FLIP ROBO

Micro-Credit Defaulter Model

Submitted by:

Pritish Khuspe

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 loan defaulters are learned from Data Trained Institute and below documentations.

- https://scikit-learn.org/stable/
- https://seaborn.pydata.org/
- https://www.scipy.org/
- Stack-overflow
- https://imbalanced-learn.org/stable/

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).

Using the historical data of the customer on their recharges, we will be predicting the defaulters with the help of Machine Learning models.

Analytical Problem Framing

The given dataset has 209593 rows and 35 columns. Using this dataset we will be training the Machine Learning models on 77% of the data and the models will be tested on 33% data.

Although the given dataset doesn't have any null value, we can expect outliers and un-realistic values for certain variables.

This data was collected for the UPW telecom circle in the year 2016. Below are the definition for each variable available on the dataset

label	Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}
msisdn	mobile number of user
aon	age on cellular network in days
daily_decr30	Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)
daily_decr90	Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)
rental30	Average main account balance over last 30 days
rental90	Average main account balance over last 90 days
last_rech_date_m a	Number of days till last recharge of main account
last_rech_date_da	Number of days till last recharge of data account
last_rech_amt_ma	Amount of last recharge of main account (in Indonesian Rupiah)
cnt_ma_rech30	Number of times main account got recharged in last 30 days
fr_ma_rech30	Frequency of main account recharged in last 30 days
sumamnt_ma_rec h30	Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)
medianamnt_ma_ rech30	Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)
medianmarechpre bal30	Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)
cnt_ma_rech90	Number of times main account got recharged in last 90 days
fr_ma_rech90	Frequency of main account recharged in last 90 days
sumamnt_ma_rec h90	Total amount of recharge in main account over last 90 days (in Indonesian Rupiah)
medianamnt_ma_ rech90	Median of amount of recharges done in main account over last 90 days at user level (in Indonesian Rupiah)
medianmarechpre bal90	Median of main account balance just before recharge in last 90 days at user level (in Indonesian Rupiah)
cnt_da_rech30	Number of times data account got recharged in last 30 days
fr_da_rech30	Frequency of data account recharged in last 30 days
cnt_da_rech90	Number of times data account got rechargedin last 90 days

fr_da_rech90	Frequency of data account recharged in last 90 days
cnt_loans30	Number of loans taken by user in last 30 days
amnt_loans30	Total amount of loans taken by user in last 30 days
maxamnt_loans3 0	maximum amount of loan taken by the user in last 30 days
medianamnt_loan s30	Median of amounts of loan taken by the user in last 30 days
cnt_loans90	Number of loans taken by user in last 90 days
amnt_loans90	Total amount of loans taken by user in last 90 days
maxamnt_loans9 0	maximum amount of loan taken by the user in last 90 days
medianamnt_loan s90	Median of amounts of loan taken by the user in last 90 days
payback30	Average payback time in days over last 30 days
payback90	Average payback time in days over last 90 days
pcircle	telecom circle
pdate	date

```
dataset = pd.read_csv(r'E:\Post Graduation\internship\Micro-Credit-Project\Data file.csv')
dataset.head()
```

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30
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

5 rows × 37 columns

Pre-Processing:

Just by looking at the glimpse of the dataset, we can see that there are few un-necessary features in the dataset. We'll be removing the same from the dataset.

They are:

1. "Unnamed:0" – This row is a dummy row just like an indexing starting from 1
2. "msisdn" – The data definition column above clearly states that this is a subscriber mobile number and they are randomly generated and will not have any meaning in the prediction of credit defaulters.
3. "pcircle" – This feature is same throughout the rows (UPW) and will not have any effect on the target variable

Post removal of the columns Unnamed:0 and msisdn, We can see the datatypes of the remaining columns

```
pd. set_option("display.max_rows", None)
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209593 entries, 0 to 209592
Data columns (total 35 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   label                 209593 non-null  int64  
 1   aon                   209593 non-null  float64
 2   daily_decr30          209593 non-null  float64
 3   daily_decr90          209593 non-null  float64
 4   rental30              209593 non-null  float64
 5   rental90              209593 non-null  float64
 6   last_rech_date_ma     209593 non-null  float64
 7   last_rech_date_da     209593 non-null  float64
 8   last_rech_amt_ma      209593 non-null  int64  
 9   cnt_ma_rech30         209593 non-null  int64  
 10  fr_ma_rech30          209593 non-null  float64
 11  sumamnt_ma_rech30     209593 non-null  float64
 12  medianamnt_ma_rech30  209593 non-null  float64
 13  medianmarechprebal30  209593 non-null  float64
 14  cnt_ma_rech90         209593 non-null  int64  
 15  fr_ma_rech90          209593 non-null  int64  
 16  sumamnt_ma_rech90     209593 non-null  int64  
 17  medianamnt_ma_rech90  209593 non-null  float64
 18  medianmarechprebal90  209593 non-null  float64
 19  cnt_da_rech30         209593 non-null  float64
 20  fr_da_rech30          209593 non-null  float64
 21  cnt_da_rech90         209593 non-null  int64  
 22  fr_da_rech90          209593 non-null  int64  
 23  cnt_loans30           209593 non-null  int64  
 24  amnt_loans30          209593 non-null  int64  
 25  maxamnt_loans30       209593 non-null  float64
 26  medianamnt_loans30    209593 non-null  float64
 27  cnt_loans90           209593 non-null  float64
 28  amnt_loans90          209593 non-null  int64  
 29  maxamnt_loans90       209593 non-null  int64  
 30  medianamnt_loans90    209593 non-null  float64
 31  payback30             209593 non-null  float64
 32  payback90             209593 non-null  float64
 33  pcircle               209593 non-null  object 
 34  pdate                 209593 non-null  object 
dtypes: float64(21), int64(12), object(2)
memory usage: 56.0+ MB
```

From the above image, I can say that most of the columns are of numerical data type except pcircle and pdate (Telecom circle and Date respectively) and I have the confirmation that the dataset has no null values.

Since we have dropped the pcircle, we will be extracting the features from the date, here we'll be extracting date and month ignoring year because it's the same for every row (i.e., 2016).

Post data extraction from the date column, we deleted the pdate and currently have 35 features

Now that we have all the columns in numerical type. We can explore thedata and its relationship with the target variable.

I have used ".describe" to understand the shape of the data. You can view the snippets of the same below.

```
dataset.describe().iloc[:,:15]
```

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_am
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.00000
mean	0.875177	8112.343445	5381.402289	6082.515068	2692.581910	3483.406534	3755.847800	3712.202921	2064.452797
std	0.330519	75696.082531	9220.623400	10918.812767	4308.586781	5770.461279	53905.892230	53374.833430	2370.786034
min	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000	-29.000000	0.000000
25%	1.000000	246.000000	42.440000	42.692000	280.420000	300.260000	1.000000	0.000000	770.000000
50%	1.000000	527.000000	1469.175667	1500.000000	1083.570000	1334.000000	3.000000	0.000000	1539.000000
75%	1.000000	982.000000	7244.000000	7802.790000	3356.940000	4201.790000	7.000000	0.000000	2309.000000
max	1.000000	999860.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733	999171.809410	55000.00000

```
dataset.describe().iloc[:,15:]
```

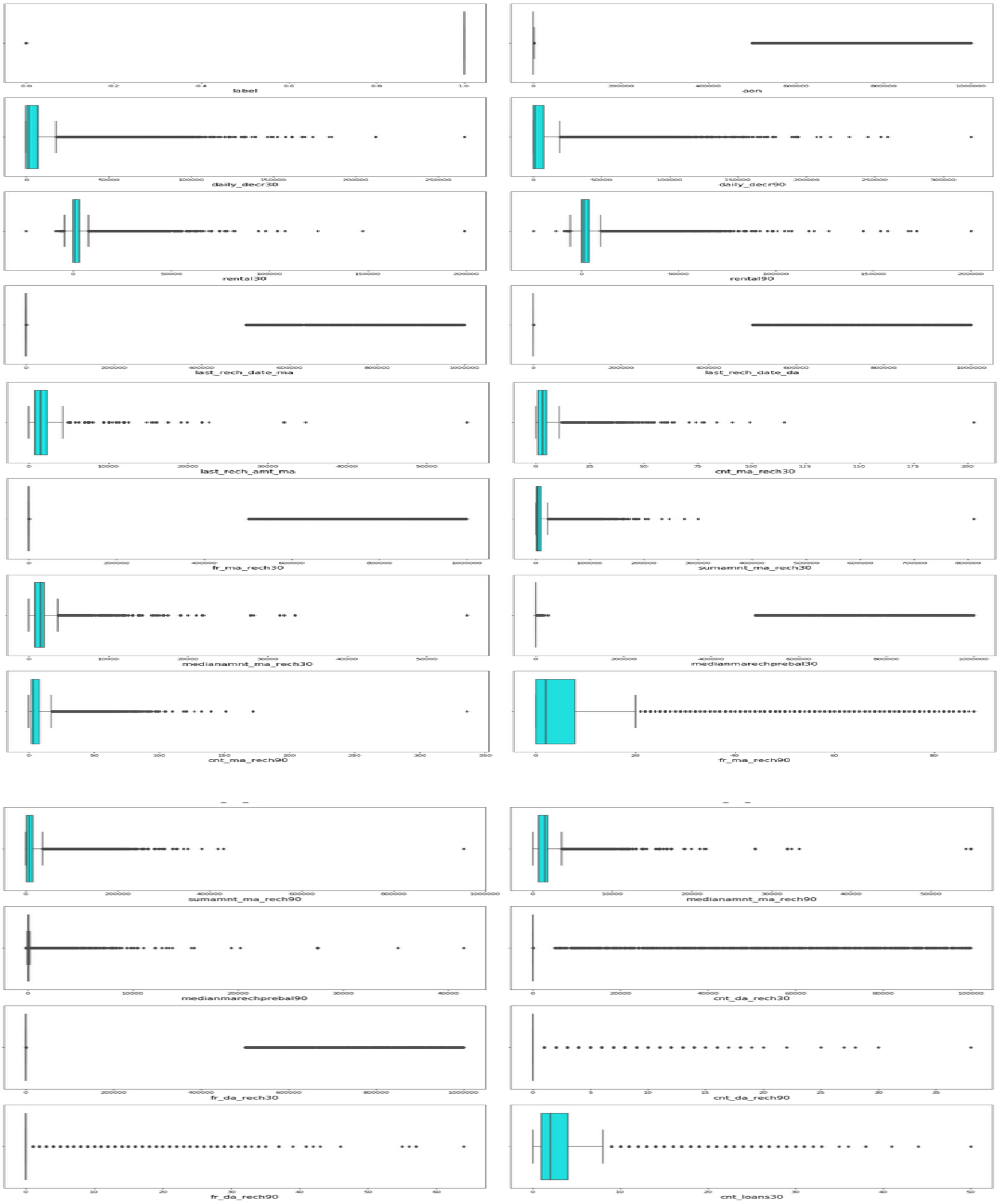
	fr_ma_rech90	sumamnt_ma_rech90	medianamnt_ma_rech90	medianmarechprebal90	cnt_da_rech30	fr_da_rech30	cnt_da_rech90	fr_da_rech
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.00
mean	7.716780	12396.218352	1864.595821	92.025541	262.578110	3749.494447	0.041495	0.045712
std	12.590251	16857.793882	2081.680664	369.215658	4183.897978	53885.414979	0.397556	0.951386
min	0.000000	0.000000	0.000000	-200.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	2317.000000	773.000000	14.600000	0.000000	0.000000	0.000000	0.000000
50%	2.000000	7226.000000	1539.000000	36.000000	0.000000	0.000000	0.000000	0.000000
75%	8.000000	16000.000000	1924.000000	79.310000	0.000000	0.000000	0.000000	0.000000
max	88.000000	953036.000000	55000.000000	41456.500000	99914.441420	999809.240107	38.000000	64.000000

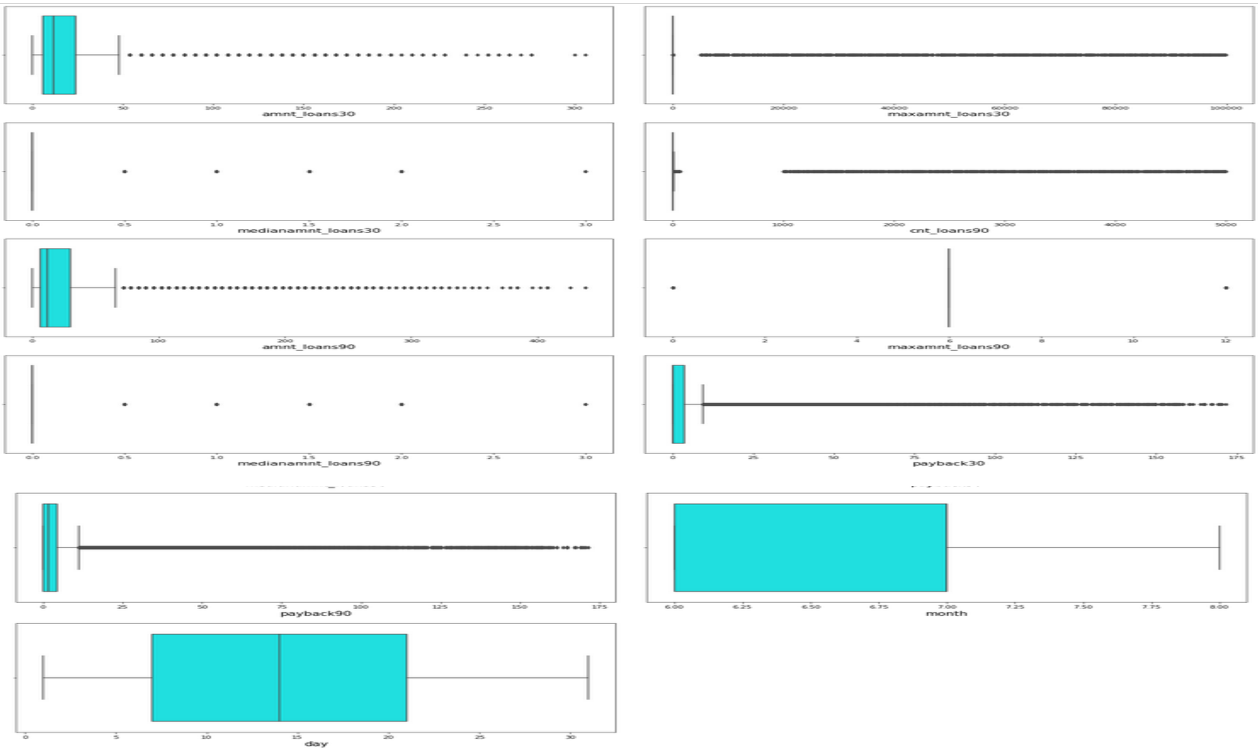
Looking at the variables, I can see that the standard deviation of almost every independent variable is larger than the mean value. This implies that the data might have outliers. Further, we can also see that the max value for the variables are unrealistic and is the reason that the standard deviation is almost twice the mean.

Therefore we can safely assume that the dataset has outliers.

Here we can visualize the outliers in data using boxplot from seaborn documentation.

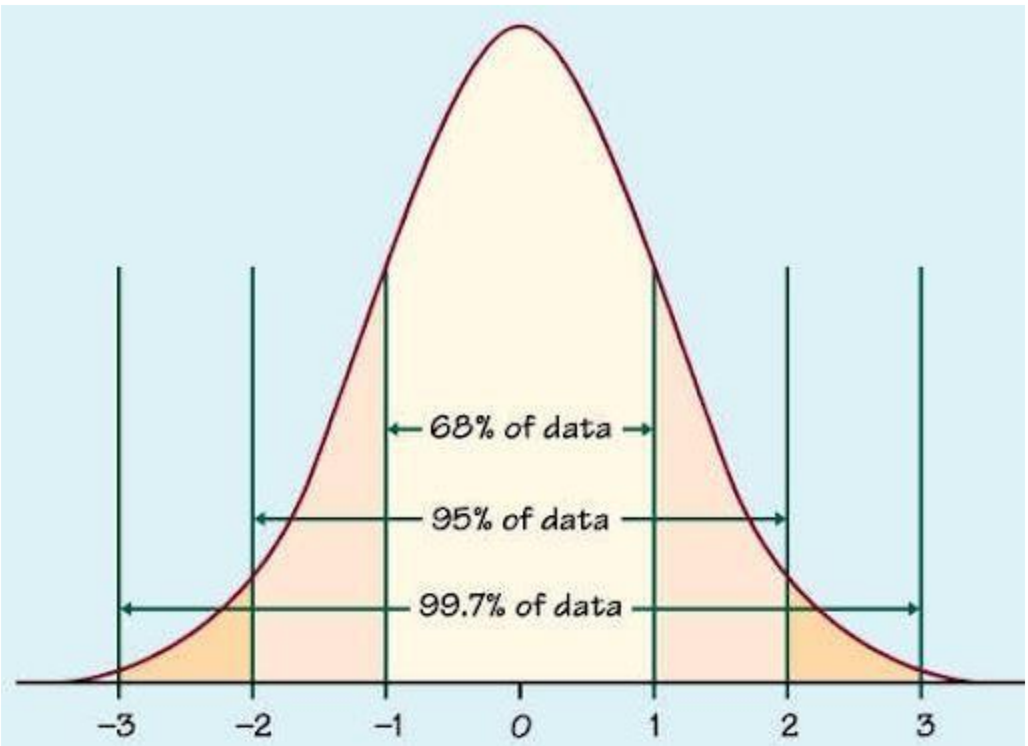
```
plt.figure(figsize = (20,80))
pltnum = 1
for i in dataset:
    if pltnum<=36:
        plt.subplot(18,2,pltnum)
        sns.boxplot(dataset[i], color = 'cyan', orient = 'h')
        plt.xlabel(i, fontsize = 15)
    pltnum+=1
plt.tight_layout()
```





From the above boxplots we can clearly say that most of the continuous variables has a lot of outliers and we can remove them using the statistics.

First we find the p-value of each and every data point and we take only data with less than 2.8 p-value.



It is assumed that close to 99.7% data lies between -3 to +3 standard deviation. We can consider the remaining data (0.03) to be outlier.

Therefore using the z-score method, I'm taking the data within the range of -2.8 to +2.8 standard deviation to control the outlier.

```
z = np.abs(zscore(dataset[['aon','daily_decr90','rental90','last_rech_date_ma','cnt_ma_rech30','cnt_da_rech30','last_rech_date_ma
']]))
z.head()
```

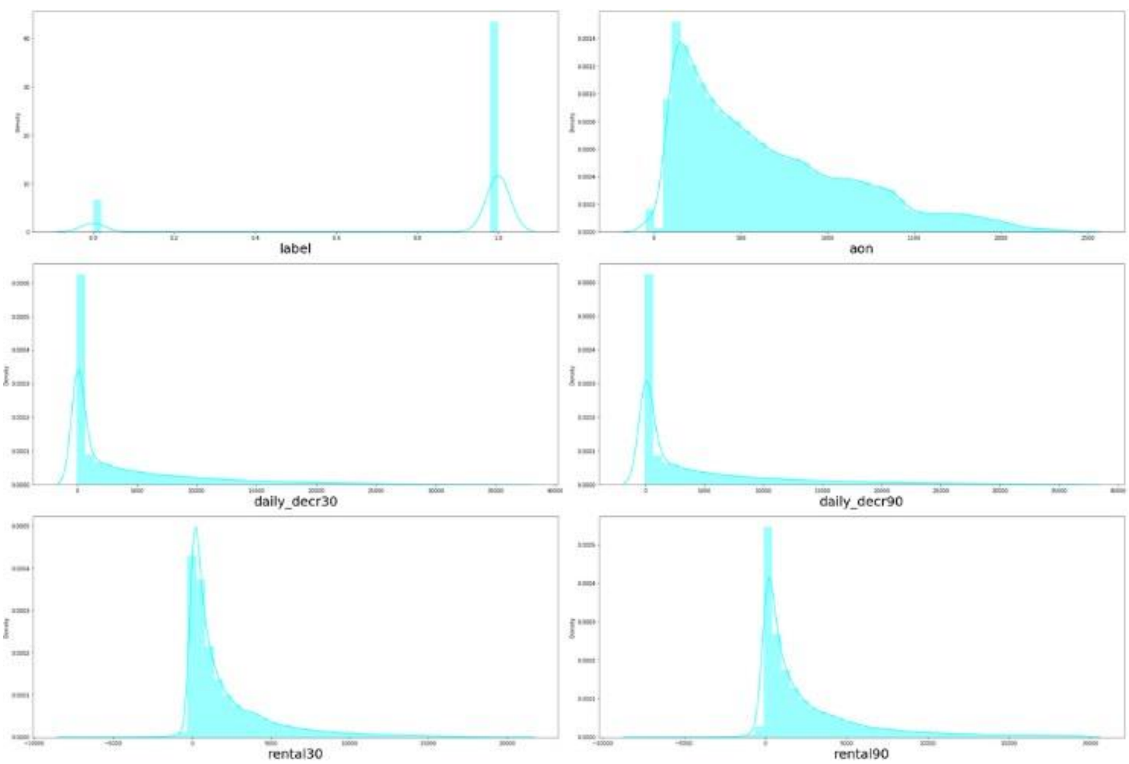
	aon	daily_decr90	rental90	last_rech_date_ma	cnt_ma_rech30	cnt_da_rech30	last_rech_date_ma
0	0.103577	0.276346	0.558583	0.069637	0.464760	0.062759	0.069637
1	0.097764	0.553380	0.036020	0.069303	0.699718	0.062759	0.069303
2	0.100102	0.429033	0.447674	0.069619	0.699718	0.062759	0.069619
3	0.103986	0.555125	0.576036	0.068914	0.934677	0.062759	0.068914
4	0.094660	0.543274	0.413227	0.069600	0.710030	0.062759	0.069600

```
new_data = dataset[(z<2.8).all(axis = 1)]
print(new_data.shape)
print(dataset.shape)
```

```
(192794, 35)
(209593, 35)
```

Post outlier removal I can see that the "new_data" dataset contains 192794 rows out of the actual dataset consisting of 209593 rows. If we are to proceed with outlier then there should be only 7 to 8 % data loss and we are losing close to 8% data loss therefore we are proceeding with the outlier removal.

Now let's check the data distribution to understand the data.



last_rech_date_ma

last_rech_date_da

last_rech_amt_ma

cnt_ma_rech30

fr_ma_rech30

sumamnt_ma_rech30

cnt_ma_rech90

fr_ma_rech90

sumamnt_ma_rech90

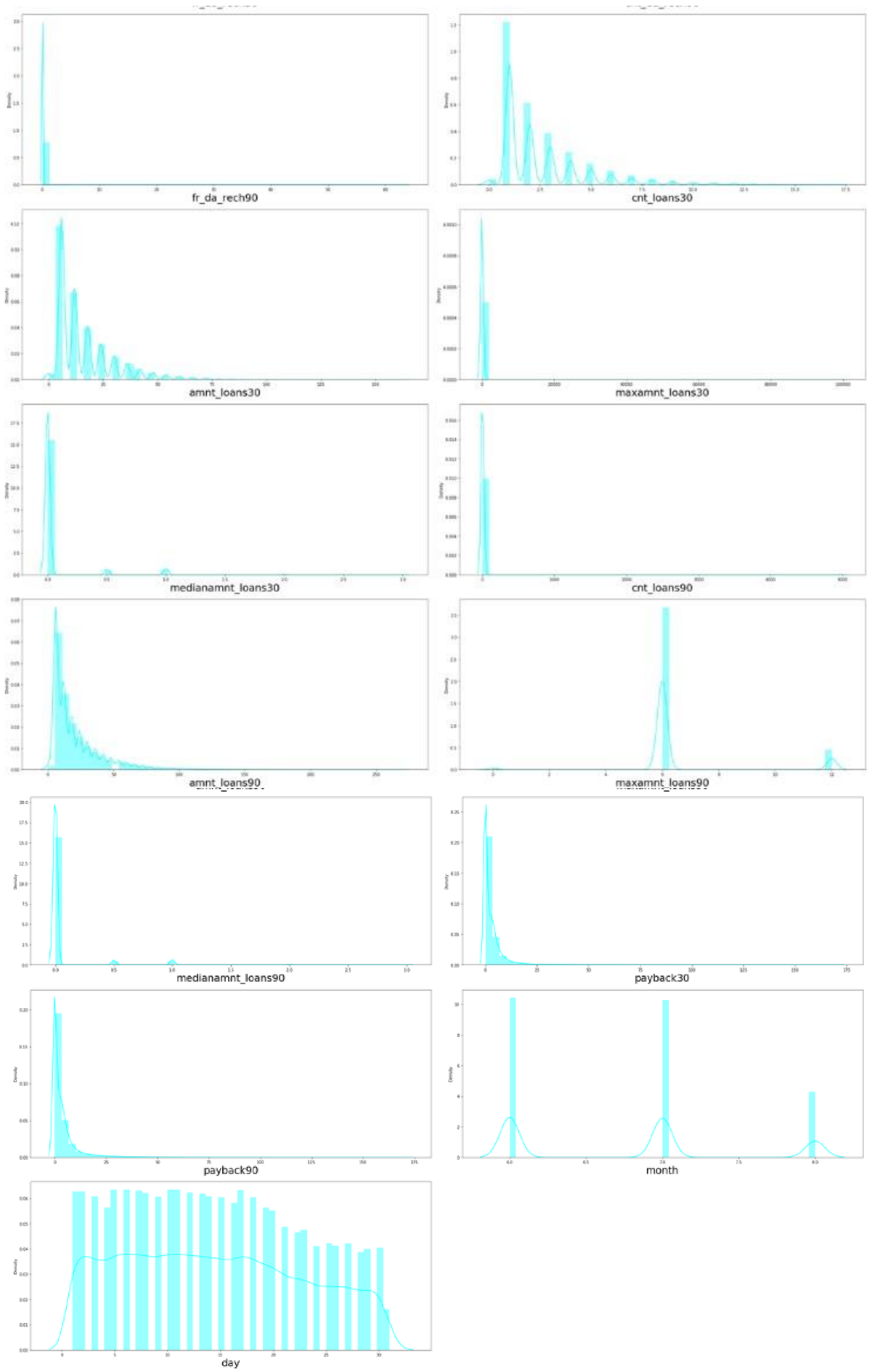
medianamnt_ma_rech90

m •Ninnrr- dr.*I h9rPhd!90

cnt_da_rech30

fr da rech30

cnt da rech90



From the above distribution, we can clearly see that there are few ordinal variables, few discrete variables and most part with continuous variable.

In order to be a good dataset, we assume that all the continuous variables follow normal distribution. However I can see that the continuous columns are skewed and we will have to control skewness to make data follow normal distribution.

Skewness coefficient:

```
new_data.skew()
```

```
label                    -2.183891
aon                       0.945558
daily_decr30              1.971144
daily_decr90              2.000623
rental30                  2.200720
rental90                  2.118841
last_rech_date_ma         3.036533
last_rech_date_da        14.771100
last_rech_amt_ma          3.186359
cnt_ma_rech30             1.222543
fr_ma_rech30             14.726795
sumamnt_ma_rech30         3.168908
medianamnt_ma_rech30      3.130302
medianmarechprebal30     14.847538
cnt_ma_rech90             1.751959
fr_ma_rech90              2.215620
sumamnt_ma_rech90         2.636287
medianamnt_ma_rech90      3.392253
medianmarechprebal90     46.596659
cnt_da_rech30            54.081252
fr_da_rech30             14.693527
cnt_da_rech90            24.793890
fr_da_rech90             29.662829
cnt_loans30               1.819690
amnt_loans30              1.952425
maxamnt_loans30          17.786668
medianamnt_loans30        4.500744
cnt_loans90              16.588551
amnt_loans90              2.354312
maxamnt_loans90           1.858944
medianamnt_loans90        4.809960
payback30                 8.197174
payback90                 6.842662
month                     0.415086
day                       0.182975
dtype: float64
```

It is assumed that the skewness co-efficient within the range of -0.5 to +0.5 is acceptable. Proceeding with the same assumption to get the skewness under control.

In order to perform the same we are using power transformation using ‘yeo-johnson’ method and cube-root transformation on the entire dataset excluding the target variable.

Once performed, most of the skewness are under control except few and we are proceeding with the model building assuming that outliers is not the cause of the skewness in the dataset.

Skewness co-efficient post transformation:

```
x.skew()
```

```
aon                     0.068897
daily_decr30            0.321136
daily_decr90            0.342516
rental30                0.370647
rental90                0.331566
last_rech_date_ma       0.641854
last_rech_date_da       9.414281
last_rech_amt_ma       -0.158959
cnt_ma_rech30          -0.013846
fr_ma_rech30            8.649814
sumamnt_ma_rech30       0.132940
medianamnt_ma_rech30   -0.062924
medianmarechprebal30    1.826726
cnt_ma_rech90           0.158388
fr_ma_rech90            0.372789
sumamnt_ma_rech90       0.138575
medianamnt_ma_rech90   -0.078383
medianmarechprebal90    0.544309
cnt_da_rech30           8.242077
fr_da_rech30           13.905260
cnt_da_rech90           5.983559
fr_da_rech90           15.896629
cnt_loans30            -0.025962
amnt_loans30            0.337138
maxamnt_loans30         3.134176
medianamnt_loans30      3.417175
cnt_loans90             0.832249
amnt_loans90            0.218679
maxamnt_loans90         2.288051
medianamnt_loans90      3.712997
payback30               0.318990
payback90               0.257613
month                  -0.283212
day                    -0.033709
dtype: float64
```

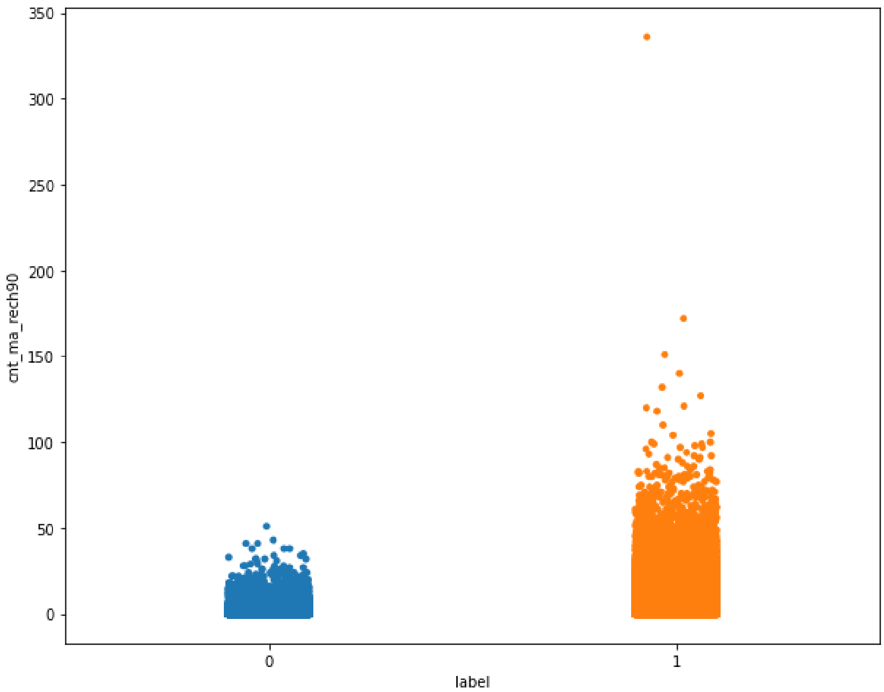
Let’s understand the correlation between the dependent and independent variables, based on which we can determine the major factors that are contributing for loan repayments.

```
data_corr = dataset.corr()
data_corr['label'].sort_values(ascending = False)
```

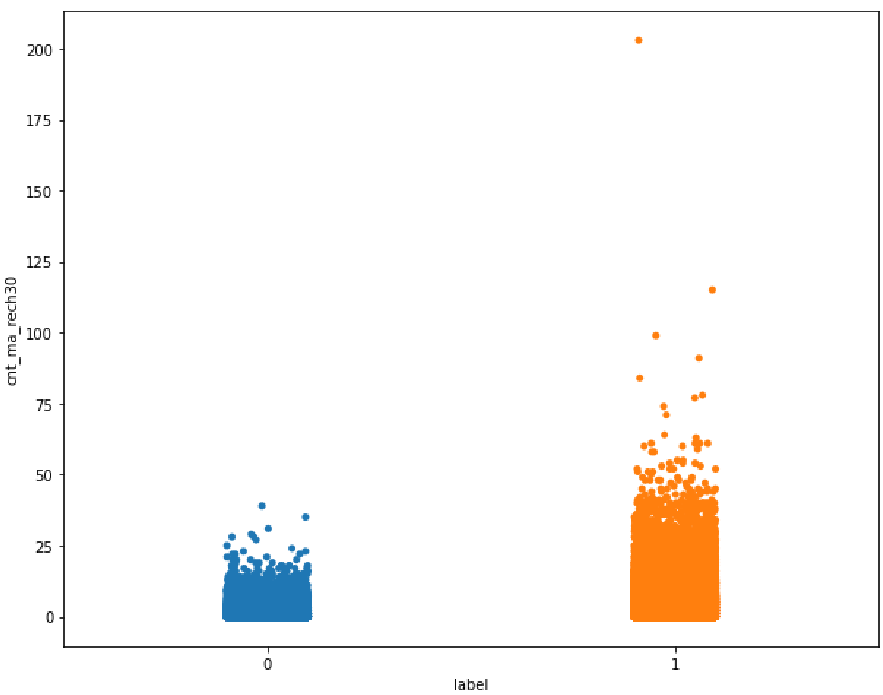
```
label                   1.000000
cnt_ma_rech30           0.237331
cnt_ma_rech90           0.236392
sumamnt_ma_rech90       0.205793
sumamnt_ma_rech30       0.202828
amnt_loans90            0.199788
amnt_loans30            0.197272
cnt_loans30             0.196283
daily_decr30            0.168298
daily_decr90            0.166150
month                   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
day                     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
year                         NaN
Name: label, dtype: float64
```

Although there is no high correlation between the features and the target variable, we can see that there is a maximum correlation of 0.24 between cnt_ma_rech30 and the target variable. The lowest correlation is between maxamnt_loans30 and label (0.0002).

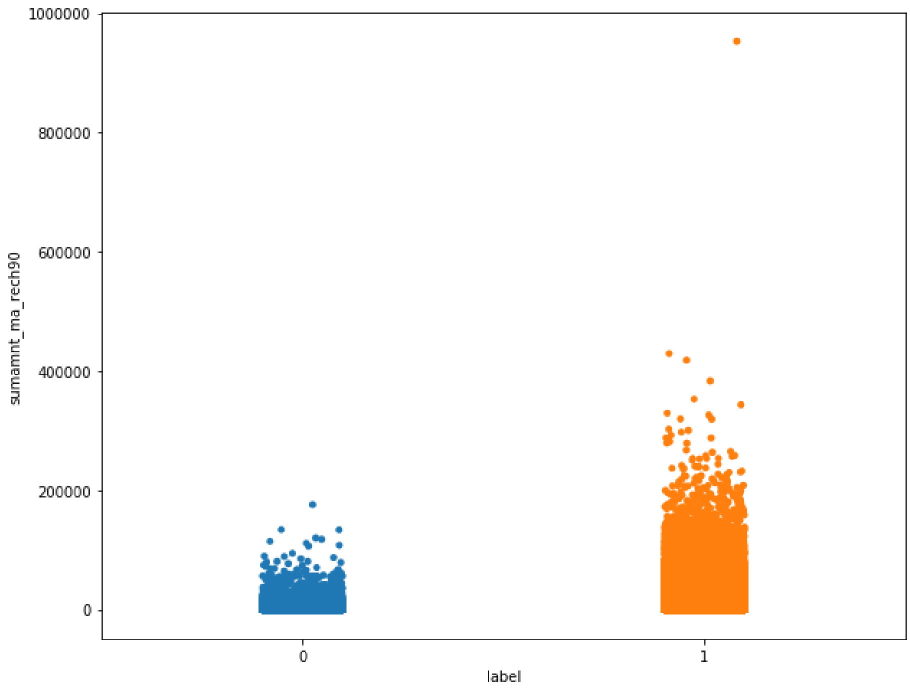
Let’s proceed with visualizing some of the highly correlated variable with the target variable and we are using strip-plot to visualize the same.



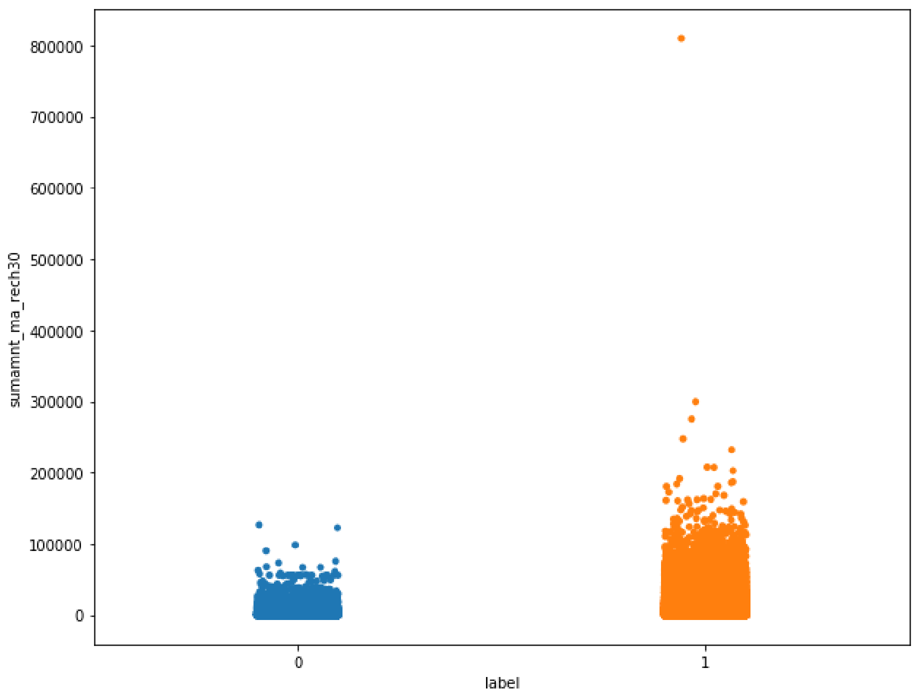
We can see that the customers who repaid the loan made recharges more than 100 times and all the customers who didn’t repay the loan recharged less than 50 times in 90 days



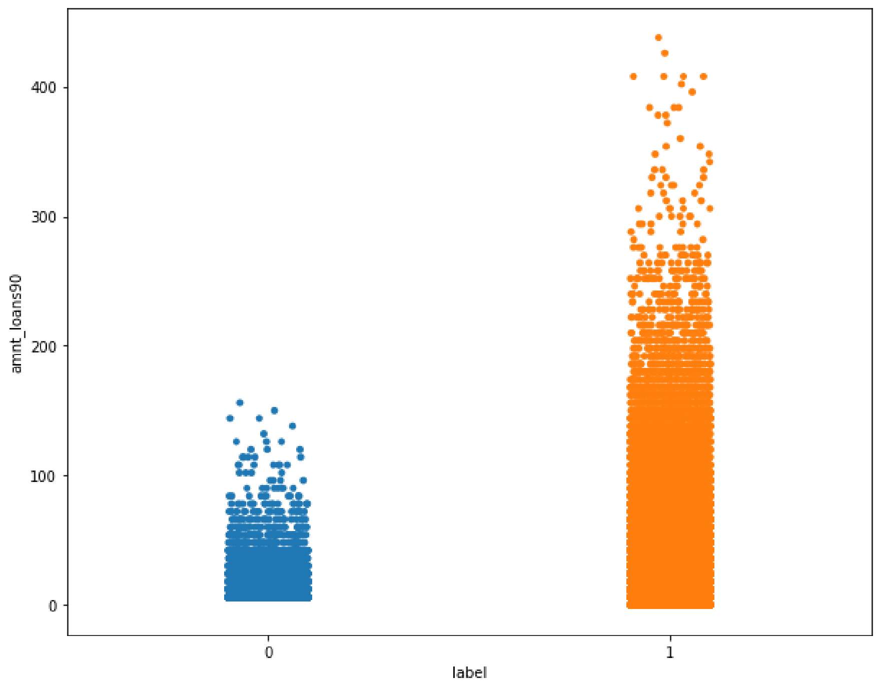
We can say that the customers who didn’t repay the loan, recharged the main account less than 25 times.



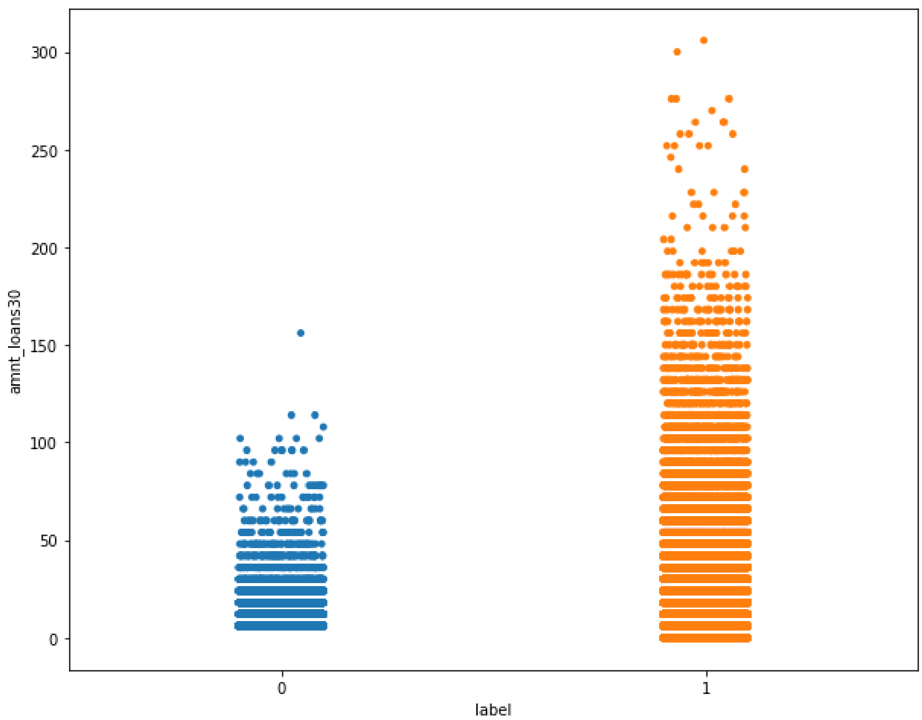
From the above observation, I can say that the customers who didn't repay the loan recharged their main account less than 100000 Indonesian over the past 90 days



From the above observation, I can say that the customers who didn't repay the loan recharged their main account less than 50000 Indonesian over the past 30 days

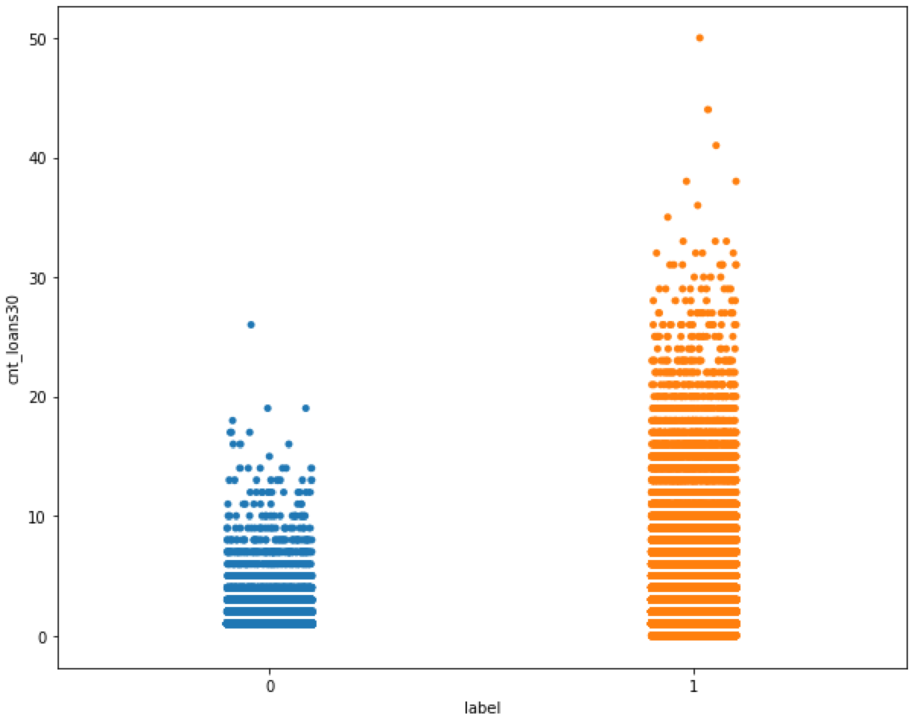


Upon reviewing, we can say that the customers didn't repay the loan, took less than 100 Indonesian rupiah as loans over the 90 day period.

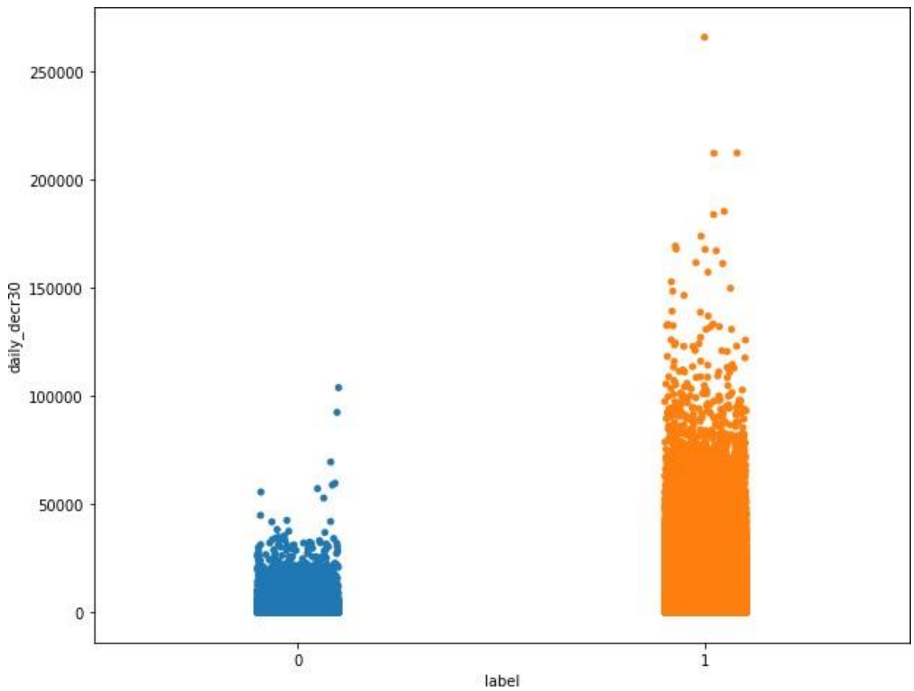


When we look at the 30 day period, customers who took less than 100 Indonesian rupiah as loans didn't repay the loan.

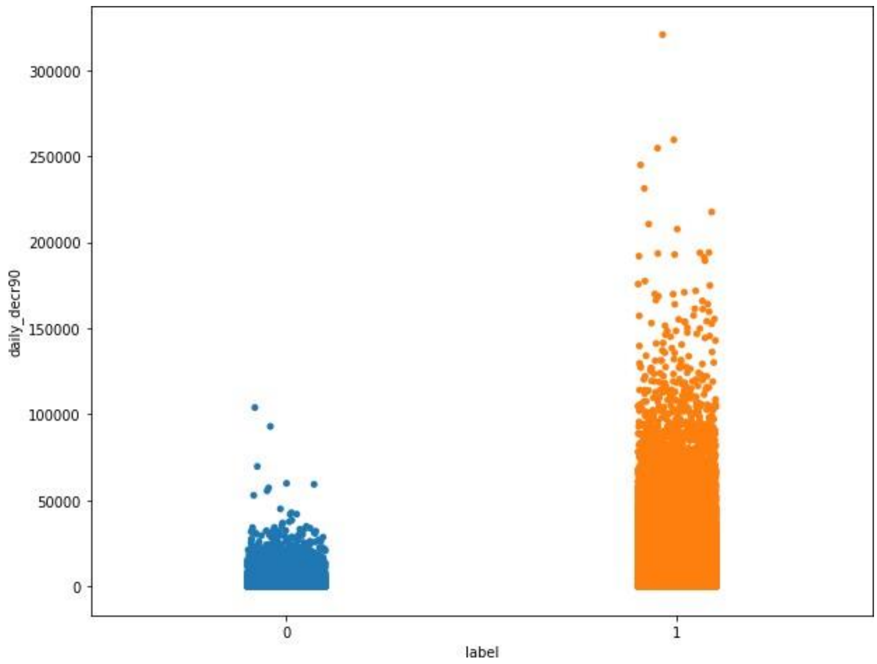
In both of the above cases, we can also say that the lower number of loans were recorded as a result of customers not repaying the amount in time. Hence lesser amount



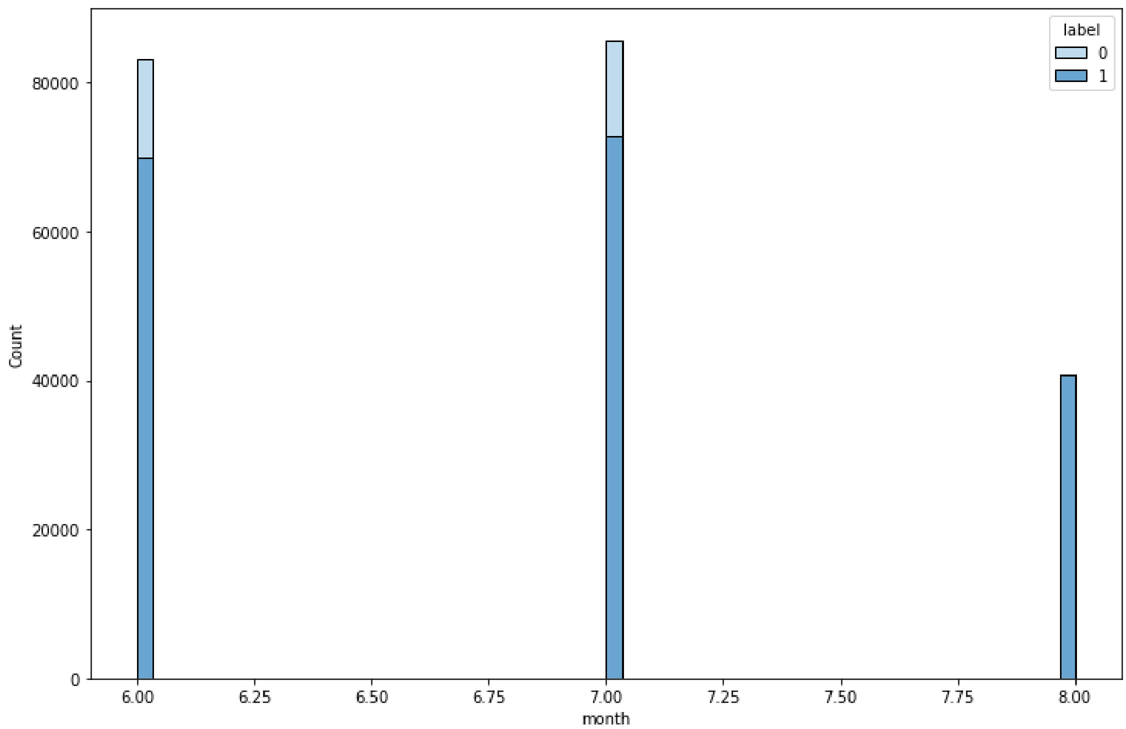
Here, we can also say that, the lesser the number of loans taken, higher the chance of defaulting the loan. The above figure suggests that the customers didn't repay when they took less than 15 loans over the period of 30 days.



The above figure suggests that the customers who didn't repay the loan spent less than 50000 Indonesian rupiah on an average over the past 30 days.



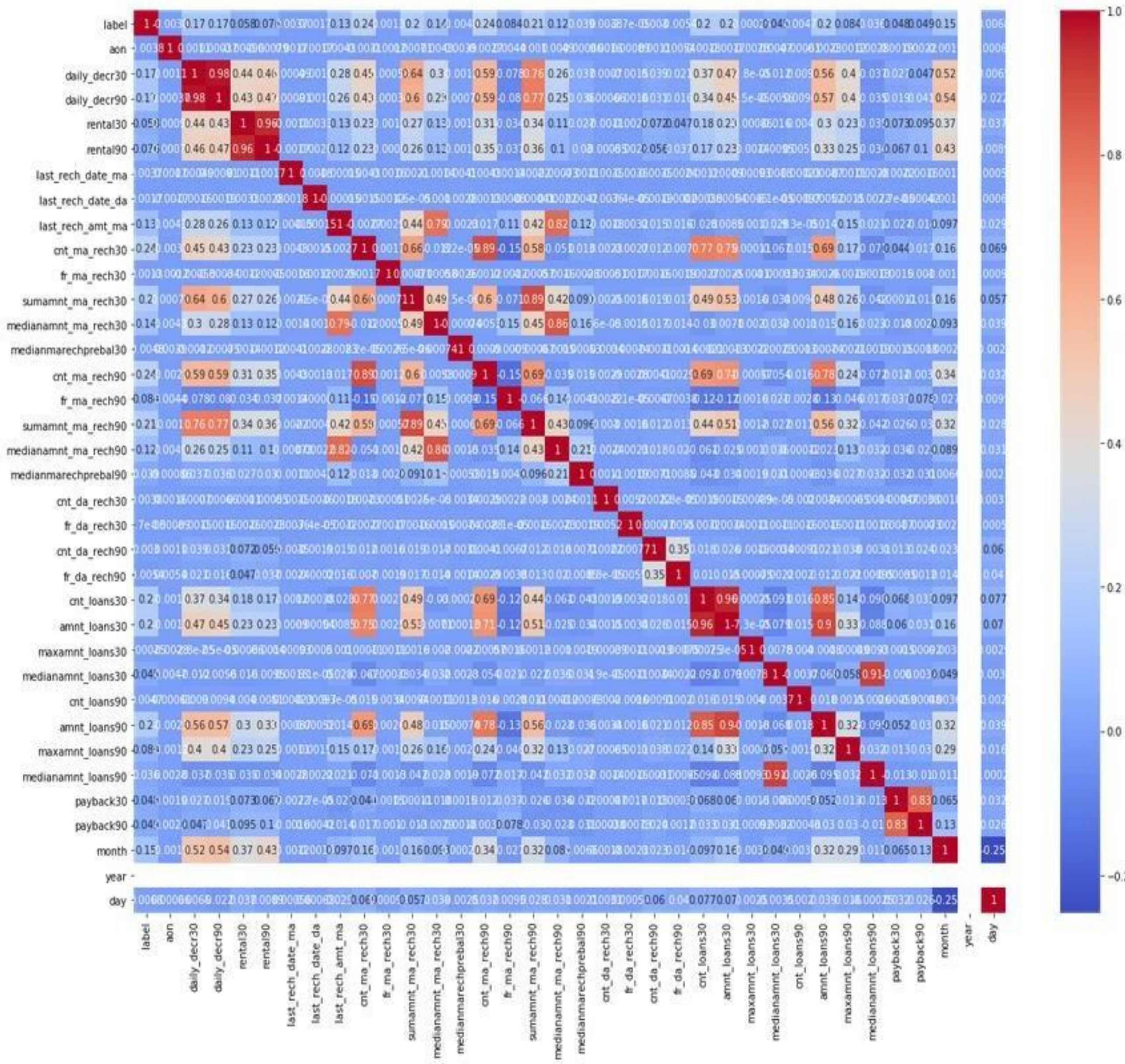
The above figure suggests that the customers who didn't repay the loan spent less than 35000 Indonesian rupiah on an average over the past 90 days.



The above figure suggests that the customers who took the loan in the month of August didn't default the loan when compared to other months.

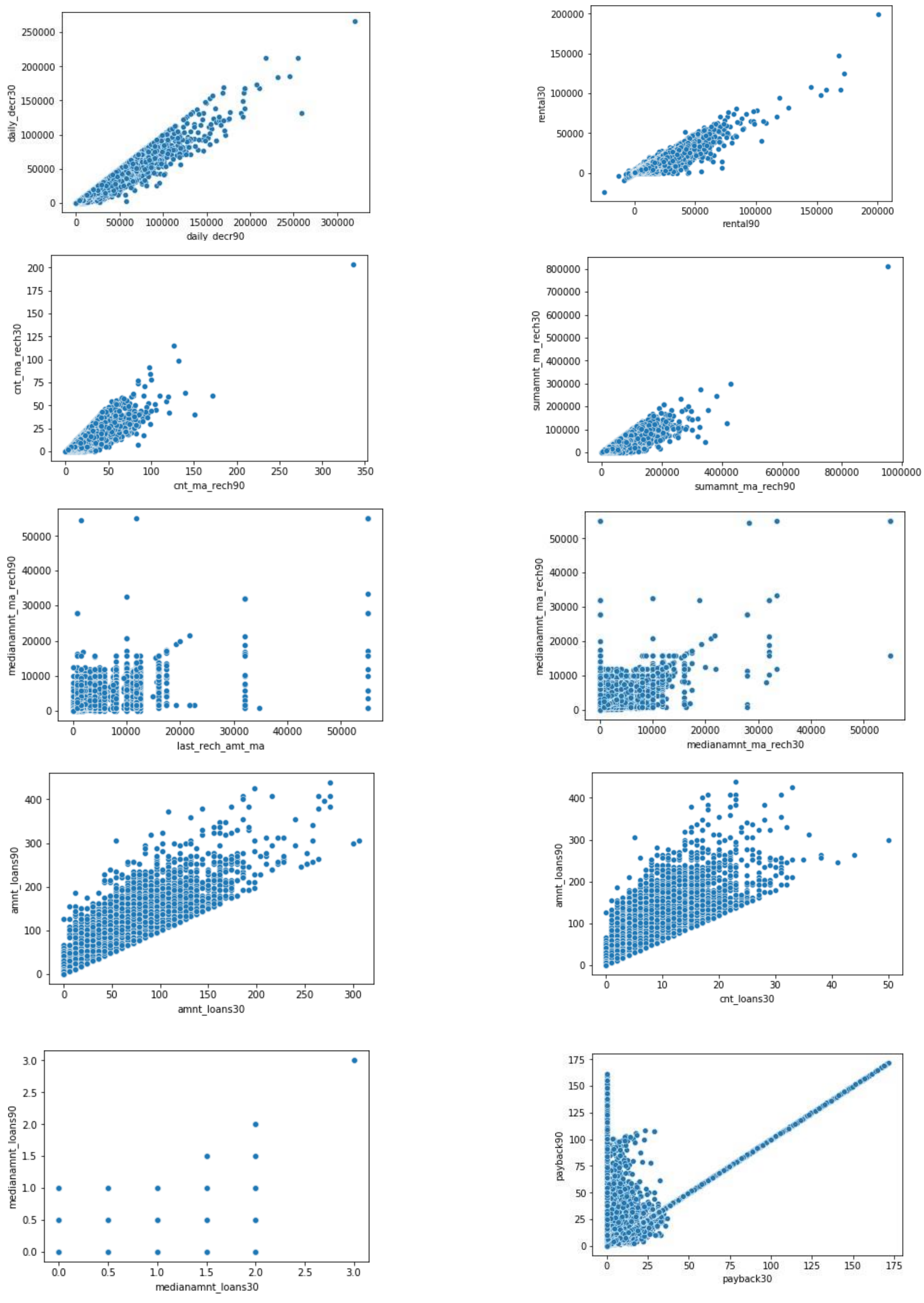
Now that we have analysed the correlation between the dependent and independent variables, we can move forward in analysing the multi-collinearity.

We can use the correlation table which is plotted using a heat map, from the seaborn library.



From the heat-map I can see that there are lot of independent variables are correlated and I can see multi-collinearity in the dataset. However since the business problem is to predict the outcome (whether the customers are going to default the loan or not), let’s proceed with model building assuming that the multi- collinearity won’t affect the prediction.

Visualizing the correlated independent variable with greater than 0.8 correlation coefficient.



Assumptions:

1. Although there is skewness present in some of the variables post transformation, we assume that this is not because of the presence of outliers, however it is the shape of the data itself
2. We assume that all the variables follow normal distribution
3. The multi-collinearity in the dataset will not affect the prediction.
4. Scaling the data in order to keep all the variables with in the desired numerical range.

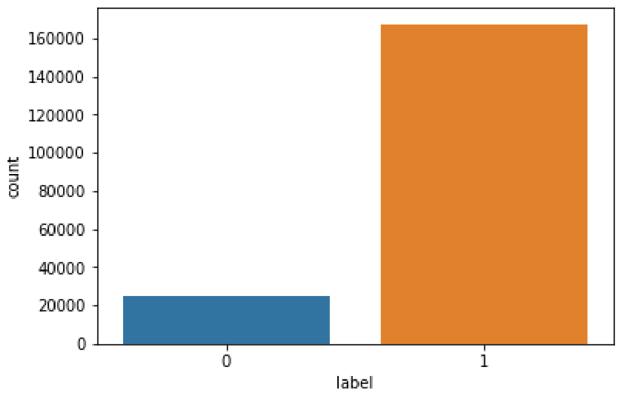
Hardware and Software Requirements and Tools Used:

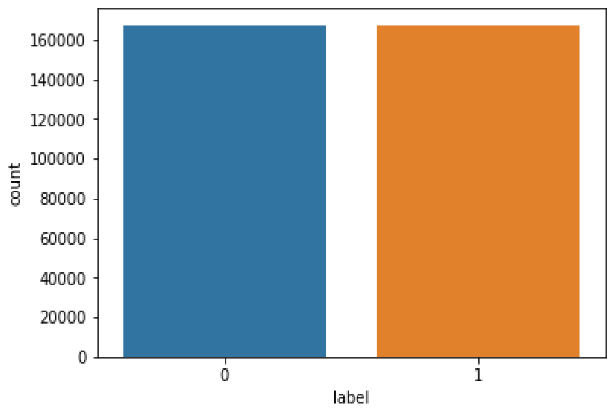
1. Python version 3
2. Jupyter interactive notebook
3. Windows 10 professional
4. Sci-kit learn Library
5. Sci-py Library
6. Seaborn Library
7. Matplotlib Library
8. Intel-core i3 processor
9. 4GB RAM and 500 MB ROM
10. Snipping tool

Model/s Development and Evaluation

We identified that there was class imbalance in the dataset, before proceeding with the model building, I'm balancing the class using SMOTE over-sampling technique. This is necessary because, if the balancing is not done the model tends to predict the majority class better and the minority class will not be accurately predicted.

Below plots compare the class count before and after performing SMOTE.





Further, before build the model we will have to split the data to test and train. The best possible way to split the data is by finding the best random state to split and the benefit is that we can control over fitting up to certain extent before even building the model.

We are trying to match the accuracy score of the training data set and the test dataset, which ever split (**random state**) satisfies the condition (**accuracy score of training dataset = accuracy score of testing dataset**). We'll take the same random state to split the dataset and build the model.

We are using a simple for loop to achieve the same.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, roc_auc_score
rs = 0
for i in range(0,2000):
    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 1999
```

Now, I can say that the best random state for the split is 1999 and we will be splitting the dataset 77% train and 33% test with the random state 1999.

Since the dataset is large to my system configurations, ensemble techniques will be efficient although I'm testing the results with the below algorithms.

1. Logistic Regression
2. Random Forest Classifier
3. Extra Trees Classifier
4. XG Boost Classifier
5. K-Nearest Neighbors Classifier

In order to test the model, I'm using accuracy score, AUC ROC score and F1 score, further in order to verify the model's fit, I'm using cross val score to identify the best model.

Model 1: Logistic Regression

The first Machine Learning model I'm using to predict the outcome of the loan is Linear Regression, this gives us with better understanding of the dataset and it's a simple model to build.

```
lg = LogisticRegression()
lg.fit(x_train,y_train)
lg_pred = lg.predict(x_test)
lg_score = accuracy_score(y_test,lg_pred)
lg_score
```

```
0.7715167085813537
```

```
print(classification_report(y_test, lg_pred))
```

```
              precision    recall  f1-score   support

           0       0.76      0.80      0.78     55410
           1       0.79      0.74      0.76     55132

    accuracy                           0.77    110542
   macro avg       0.77      0.77      0.77    110542
weighted avg       0.77      0.77      0.77    110542
```

```
print(roc_auc_score(y_test, lg_pred))
```

```
0.7714443043705657
```

Using the Logistic Regression, we were able to get the accuracy score of 0.77, the F1 score (Balanced precision and Recall) of 0.77 and the AUC score of0.77.

The AUC score (Area Under the Curve) simply means that the Logistic Regression model is able to distinguish between classes up to 77% of data.

Further, I'm verifying the fit using cross_val_score with cross validation of 5 and see that the model is not overfitting.

```
from sklearn.model_selection import cross_val_score
cv = cross_val_score(lg, x_over,y_over,cv = 5)
cv = cv.mean()
cv
```

```
0.7711016400480433
```

Model 2: Random Forest Classifier

As we discussed before, I'm using ensemble techniques to predict the outcome and I'm using Random Forest Classifier here.

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(x_train,y_train)
rf_pred = rf.predict(x_test)
rf_score = accuracy_score(y_test,rf_pred)
rf_score
```

```
0.9494762171844185
```

```
print(classification_report(y_test, rf_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.95      0.95     55410
           1       0.95      0.95      0.95     55132

    accuracy                           0.95    110542
   macro avg       0.95      0.95      0.95    110542
weighted avg       0.95      0.95      0.95    110542
```

```
print(roc_auc_score(y_test, rf_pred))
```

```
0.9494705967478514
```

Here I can see that the Random Forest Classifier, is predicting the outcome with 95% accuracy, 95% F1-score, and the measure of the ability of a classifier to distinguish between classes (AUC) is also 95%.

Further, I'm verifying the fit using cross_val_score with cross validation of 5 and see that the model is not overfitting.

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

```
0.946491430160233
```

Model 3: Extra Trees Classifier

The Extra Trees algorithm works by creating a large number of unpruned decision trees from the training dataset. Predictions are made by averaging the prediction of the decision trees in the case of regression or using majority voting in the case of classification.

```
from sklearn.ensemble import ExtraTreesClassifier
et = ExtraTreesClassifier()
et.fit(x_train,y_train)
et_pred = et.predict(x_test)
et_score = accuracy_score(y_test,et_pred)
et_score
```

```
0.9565142660708147
```

```
print(classification_report(y_test, et_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.97      0.96     55410
           1       0.97      0.94      0.96     55132

    accuracy                           0.96    110542
   macro avg       0.96      0.96      0.96    110542
weighted avg       0.96      0.96      0.96    110542
```

```
print(roc_auc_score(y_test, et_pred))
```

```
0.9564793873712457
```

Here I can see that the Extra Trees Classifier, is predicting the outcome with 96% accuracy, 96% F1-score, and the measure of the ability of a classifier to distinguish between classes (AUC) is also 96%.

Further, I'm verifying the fit using cross_val_score with cross validation of 5 and see that the model is not overfitting.

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

```
0.962582183424308
```

Model 4: XG Boost Classifier

Extreme Gradient Boosting Algorithm. Gradient boosting refers to a class of ensemble machine learning algorithms that can be used for classification or regression predictive modelling problems. Trees are added one at a time to the ensemble and fit to correct the prediction errors made by prior models.

```
from xgboost import XGBClassifier
xgb = XGBClassifier(eval_metric = 'logloss')
xgb.fit(x_train,y_train)
xgb_pred = xgb.predict(x_test)
xgb_score = accuracy_score(y_test,xgb_pred)
xgb_score
```

```
0.9457762660346294
```

```
print(classification_report(y_test, xgb_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95     55410
           1       0.94      0.96      0.95     55132

    accuracy                           0.95    110542
   macro avg       0.95      0.95      0.95    110542
weighted avg       0.95      0.95      0.95    110542
```

```
print(roc_auc_score(y_test,xgb_pred))
```

```
0.9458015857674219
```

Here I can see that the XG Boost Classifier, is predicting the outcome with 95% accuracy, 95% F1-score, and the measure of the ability of a classifier to distinguish between classes (AUC) is also 95%.

Further, I'm verifying the fit using cross_val_score with cross validation of 5 and see that the model is not overfitting.

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

```
0.9323590454991673
```

Model 5: K-Neighbors Classifier

K-Neighbors Classifier is a nearest-neighbor classification model in which you can alter both the distance metric and the number of nearest neighbors. Because a ClassificationKNN classifier stores training data, you can use the model to compute resubstitution predictions

```
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
```

```
0.8673173997213729
```

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

```
              precision    recall  f1-score   support

           0       0.80      0.97      0.88     55410
           1       0.96      0.76      0.85     55132

    accuracy                           0.87    110542
   macro avg       0.88      0.87      0.87    110542
weighted avg       0.88      0.87      0.87    110542
```

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

```
0.8670539048394729
```

Here I can see that the K-Neighbors Classifier, is predicting the outcome with 87% accuracy, 87% F1-score, and the measure of the ability of a classifier to distinguish between classes (AUC) is also 87%.

Further, I'm verifying the fit using cross_val_score with cross validation of 5 and see that the model is not overfitting.

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

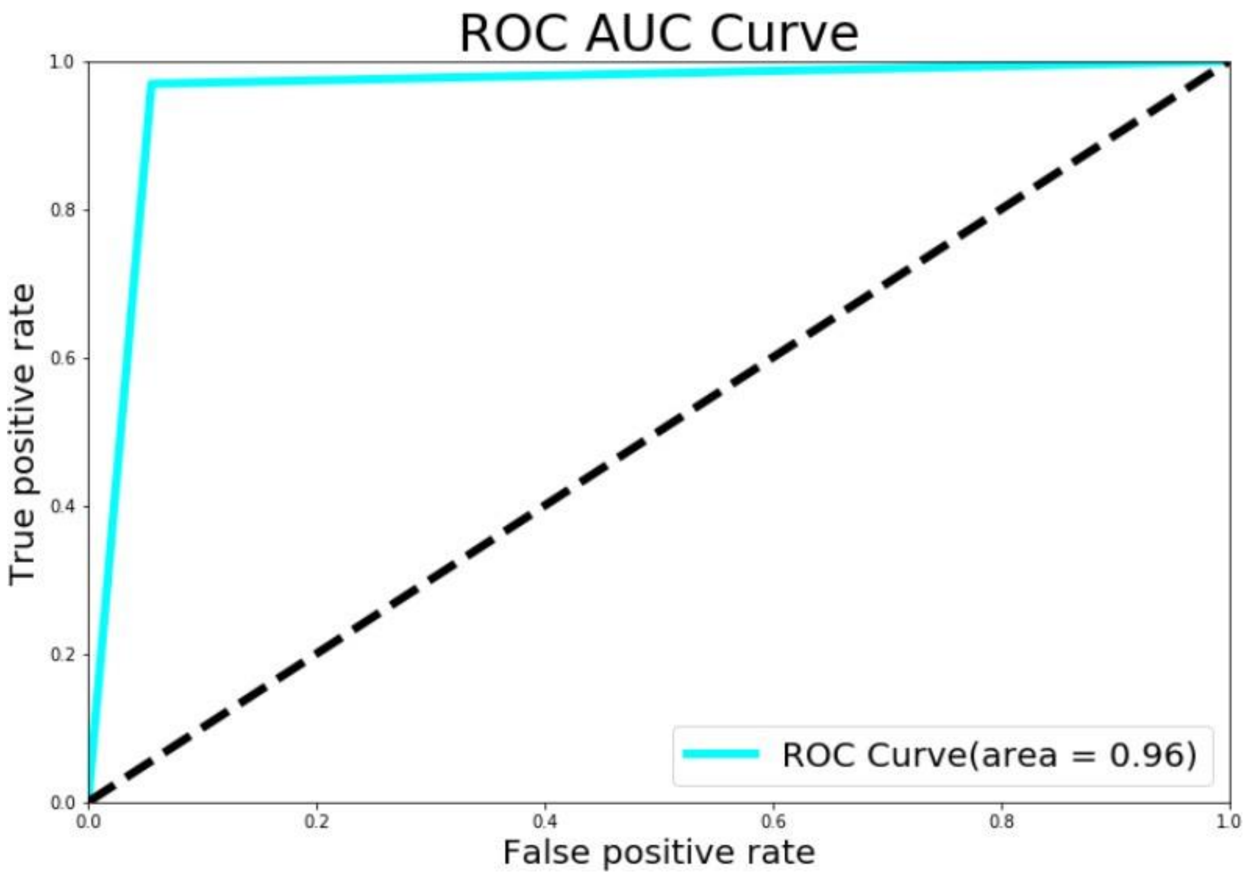
```
0.876796409278179
```

Finding the best model by subtracting the model's accuracy with the cross validation scores.

```
model =[lg_score, rf_score, et_score,xgb_score,knn_score]
cross_val = [cv,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	0.771517	0.771102	0.000415
1	0.949476	0.946491	0.002985
2	0.956514	0.962582	-0.006068
3	0.945776	0.932359	0.013417
4	0.867317	0.876796	-0.009479

Here I can see that the Extra Trees model has highest accuracy and the highest F1- score. Further the model is also not overfitting.



Therefore, Extra Trees is the best Machine Learning model for the dataset. Therefore proceeding with the Hyper Parameter Tuning.

```
params ={'n_estimators':[100,200,300,400],
        'criterion':['gini','entropy'],
        'max_depth':[23,25,29,31],
        'min_samples_split':[2,3,4,5],
        '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': [13, 15, 16, 17],
                         'min_samples_split': [2, 3, 4, 5, 6],
                         'n_estimators': [100, 200, 300, 400]})
```

```
final.best_params_
```

```
{'bootstrap': False,
 'criterion': 'gini',
 'max_depth': 17,
 'min_samples_split': 2,
 'n_estimators': 300}
```

```
final_rf = ExtraTreesClassifier(bootstrap = False, criterion= 'gini', max_depth = 32, min_samples_split = 3, n_estimators =60
0)
final_rf.fit(x_train,y_train)
final_pred = final_rf.predict(x_test)
final_score = accuracy_score(y_test,final_pred)
final_score
```

```
0.9414792567530893
```


Performing the hyper parameter tuning doesn't improve the scores, therefore finalizing the base Extra Trees model because it is providing the accuracy score of 0.96.

The Key Metric used to finalize the model was AUC_ROC_CURVE, cross_val_score and the F1-Score. And the Extra Trees is the best model at predicting the Micro- Credit Defaulters.

CONCLUSION

We have successfully built a model using multiple models and found that the Extra Trees Classifier model.

Below are the details of the model's metrics predicting the dataset

1. Average precision of 0.96
2. Average recall of 0.96
3. F1 Score is 0.96
4. The ability of a classifier to distinguish between classes (AUC) is also 0.96. Below are the

major contributing variables for the prediction:

- 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
- month - 0.154949
- medianamnt_ma_rech30 - 0.141490
- last_rech_amt_ma - 0.131804
- medianamnt_ma_rech90 - 0.120855

You can view the same from the visualizations on the correlation of independent variable over dependent variable (target)

As we can see from the boxplot, I couldn't remove all the outliers yet since the data is expensive, I have to proceed with the dataset with outliers

Further, I couldn't get skewness under control for few variables through couple of transformation techniques, yet I have proceeded with building the model.

Looking at the heatmap for correlation, I could see there were few variables which were correlated with each other, yet I have not removed any variable based on their correlation because multi-collinearity will not affect prediction.

Limitations of this work and Scope for Future Work.

1. Due to the presence of lot of outlier, we are unsure whether the model is going to perform well to a completely new dataset.
2. Due to a class imbalance, we had to rebalance the class 0. This might also have some effect while trying to predict the outcome with completely new data
3. During data-collection, we could place limits on few continuous variables, where the customer could enter data within a limit because the variables like age on the network cannot be more than certain months

Other than these above limitations, I couldn't find more scope for improvement.