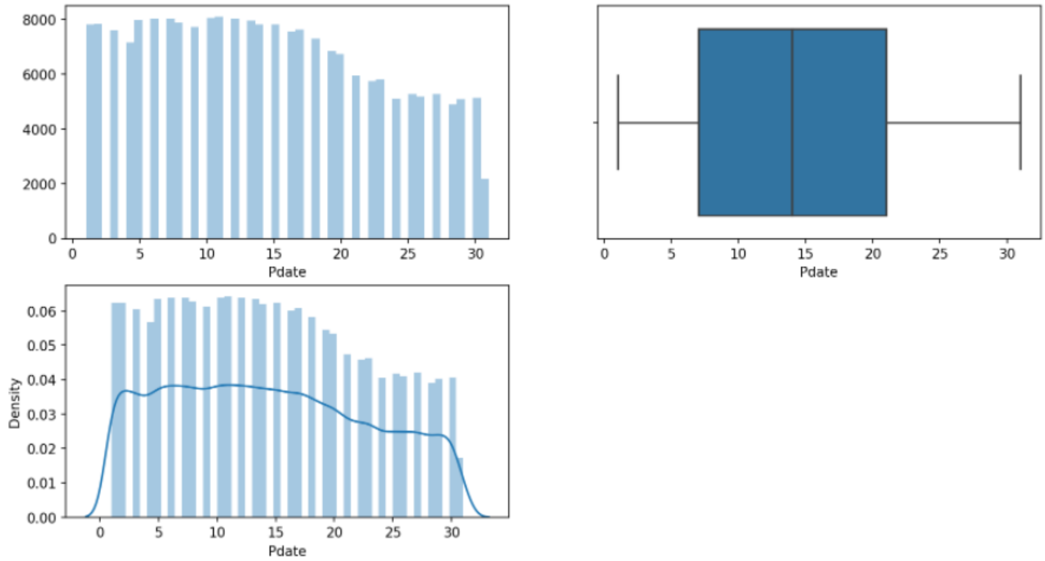
```
plt.figure(figsize=(5,3),dpi=150)
sns.countplot(data.maxamnt_loans90);
```




I can see that the column has the highest count for only one attribute "6" and the least count for the category "0"

**Pdate :**

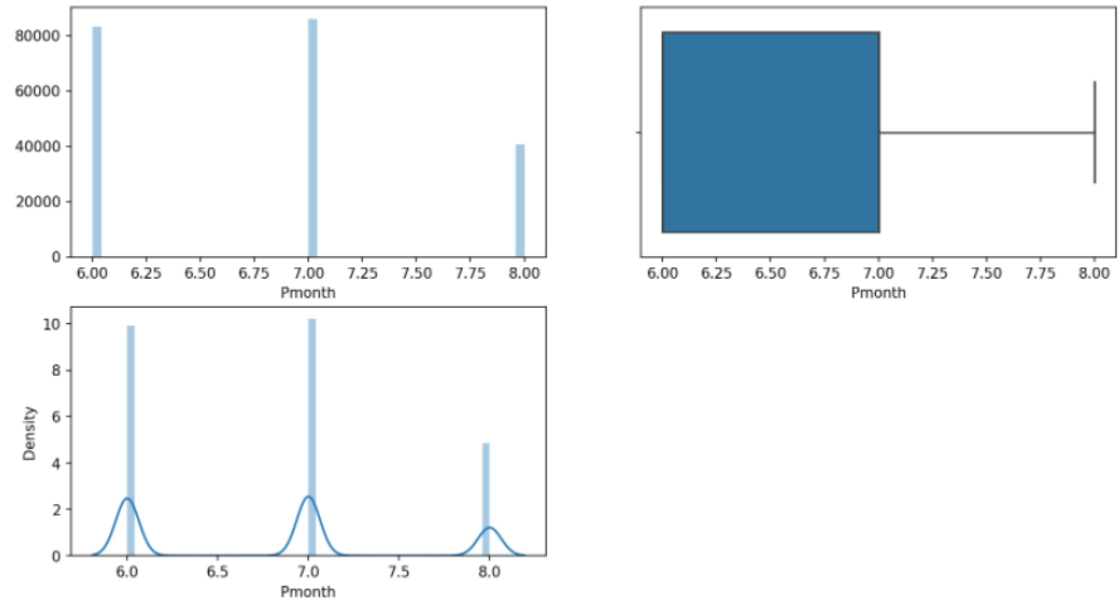
```
plt.figure(figsize=(13,7),dpi=150)
plt.subplot(2,2,1)
sns.distplot(data['Pdate'], kde=False);
plt.subplot(2,2,2)
sns.boxplot(data['Pdate']);
plt.subplot(2,2,3)
sns.distplot(data['Pdate']);
```




I can see that the column has no outliers and the distribution curve is very broad.

**Pmonth :**

```
plt.figure(figsize=(13,7),dpi=150)
plt.subplot(2,2,1)
sns.distplot(data['Pmonth'], kde=False);
plt.subplot(2,2,2)
sns.boxplot(data['Pmonth']);
plt.subplot(2,2,3)
sns.distplot(data['Pmonth']);
```

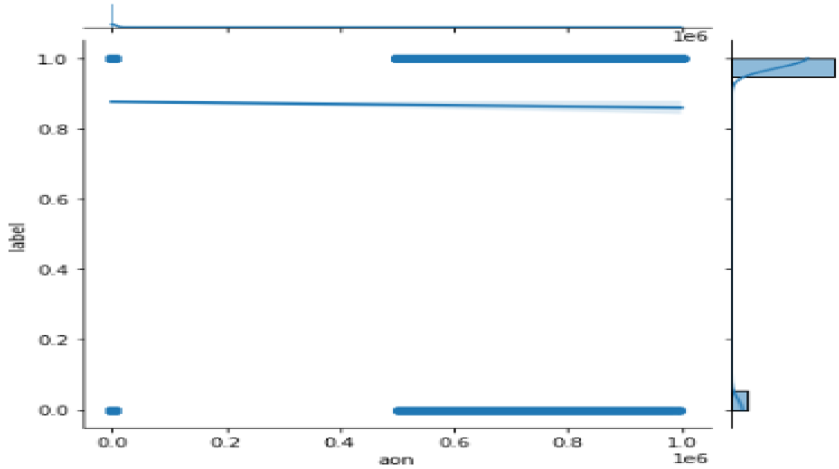



I can see that the column has no outliers seen and the ditribution curve is with multiple peaks.

# Bivariate Analysis

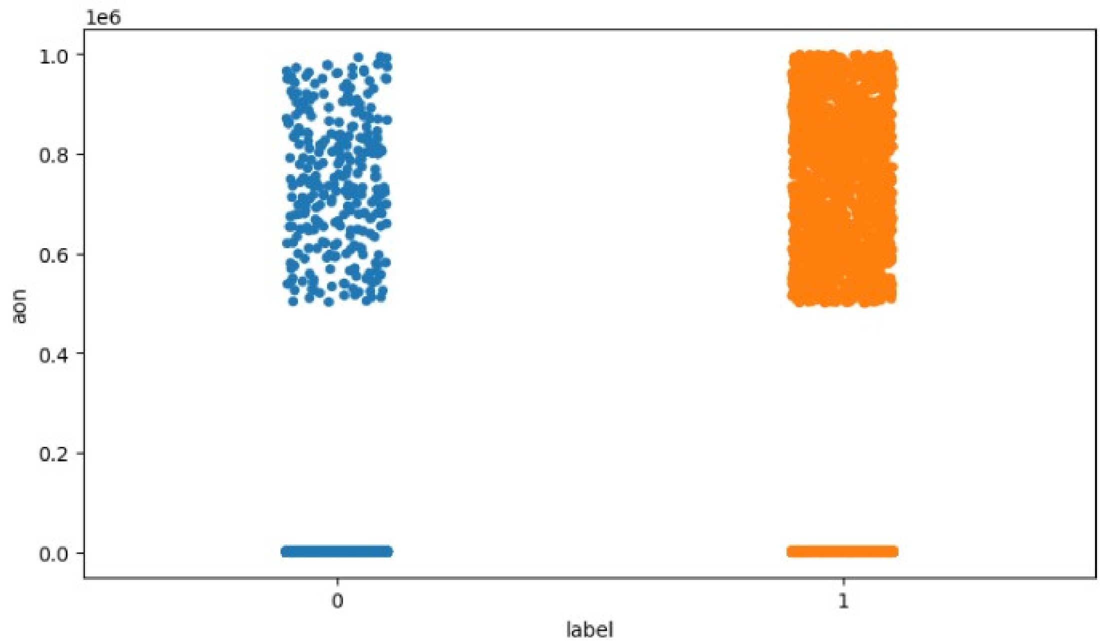
➢ **Aon with label**

```
sns.jointplot(data=data, x='aon', y='label', kind='reg');
```




```
plt.figure(figsize = (9,5),dpi=100)
sns.stripplot(x = 'label',y ='aon', data = data)
```

<AxesSubplot:xlabel='label', ylabel='aon'>

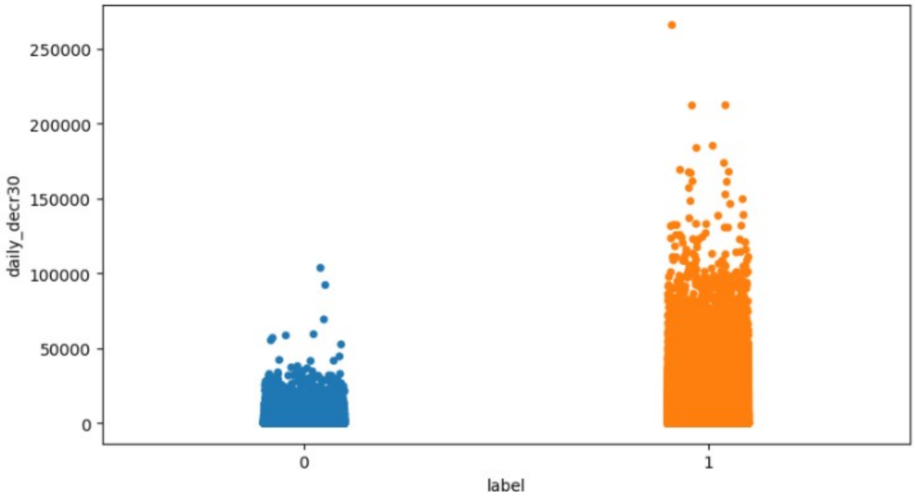



I can see that the label 1 attribute has high and dense customers than the label 0 attribute.

➤ **Daily_decr30 with label**

```
plt.figure(figsize = (9,5),dpi=100)
sns.stripplot(x = 'label',y ='daily_decr30', data = data)
```

<AxesSubplot:xlabel='label', ylabel='daily_decr30'>

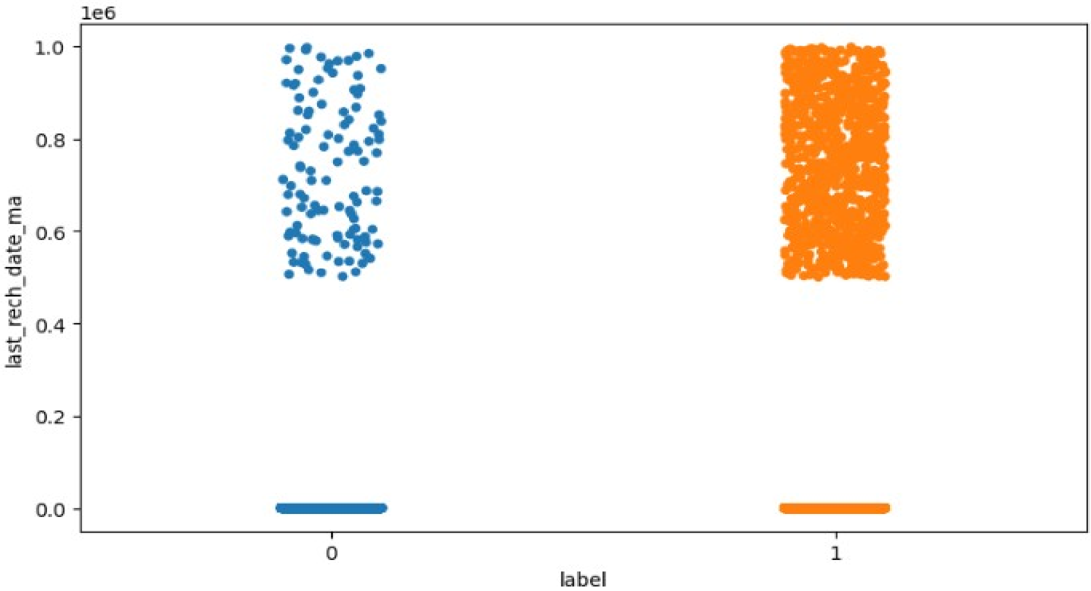



I can see that the high density is present in the label 1 attribute with a high value.

➤ **Last_rech_date_ma with label**

```
plt.figure(figsize = (9,5),dpi=100)
sns.stripplot(x = 'label',y ='last_rech_date_ma', data = data)
```

<AxesSubplot:xlabel='label', ylabel='last_rech_date_ma'>

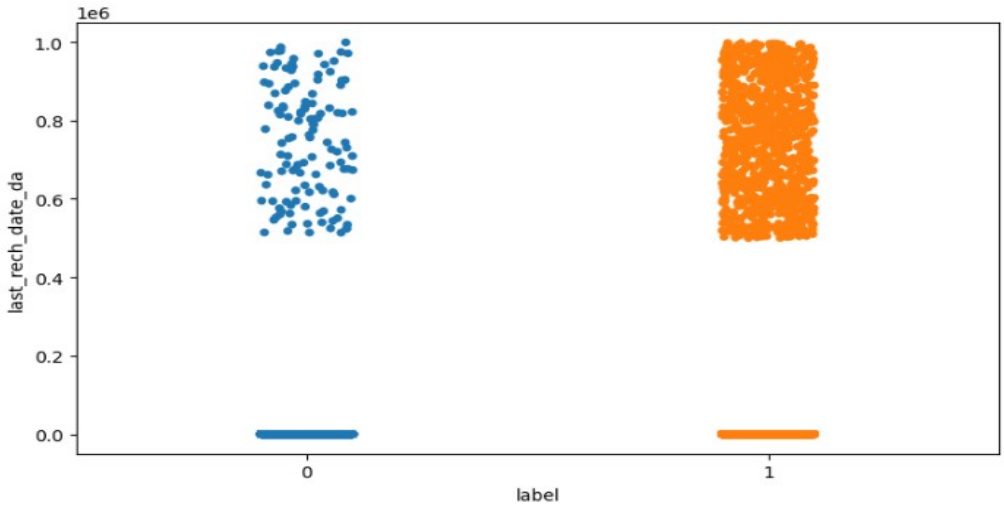



I can see that the high density of customers who have no, of days for last recharge of main account are for label 1 attribute ie.,non-defaulters

- **Last_rech_date_da with label**

```
plt.figure(figsize = (9,5),dpi=100)
sns.stripplot(x = 'label',y ='last_rech_date_da', data = data)
```

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

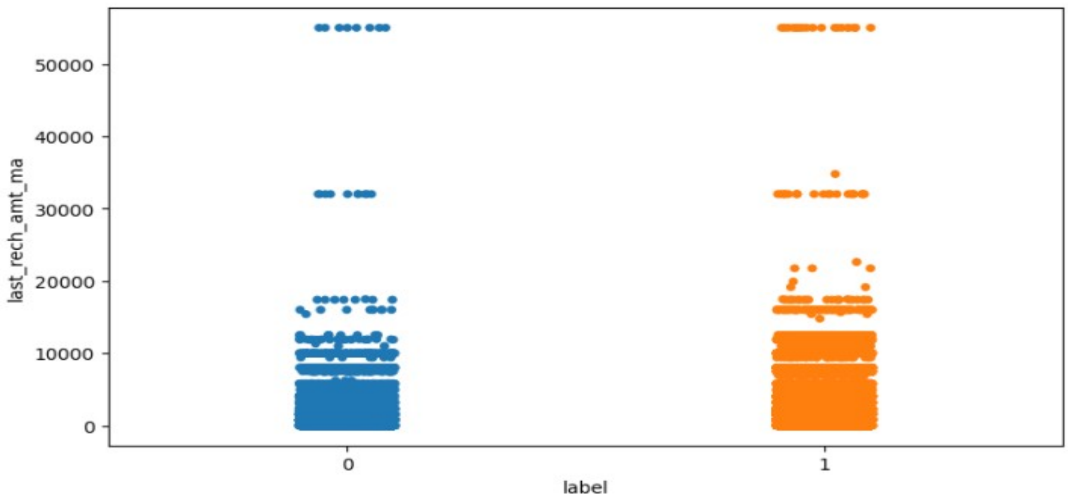



I can see that the high density of customers who have no, of days for last recharge of data account are for label 1 attribute ie.,non-defaulters.

- **Last_rech_amt_ma with label**

```
plt.figure(figsize = (9,5),dpi=100)
sns.stripplot(x = 'label',y ='last_rech_amt_ma', data = data)
```

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

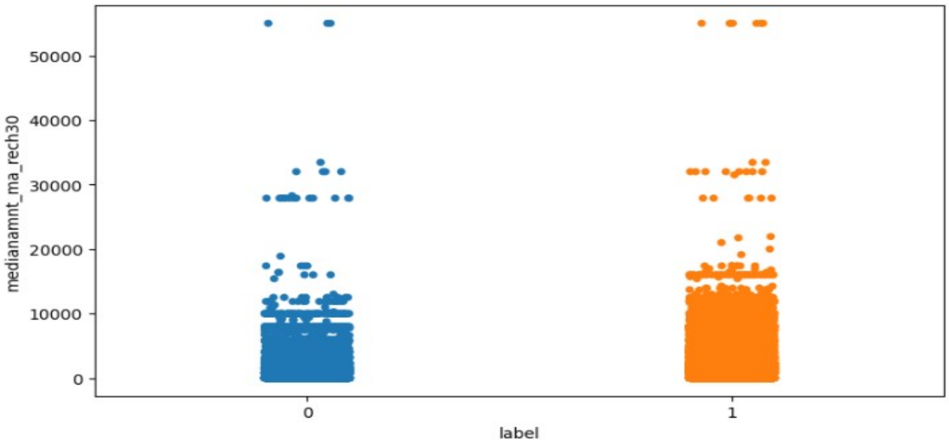



I can see that there is high density of customers is for both the label attributes at their starting points is same but as the amount for the last recharge increases there we can see the decrease in the density of the customers and at the highest last recharge amount point the density for both of the label attributes is almost same but when compared to label 0 ie., defaulters the label 1 ie., non-defaulters attribute has the high density.

- **Medianamnt_ma_rech30 with label**

```
plt.figure(figsize = (9,5),dpi=100)
sns.stripplot(x = 'label',y ='medianamnt_ma_rech30', data = data)
```

```
<AxesSubplot:xlabel='label', ylabel='medianamnt_ma_rech30'>
```

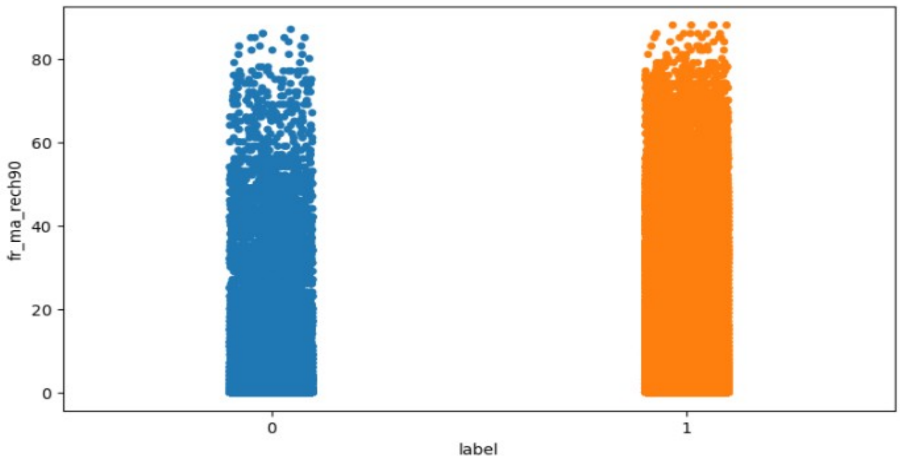



I can see that there is high density of customers is for both the label attributes at their starting points is same but as the amount for the last recharge increases there we can see the decrease in the density of the customers and at the highest last recharge amount point the density for both of the label attributes is almost same but when compared to label 0 ie., defaulters the label 1 ie., non-defaulters attribute has the high density.

- **Fr_ma_rech90 with label**

```
plt.figure(figsize = (9,5),dpi=100)
sns.stripplot(x = 'label',y ='fr_ma_rech90', data = data)
```

```
<AxesSubplot:xlabel='label', ylabel='fr_ma_rech90'>
```

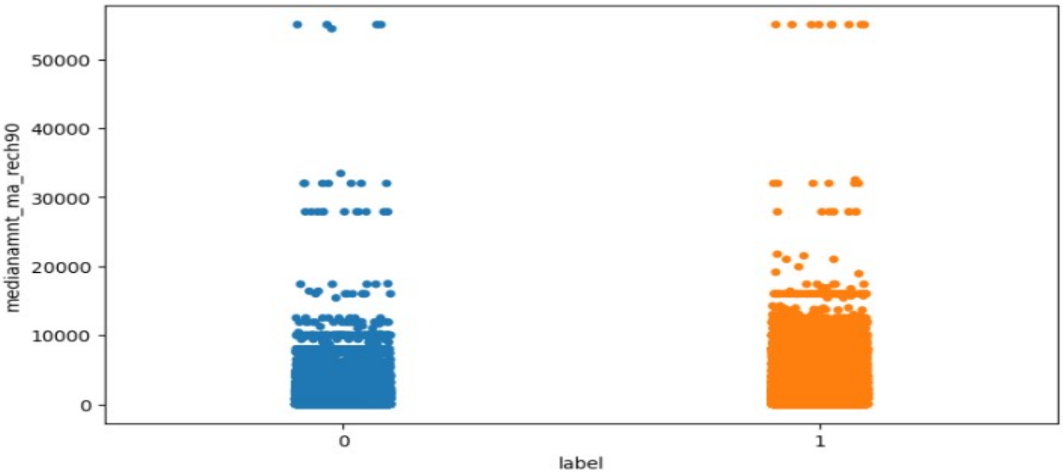



I can see that the high density is present at the starting stage of the frequency of main account recharged but as there is increase in the day count the density decreased in label 0 attribute but there is density remained in the label 1 attribute and at the final point the density becomes less in label 1 attribute and it becomes negligible in label 0 attribute.

➢ **Medianamnt_ma_rech90 with label**

```
plt.figure(figsize = (9,5),dpi=100)
sns.stripplot(x = 'label',y ='medianamnt_ma_rech90', data = data)
```

```
<AxesSubplot:xlabel='label', ylabel='medianamnt_ma_rech90'>
```

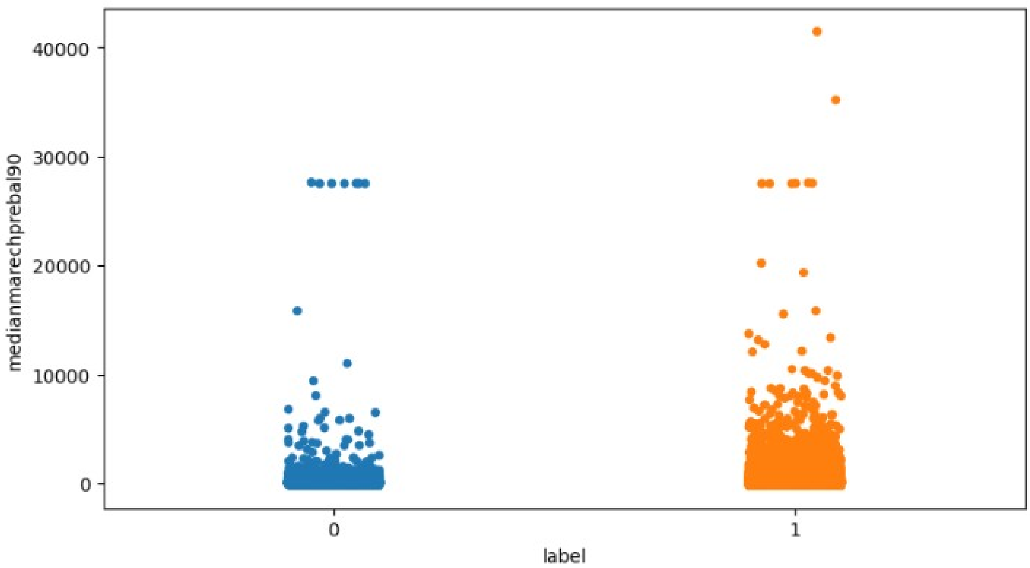



I can see that there is high density of customers is for both the label attributes at their starting points is same but as the median amount for the last recharge of the main account increases there we can see the decrease in the density of the customers and at the highest last recharge amount point the density for both of the label attributes is almost same but when compared to label 0 ie., defaulters the label 1 ie., non-defaulters attribute has the high density.

➢ **Medianmarechprebal90 with label**

```
plt.figure(figsize = (9,5),dpi=100)
sns.stripplot(x = 'label',y ='medianmarechprebal90', data = data)
```

```
<AxesSubplot:xlabel='label', ylabel='medianmarechprebal90'>
```

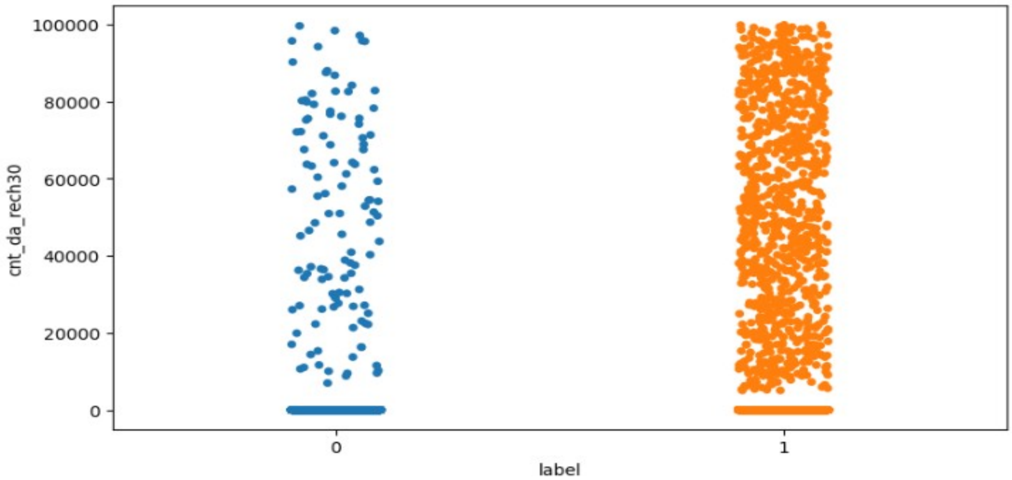



I can see that the high density is present in the label 1 attribute with a high value

➤ **Cnt_da_rech30 with label**

```
plt.figure(figsize = (9,5),dpi=100)
sns.stripplot(x = 'label',y ='cnt_da_rech30', data = data)
```

<AxesSubplot:xlabel='label', ylabel='cnt_da_rech30'>

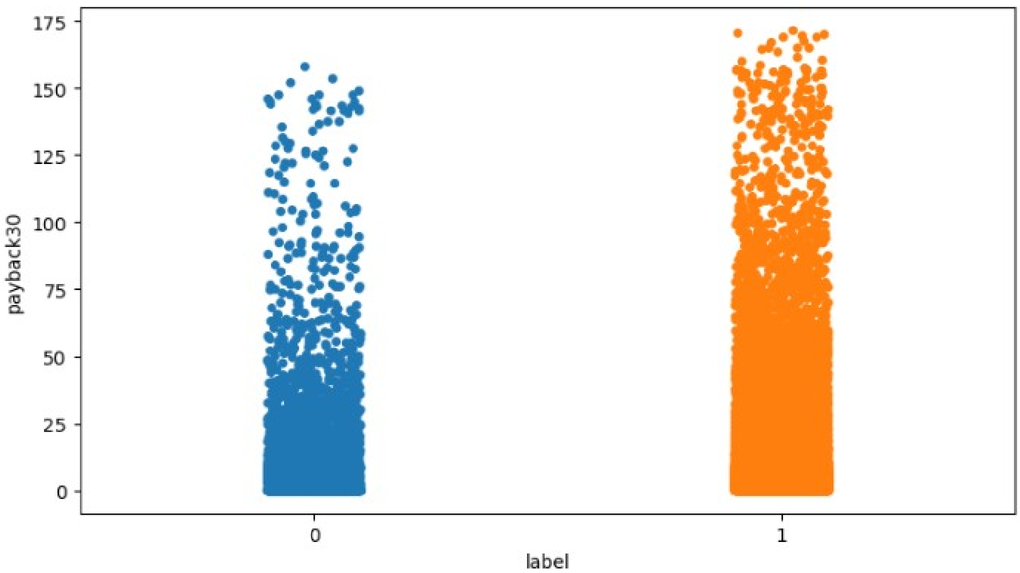



I can see that the high density of customers are for label 1 attribute ie.,non-defaulters.

➤ **Payback30 with label**

```
plt.figure(figsize = (9,5),dpi=100)
sns.stripplot(x = 'label',y ='payback30', data = data)
```

<AxesSubplot:xlabel='label', ylabel='payback30'>

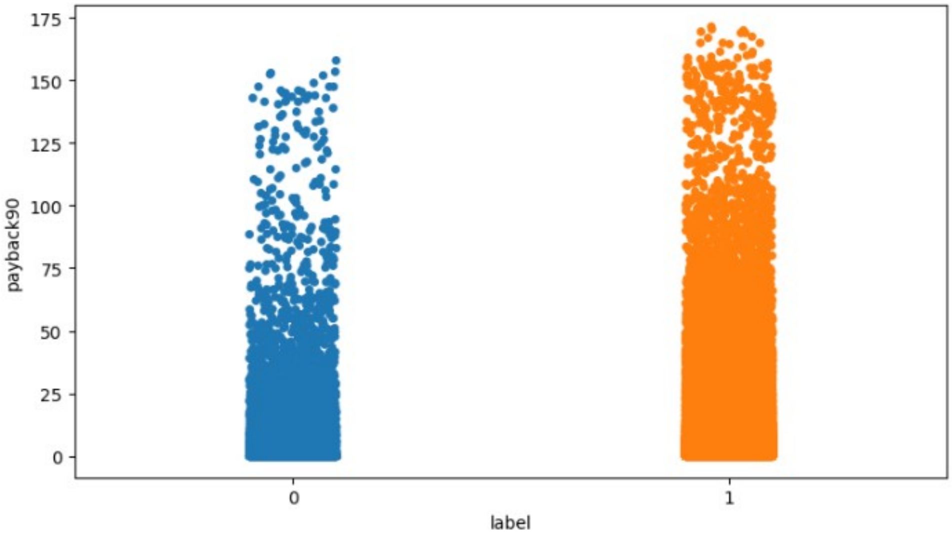



I can see that the high density is present at the starting stage decreased in label 0 attribute but there is density remained in the label 1 attribute and at the final point the density becomes less in label 1 attribute and it becomes negligible in label 0 attribute.

➢ **Payback90 with label**

```
plt.figure(figsize = (9,5),dpi=100)
sns.stripplot(x = 'label',y ='payback90', data = data)
```

```
<AxesSubplot:xlabel='label', ylabel='payback90'>
```




I can see that the high density is present at the starting stage decreased in label 0 attribute but there is density remained in the label 1 attribute and at the final point the density becomes less in label 1 attribute and it becomes negligible in label 0 attribute.

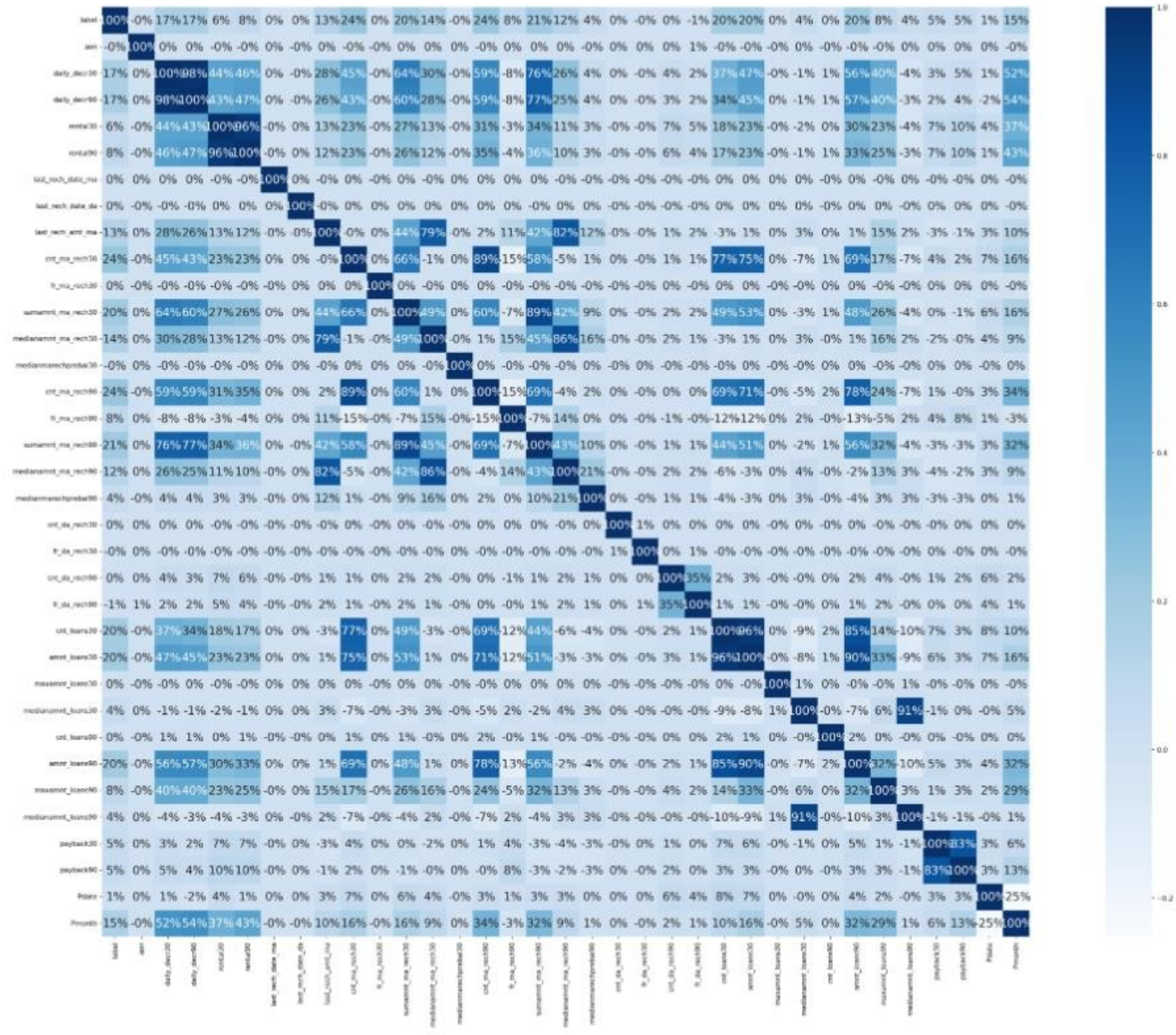
# Correlation :

***Here I find the correlation between the features and the label:***

```
corr_data = data.corr()
corr_data['label'].sort_values(ascending = False)
```

```
label                   1.000000
cnt_ma_rech30           0.237331
cnt_ma_rech90           0.236392
sumamnt_ma_rech90       0.205793
sumamnt_ma_rech30       0.202828
amnt_loans90            0.199788
amnt_loans30            0.197272
cnt_loans30             0.196283
daily_decr30            0.168298
daily_decr90            0.166150
Pmonth                  0.154949
medianamnt_ma_rech30    0.141490
last_rech_amt_ma        0.131804
medianamnt_ma_rech90    0.120855
fr_ma_rech90            0.084385
maxamnt_loans90         0.084144
rental90                0.075521
rental30                0.058085
payback90               0.049183
payback30               0.048336
medianamnt_loans30      0.044589
medianmarechprebal90    0.039300
medianamnt_loans90      0.035747
Pdate                   0.006825
cnt_loans90             0.004733
cnt_da_rech30           0.003827
last_rech_date_ma       0.003728
cnt_da_rech90           0.002999
last_rech_date_da       0.001711
fr_ma_rech30            0.001330
maxamnt_loans30         0.000248
fr_da_rech30           -0.000027
aon                    -0.003785
medianmarechprebal30   -0.004829
fr_da_rech90           -0.005418
Name: label, dtype: float64
```

```
plt.figure(figsize =(40,25),dpi=150)
sns.heatmap(corr_data,annot = True,fmt = ".0%",cbar = True,square = True,annot_kws = {'size': 17 }, cmap = 'Blues')
plt.show()
```

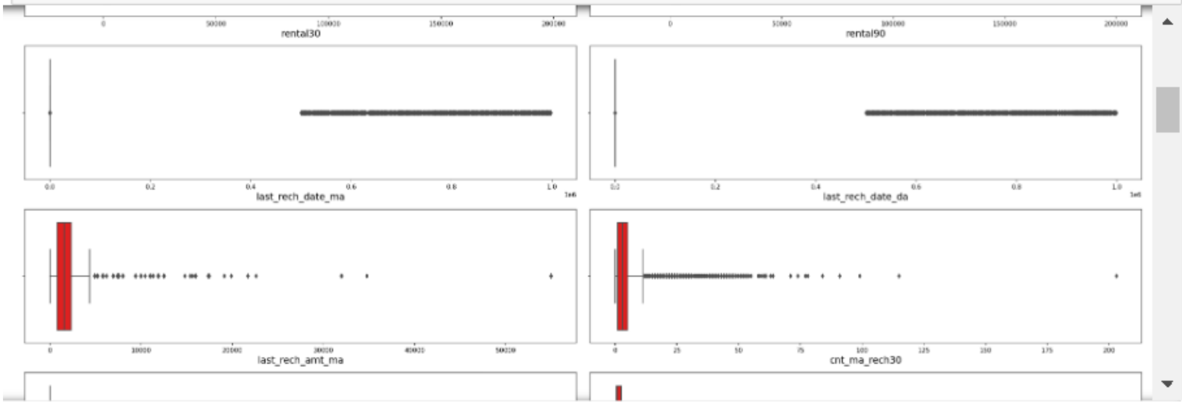


Documentation :

- I can see that the maximum correlation is present between:
- Medianamnt_loans90 and medianamnt_loans30
- Rental_90 and rental_30
- Daily_decr90 and daily_decr30
- Amnt_loans90 and amnt_loans90
- Cnt-loans30 and amnt_loans30 etc.
- I have a lot features which with high correlation above 80%

# Detection of the Outliers

```
plt.figure(figsize = (25,65),dpi=150)
pltnum = 1
for i in data:
    if pltnum<=36:
        plt.subplot(18,2,pltnum)
        sns.boxplot(data[i], color = 'red', orient = 'h')
        plt.xlabel(i, fontsize = 15)
    pltnum+=1
plt.tight_layout()
```




I can see that most of the columns have outliers present in them and also with dense and also with number of outliers present and so we have to treat them for better accuracy in our model building.

# Treating the outliers

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

```
z = np.abs(zscore(data))
z.shape
```

```
(209593, 35)
```

```
threshold = 5.5
print(np.where(z>5.5))
```

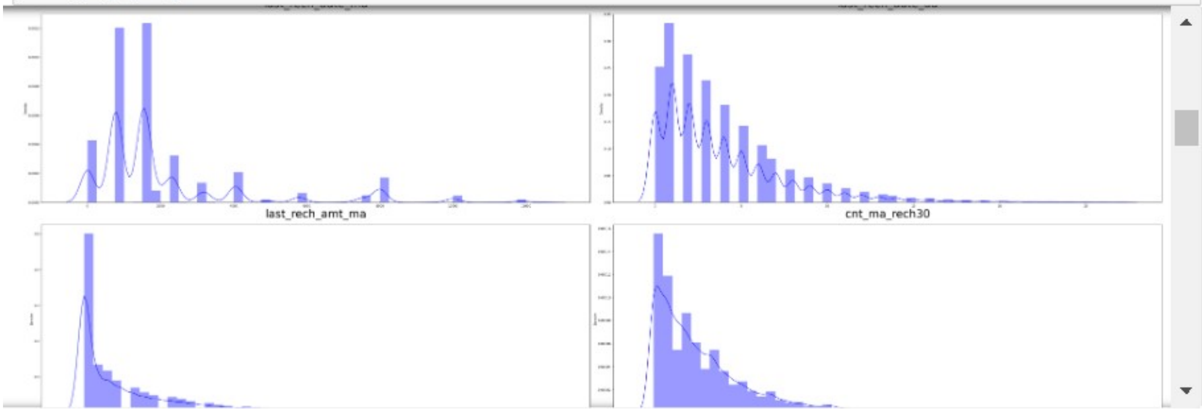
```
(array([     30,      53,      65, ..., 209531, 209533, 209576], dtype=int64), array([6, 6, 1, ..., 7, 6, 1], dtype=int64))
```

```
data_new = data[(z<5.5).all(axis = 1)]
print(data.shape)
print(data_new.shape)
```

```
(209593, 35)
(192459, 35)
```

# Checking for Skewness

```
plt.figure(figsize = (50,165),dpi=150)
pltnum = 1
for i in data_new:
    if pltnum<=36:
        plt.subplot(18,2,pltnum)
        sns.distplot(data_new[i], color = 'blue')
        plt.xlabel(i, fontsize = 35)
    pltnum+=1
plt.tight_layout()
```




```
data_new.skew().sort_values()
```

```
label                   -2.248861
Pdate                    0.206862
Pmonth                   0.371847
aon                      0.947905
cnt_ma_rech30            1.730082
maxamnt_loans90          1.747901
cnt_ma_rech90            1.891819
cnt_loans30              1.945914
amnt_loans30             1.967762
fr_ma_rech30             2.010125
sumamnt_ma_rech30        2.176749
last_rech_amt_ma         2.221092
fr_ma_rech90             2.228096
amnt_loans90             2.244584
sumamnt_ma_rech90        2.268428
daily_decr30             2.372679
medianamnt_ma_rech30     2.451058
medianamnt_ma_rech90     2.465278
daily_decr90             2.514251
rental30                 2.561126
rental90                 2.689101
last_rech_date_ma        3.097659
payback90                3.601847
payback30                3.903939
medianamnt_loans30       4.075996
medianamnt_loans90       4.453088
medianmarechprebal90     5.252733
cnt_da_rech90            7.427612
last_rech_date_da        9.974328
medianmarechprebal30    10.838790
cnt_da_rech30           35.421656
maxamnt_loans30         37.890780
cnt_loans90             54.802217
fr_da_rech90            68.727574
fr_da_rech30            88.162720
dtype: float64
```

Most of the features have skewness and except the label column all the other feature columns are positively skewed, in that few columns are with medium positive skewness and there are also few columns with very high positive skewness.

# Before removing the skewness we will split the data into features and label

```
x = data_new.drop(columns = 'label')
y = data_new['label']
```

```
x
```

| | aon | daily_decr30 | daily_decr90 | rental30 | rental90 | last_rech_date_ma | last_rech_date_da | last_rech_amt_ma | cnt_ma_rech30 | fr_ma_rech30 | ... | max |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 272.0 | 3055.050000 | 3065.150000 | 220.13 | 260.13 | 2.0 | 0.0 | 1539 | 2 | 21.0 | ... | |
| **1** | 712.0 | 12122.000000 | 12124.750000 | 3691.26 | 3691.26 | 20.0 | 0.0 | 5787 | 1 | 0.0 | ... | |
| **2** | 535.0 | 1398.000000 | 1398.000000 | 900.13 | 900.13 | 3.0 | 0.0 | 1539 | 1 | 0.0 | ... | |
| **3** | 241.0 | 21.228000 | 21.228000 | 159.42 | 159.42 | 41.0 | 0.0 | 947 | 0 | 0.0 | ... | |
| **4** | 947.0 | 150.619333 | 150.619333 | 1098.90 | 1098.90 | 4.0 | 0.0 | 2309 | 7 | 2.0 | ... | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **209588** | 404.0 | 151.872333 | 151.872333 | 1089.19 | 1089.19 | 1.0 | 0.0 | 4048 | 3 | 2.0 | ... | |
| **209589** | 1075.0 | 36.936000 | 36.936000 | 1728.36 | 1728.36 | 4.0 | 0.0 | 773 | 4 | 1.0 | ... | |
| **209590** | 1013.0 | 11843.111670 | 11904.350000 | 5861.83 | 8893.20 | 3.0 | 0.0 | 1539 | 5 | 8.0 | ... | |
| **209591** | 1732.0 | 12488.228330 | 12574.370000 | 411.83 | 984.58 | 2.0 | 38.0 | 773 | 5 | 4.0 | ... | |
| **209592** | 1581.0 | 4489.362000 | 4534.820000 | 483.92 | 631.20 | 13.0 | 0.0 | 7526 | 2 | 1.0 | ... | |

192459 rows × 34 columns

```
y
```

```
0        0
1        1
2        1
3        1
4        1
```

# Scaling the data

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler = MinMaxScaler()
scaled = scaler.fit_transform(x)
x = pd.DataFrame(scaled, columns = x.columns)
x
```

| | aon | daily_decr30 | daily_decr90 | rental30 | rental90 | last_rech_date_ma | last_rech_date_da | last_rech_amt_ma | cnt_ma_rech30 | fr_ma_rech30 | ... | m |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.128617 | 0.055251 | 0.047018 | 0.234328 | 0.187155 | 0.218310 | 0.201389 | 0.12312 | 0.074074 | 0.552632 | ... | |
| **1** | 0.305466 | 0.216785 | 0.183925 | 0.335913 | 0.266956 | 0.345070 | 0.201389 | 0.46296 | 0.037037 | 0.000000 | ... | |
| **2** | 0.234325 | 0.025730 | 0.021825 | 0.254229 | 0.202040 | 0.225352 | 0.201389 | 0.12312 | 0.037037 | 0.000000 | ... | |
| **3** | 0.116158 | 0.001202 | 0.001019 | 0.232551 | 0.184812 | 0.492958 | 0.201389 | 0.07576 | 0.000000 | 0.000000 | ... | |
| **4** | 0.399920 | 0.003507 | 0.002975 | 0.260046 | 0.206663 | 0.232394 | 0.201389 | 0.18472 | 0.259259 | 0.052632 | ... | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **192454** | 0.181672 | 0.003529 | 0.002993 | 0.259762 | 0.206437 | 0.211268 | 0.201389 | 0.32384 | 0.111111 | 0.052632 | ... | |
| **192455** | 0.451367 | 0.001481 | 0.001257 | 0.278468 | 0.221303 | 0.232394 | 0.201389 | 0.06184 | 0.148148 | 0.026316 | ... | |
| **192456** | 0.426447 | 0.211817 | 0.180595 | 0.399437 | 0.387943 | 0.225352 | 0.201389 | 0.12312 | 0.185185 | 0.210526 | ... | |
| **192457** | 0.715434 | 0.223310 | 0.190720 | 0.239938 | 0.204004 | 0.218310 | 0.465278 | 0.06184 | 0.185185 | 0.105263 | ... | |
| **192458** | 0.654743 | 0.080805 | 0.069228 | 0.242048 | 0.195785 | 0.295775 | 0.201389 | 0.60208 | 0.074074 | 0.026316 | ... | |

192459 rows × 34 columns

I transform the features present in the variable x to be scaled and then we will get a scaled and transformed data of x, which is used for removing the skewness present in the features.

# Removing skewness

```
from sklearn.preprocessing import power_transform
```

```
transform_data = power_transform(x, method = 'yeo-johnson')
x = pd.DataFrame(transform_data, columns = x.columns)
x
```

| | aon | daily_decr30 | daily_decr90 | rental30 | rental90 | last_rech_date_ma | last_rech_date_da | last_rech_amt_ma | cnt_ma_rech30 | fr_ma_rech30 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | -0.772547 | 0.159357 | 0.095064 | -0.882238 | -0.891112 | -0.400277 | -0.124951 | 0.111881 | -0.374333 | 1.857935 | ... |
| **1** | 0.390238 | 1.451604 | 1.382771 | 0.947386 | 0.707262 | 1.608371 | -0.124951 | 1.734547 | -0.885060 | -1.012938 | ... |
| **2** | -0.008536 | -0.366210 | -0.400540 | -0.363228 | -0.482142 | -0.256674 | -0.124951 | 0.111881 | -0.885060 | -1.012938 | ... |
| **3** | -0.879643 | -0.935841 | -0.921750 | -0.933980 | -0.961697 | 2.937812 | -0.124951 | -0.456250 | -1.510166 | -1.012938 | ... |
| **4** | 0.815744 | -0.876137 | -0.867678 | -0.230492 | -0.367920 | -0.117609 | -0.124951 | 0.651870 | 1.131382 | -0.090680 | ... |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **192454** | -0.359411 | -0.875566 | -0.867160 | -0.236795 | -0.373369 | -0.548592 | -0.124951 | 1.373243 | 0.045864 | -0.090680 | ... |
| **192455** | 1.008365 | -0.928518 | -0.915124 | 0.141702 | -0.041273 | -0.117609 | -0.124951 | -0.656379 | 0.393839 | -0.503695 | ... |
| **192456** | 0.918080 | 1.433202 | 1.367321 | 1.446951 | 1.611074 | -0.256674 | -0.124951 | 0.111881 | 0.683766 | 1.223114 | ... |
| **192457** | 1.706643 | 1.474640 | 1.412874 | -0.724979 | -0.432920 | -0.400277 | 6.198337 | -0.656379 | 0.683766 | 0.521957 | ... |
| **192458** | 1.579608 | 0.508042 | 0.436928 | -0.668165 | -0.646009 | 0.955357 | -0.124951 | 1.927979 | -0.374333 | -0.503695 | ... |

192459 rows × 34 columns

Documentation :

- I can see that I have imported the “powertransform” library and passed the features present in the variable x into the powertransform and then I have removed the skewness of the data and converted into dataframe.

- Now, here we have a look at the skewness of the features whether they are changed or not.

```
x.skew()
```

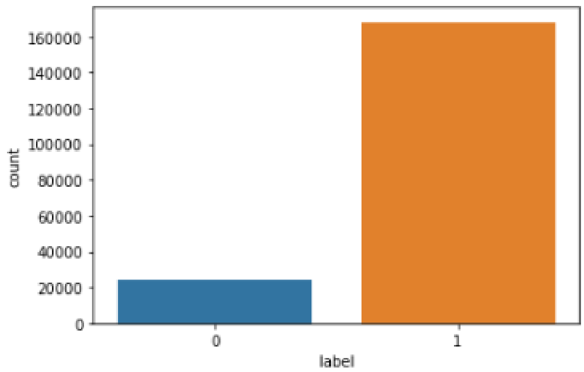
```
aon                      0.070230
daily_decr30             0.340121
daily_decr90             0.374143
rental30                 0.363987
rental90                 0.347998
last_rech_date_ma        0.647721
last_rech_date_da        6.373282
last_rech_amt_ma        -0.187723
cnt_ma_rech30           -0.026788
fr_ma_rech30             0.274985
sumamnt_ma_rech30        0.108371
medianamnt_ma_rech30    -0.083915
medianmarechprebal30     0.531069
cnt_ma_rech90            0.122436
fr_ma_rech90             0.384721
sumamnt_ma_rech90        0.135732
medianamnt_ma_rech90    -0.100244
medianmarechprebal90     0.510589
cnt_da_rech30           10.540065
fr_da_rech30            74.134435
cnt_da_rech90            6.645226
fr_da_rech90            57.088248
cnt_loans30              0.306502
amnt_loans30             0.309343
maxamnt_loans30          2.234694
medianamnt_loans30       3.513169
cnt_loans90              0.246665
amnt_loans90             0.168387
maxamnt_loans90          2.154257
medianamnt_loans90       3.850225
payback30                0.224000
payback90                0.152812
Pdate                   -0.023090
Pmonth                  -0.321926
dtype: float64
```

I can see that most of the features have change in their skewness but still I have skewness present and so we use "cuberoot" from NumPy and then we pass "x" into it and assign to the variable "x" again where we can see that most of the columns have a lot of changes in their skewness except few and so we can proceed with our model building.

# Balancing the data

```
sns.countplot(y)
```

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



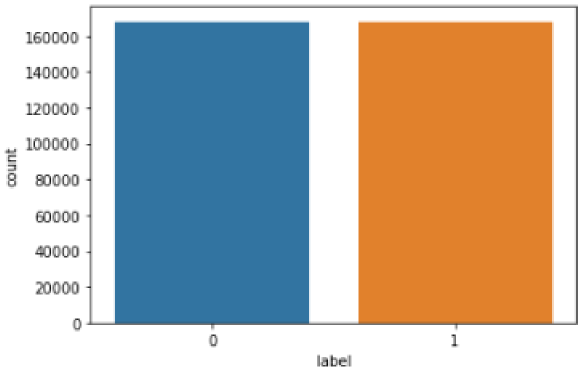

The data is imbalanced as we can see and so we will import "SMOTE" and handle the imbalanced data.

```python
from imblearn.over_sampling import SMOTE
```

```python
sm = SMOTE()
x_over, y_over = sm.fit_resample(x,y)
```

```python
sns.countplot(y_over)
```

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




I can see that I have balanced the imbalanced data and also I can see here that both of the attributes present in the label column which were actually imbalanced are now in equal proportions and so by this I can say that our model is set for training and testing of the data.

# Checking the random state

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, roc_auc_score
from sklearn.model_selection import train_test_split
```

```python
rs = 0
for i in range(0,200):
    x_train,x_test, y_train,y_test = train_test_split(x_over,y_over,test_size = 0.33, random_state = i)
    lg = LogisticRegression()
    lg.fit(x_train,y_train)
    ts_pred = lg.predict(x_test)
    tr_pred = lg.predict(x_train)
    ts_score = accuracy_score(y_test,ts_pred)
    tr_score = accuracy_score(y_train, tr_pred)
    if round(ts_score*100,1) == round(tr_score*100,1):
        if i>rs:
            rs = i
print('the best random state for the data set is', rs)
```

```
the best random state for the data set is 196
```

I have used train_test_split and passed x_over, y_over which are the variables after balancing the data and we used ". fit" method to train the data and predicted the test data and accuracy score for which we got the random state as 196.

```python
x_train,x_test, y_train,y_test = train_test_split(x_over,y_over,test_size = 0.33, random_state = rs)
```

I should proceed with the model testing with the testsize 33% and we present classification report and accuracy score for accuracy score.

# Logistic Regression

```
logreg = LogisticRegression()
logreg.fit(x_train,y_train)
logreg_pred = logreg.predict(x_test)
logreg_score = accuracy_score(y_test,logreg_pred)
logreg_score
```

```
0.7769216905469947
```

```
print(classification_report(y_test, logreg_pred))
```

```
              precision    recall  f1-score   support

           0       0.76      0.81      0.78     55615
           1       0.80      0.74      0.77     55355

    accuracy                           0.78    110970
   macro avg       0.78      0.78      0.78    110970
weighted avg       0.78      0.78      0.78    110970
```

```
print(roc_auc_score(y_test, logreg_pred))
```

```
0.7768460401150915
```

I can see that the model tested with 77% accuracy and the roc_auc_score is 77%.

# Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
```

```
randf = RandomForestClassifier()
randf.fit(x_train,y_train)
randf_pred = randf.predict(x_test)
randf_score = accuracy_score(y_test,randf_pred)
randf_score
```

```
0.9494638190501937
```

```
print(classification_report(y_test, randf_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.95      0.95     55615
           1       0.95      0.95      0.95     55355

    accuracy                           0.95    110970
   macro avg       0.95      0.95      0.95    110970
weighted avg       0.95      0.95      0.95    110970
```

```
print(roc_auc_score(y_test, randf_pred))
```

```
0.9494572919702862
```

I can see that the model tested with 94% accuracy and the roc_auc_score is 94%.

# Extra Trees Classifier

```
from sklearn.ensemble import ExtraTreesClassifier
```

```
extr = ExtraTreesClassifier()
extr.fit(x_train,y_train)
extr_pred = extr.predict(x_test)
extr_score = accuracy_score(y_test,extr_pred)
extr_score
```

```
0.957853473911868
```

```
print(classification_report(y_test, extr_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96     55615
           1       0.97      0.94      0.96     55355

    accuracy                           0.96    110970
   macro avg       0.96      0.96      0.96    110970
weighted avg       0.96      0.96      0.96    110970
```

```
print(roc_auc_score(y_test, extr_pred))
```

```
0.9578216775487678
```

I can see that the model tested with 95.7% accuracy and the roc_auc_score is 95.7%.

# KNN Classifier

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn = KNeighborsClassifier()
knn.fit(x_train,y_train)
knn_pred = knn.predict(x_test)
knn_score = accuracy_score(y_test, knn_pred)
knn_score
```

```
0.8740109939623322
```

```
print(classification_report(y_test, knn_pred))
```

```
              precision    recall  f1-score   support

           0       0.81      0.97      0.89     55615
           1       0.97      0.77      0.86     55355

    accuracy                           0.87    110970
   macro avg       0.89      0.87      0.87    110970
weighted avg       0.89      0.87      0.87    110970
```

```
print(roc_auc_score(y_test, knn_pred))
```

```
0.8737746944433985
```

I can see that the model tested with 87% accuracy and the roc_auc_score is 87%

# Checking for cross validation score

```
from sklearn.model_selection import cross_val_score
```

```
cv1 = cross_val_score(logreg, x_over,y_over,cv = 5)
cv1 = cv1.mean()
cv1
```

```
0.7766242887780394
```

```
cv2 = cross_val_score(randf, x_over,y_over,cv = 5)
cv2 = cv2.mean()
cv2
```

```
0.9482355209212148
```

```
cv3 = cross_val_score(extr, x_over,y_over,cv = 5)
cv3 = cv3.mean()
cv3
```

```
0.9641034996143866
```

```
cv4 = cross_val_score(knn, x_over,y_over,cv = 5)
cv4 = cv4.mean()
cv4
```

```
0.8818129466818341
```

I can see that out of all the models used for prediction, Extra Trees Classifier model is with high accuracy score and also high cross validation score which is 96.4%.

# Model Selection

```
model =[logreg_score, randf_score, extr_score,knn_score]
cross_val = [cv1,cv2,cv3,cv4]
selection = pd.DataFrame({})
selection['model'] = model
selection['cross_val'] = cross_val
selection['difference'] = selection['model'] - selection['cross_val']
selection
```

| | model | cross_val | difference |
|---|---|---|---|
| **0** | 0.776922 | 0.776624 | 0.000297 |
| **1** | 0.949464 | 0.948236 | 0.001228 |
| **2** | 0.957853 | 0.964103 | -0.006250 |
| **3** | 0.874011 | 0.881813 | -0.007802 |

Here we can see that our best model is "Extra Trees Classifier model" with an accuracy of 96% which is highest than the rest of the models and so we choose this model for "hyper parameter tuning"

# Hyper parameter tuning

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

```
params ={'n_estimators':[0,50],
        'criterion':['gini','entropy'],
        'max_depth':[2,4,6],
        'min_samples_split':[2,3,4],
        'bootstrap':[True,False]}
```

```
final = GridSearchCV(ExtraTreesClassifier(),params,cv=5, n_jobs =-1)
final.fit(x_train,y_train)
```

```
GridSearchCV(cv=5, estimator=ExtraTreesClassifier(), n_jobs=-1,
             param_grid={'bootstrap': [True, False],
                         'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6], 'min_samples_split': [2, 3, 4],
                         'n_estimators': [0, 50]})
```

```
final.best_params_
```

```
{'bootstrap': True,
 'criterion': 'entropy',
 'max_depth': 6,
 'min_samples_split': 2,
 'n_estimators': 50}
```

```
final_rf = ExtraTreesClassifier(bootstrap = True, criterion= 'entropy', max_depth = 6, min_samples_split = 3, n_estimators = 50)
final_rf.fit(x_train,y_train)
final_pred = final_rf.predict(x_test)
final_score = accuracy_score(y_test,final_pred)
final_score
```

```
0.7829503469406146
```

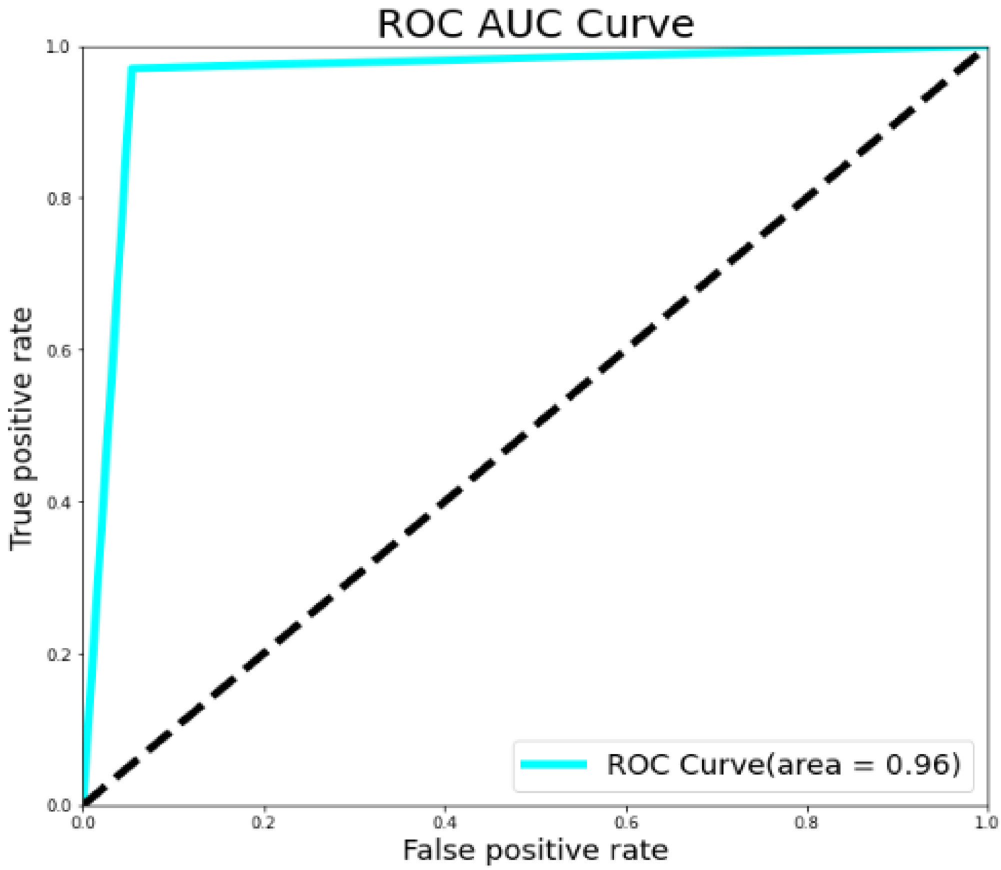
I can see that we have imported "GridSearchCV" and I have used selected parameters and here I use Cross validation "5", and I train the model and select the best parameters and also I have predicted the final accuracy score which is 94.3%

# ROC AUC Curve

```
from sklearn.metrics import roc_curve, auc
```

```
fpr,tpr, thresholds  = roc_curve(extr_pred, y_test)
roc_auc = auc(fpr,tpr)

plt.figure(figsize = (10,9))
plt.plot(fpr, tpr, lw=5, color = 'cyan',label = 'ROC Curve(area = %0.2f)'%roc_auc)
plt.plot([0,1],[0,1],lw =5, color ='black', linestyle = '--')
plt.xlim(0.0,1.0)
plt.ylim(0.0,1.0)
plt.xlabel('False positive rate', fontsize = 18)
plt.ylabel('True positive rate', fontsize  =18)
plt.title('ROC AUC Curve', fontsize = 25)
plt.legend(loc ='lower right', fontsize = 18)
plt.show()
```




```
import joblib
joblib.dump(final,'Micro_credit.pkl')
```

```
['Micro_credit.pkl']
```

## Conclusion:

- *I have built a model, I have used multiple models but the highest score that I have received is of Extra Trees Classifier model. So, this is the best model for predicting the values here.*

- *I have made box plot, so from their I come to know that there were a lot of outliers present, So, I have treated them as well as.*

- *In the dataset there was the problem of skewness that I have observed, So I have treated them also.*

- *These are the keys which are used for model prediction of our dataset: -*

    - *Average precision is 0.96, F1 Score is 0.96 and ROC – AUC Score is also 0.958*

## Limitations and Scope for the Future:

- *There was Class Imbalance which had to be handled because it we don't do that then our model would become biased, So I have to used respected functions to treat this thing and there are chances that now It may effect the model.*

- *As there were a lot of outliers and skewness present, so data loss was also there.*


CLASS OF 2019
Thank You!
ALEX'S GRAD PARTY