FLIP ROBO

MICRO CREDIT DEFAULTER MODEL

Submitted by:

ANCHAL AWASTHI

ACKNOWLEDGMENT

Gratitude takes three forms-"A feeling from heart, an expression in words and a giving in return". We take this opportunity to express our feelings.

I express my gracious gratitude to our project guide Mr. **Harsh Ayush,** SME of Flip Robo Technologies, for his valuable guidance and assistance. I am very thankful for his encouragement and inspiration that made project successful.

I express my gracious gratitude to our Trainer **Dr. Deepika Sharma,** Training Head of DataTrained - Data Analytics, Data Science Online Training, Bengaluru, for her valuable guidance and assistance throughout the PG Program. I am very thankful for her encouragement and inspiration that made project successful.

INTRODUCTION

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 MFI 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 dataset of a 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 (in Indonesian Rupiah), while, for the loan amount of 10 (in Indonesian Rupiah), the payback amount should be 12(in Indonesian Rupiah).

The sample data is provided to us from our client database. In order to improve the selection of customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers.

So, we are building 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 paid i.e. Non- defaulter, while, Label '0' indicates that the loan has not been paid i.e. defaulter.)

Conceptual Background of the Domain Problem

As we are working on the Micro Finance Services dataset we can easily understand that the data belongs to the Banking and Finance services which will eventually involve several Financial, Banking, Statistical, and Technical terms in the dataset.

Review of Literature

This section discusses about the reported work carried out in various fields of Micro Finance & Telecommunication sectors.

Robert J. Kauffman et Al., describes Information and communication technology (ICT) is an important driver in the microfinance industry. Microfinance providers, both non-profit microfinance institutions (MFIs) and for-profit banks, provide financial services to the poor, and are critical for economic development in developing nations. As the industry matures, MFIs face a competitive environment, forcing them to balance the goals of outreach and sustainability. ICT may be the instigator of this new environment and the potential solution to MFI survivability.

Farhana Ferdousi et Al., describes the proponents of inclusive financial growth believe that giving relatively larger loans to the non-poor or near-poor entrepreneurs is the response of the microfinance institutions (MFIs) toward the demand of a existing and potential clients. But opponents are more likely to consider response such as mission drift by the MFIs. Therefore, this study attempts to measure the effectiveness of such microenterprise loans on increasing entrepreneurs' incomes and innovation. Findings suggest that larger loans increase income, but less innovative business practice might threaten such income.

Saon Ray et Al., describes digital technologies are rapidly making inroads into all realms of human life and the microfinance sector is one such area. Ongoing innovations in digital technology are contributing to reshaping its operational models, governance structures, its risk profile, industry networks and dominant practices. Innovation and integration of digital technology are both important because of their potential to promote development by spurring innovation, improving efficiency and increasing inclusion. The mechanisms - innovation, inclusion and efficiency – have been integral to microfinance operations in the past, and how innovations in digital technology may be yet another opportunity for microfinance institutions to promote development.

Motivation for the Problem Undertaken

During the current modernization of the technologies ant their implement in several Industrial sectors has resulted in many new outcomes which are challenging and profitable for the growth. In this project the client has 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. As this project deals with 2 different sectors of the most renowned and well-established firm's i.e. Finance & Telecommunication. Working on this project not only helped me in learning of the model building but it also helped me to understand the different methodology involved in the MFI and Telecommunication sector. The thirst and thrive to learn new things, methodologies, concepts and problem-solving techniques motivated me to complete this project.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

We are building 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 paid i.e., Non- defaulter while Label ‘0’ indicates that the loan has not been paid i.e., defaulter. The given statement clearly shows that project is a classification problem as it has a binary 0 & 1 output in the Label column. This can be used to build the Classification models for the given dataset.

Classification is a form of data analysis that extracts models describing data classes. A classifier, or classification model, predicts categorical labels (classes). Numeric prediction models continuous-valued functions. Classification and numeric prediction are the two major types of prediction problems. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. Outcomes are labels that can be applied to a dataset.

The different Mathematical/Analytical models that are used in this project are as below.

1. Logistic regression - is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).

2. Gaussian Naive Bayes - algorithm is a special type of NB algorithm. It's specifically used when the features have continuous values. It's also assumed that all the features are following a gaussian distribution i.e., normal distribution. A Gaussian classifier is a generative approach in the sense that it attempts to model class posterior as well as input class-conditional distribution. Therefore, we can generate new samples in input space with a Gaussian classifier.

3. Decision Tree Classifier - Decision tree learning is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree to go from observations about an item to conclusions about the item's target value.

4. Random Forest Classifier - Random Forest uses multiple decision trees as base learning models in the dataset. Random forest is a meta estimator that fits several classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting in the dataset. The main concept of Random Forest is to combine multiple decision trees in determining the result rather than relying on individual decision trees.

5. AdaBoost Classifier - AdaBoost is best used to boost the performance of decision trees on binary classification problems. AdaBoost was originally called AdaBoost. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers.

6. Gradient boosting classifiers - Gradient boosting classifiers are a group of machines learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting.

7. A Bagging classifier - is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Bagging is used when the goal is to reduce the variance of a decision tree classifier.

8. Extra Trees Classifier - is an ensemble learning method fundamentally based on decision trees. Extra Trees Classifier, like Random Forest, randomizes certain decisions and subsets of data to minimize over-learning from the data and over fitting. This class implements a meta estimator that fits several randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

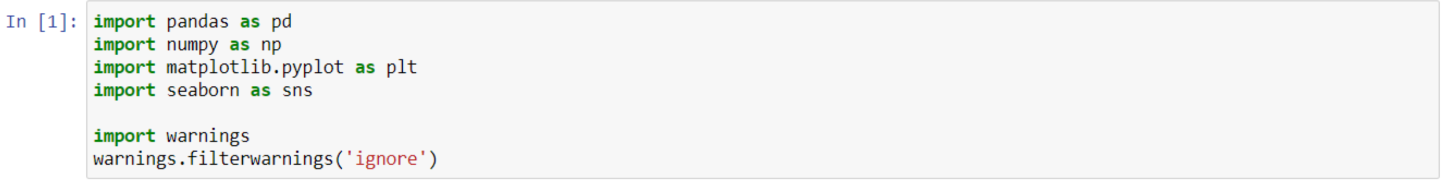
Data Sources and their formats

Feature Name	Feature Description	Datatypes
label	target variable in the dataset {1:success, 0:failure}	Int64
aon	age on cellular network in days	Object
daily_decr30	Daily amount spent from main account, averaged over last 30 days	Float64
daily_decr90	Daily amount spent from main account, averaged over last 90 days	Float64
rental30	Average main account balance over last 30 days	Float64
rental90	Average main account balance over last 90 days	Float64
last_rech_date_ma	Number of days till last recharge of main account	Float64
last_rech_date_da	Number of days till last recharge of data account	Float64
last_rech_amt_ma	Amount of last recharge of main account (in Indonesian Rupiah)	Int64
cnt_ma_rech30	Number of times main account got recharged in last 30 days	Int64
fr_ma_rech30	Frequency of main account recharged in last 30 days	Float64
sumamnt_ma_rech30	Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)	Float64
medianamnt_ma_rech30	Median of number of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)	Float64
medianmarechprebal30	Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)	Float64
cnt_ma_rech90	Number of times main account got recharged in last 90 days	Int64
fr_ma_rech90	Frequency of main account recharged in last 90 days	Int64
sumamnt_ma_rech90	Total amount of recharge in main account over last 90 days (in Indonasian Rupiah)	Int64
medianamnt_ma_rech90	Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah)	Float64
medianmarechprebal90	Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah)	Float64

cnt_da_rech30	Number of times data account got recharged in last 30 days	Float64
fr_da_rech30	Frequency of data account recharged in last 30 days	Float64
cnt_da_rech90	Number of times data account got recharged in last 90 days	Int64
fr_da_rech90	Frequency of data account recharged in last 90 days	Int64
cnt_loans30	Number of loans taken by user in last 30 days	Int64
amnt_loans30	Total amount of loans taken by user in last 30 days.	Int64
maxamnt_loans30	maximum amount of loan taken by the user in last 30 days	Float64
medianamnt_loans30	Median of amounts of loan taken by the user in last 30 days	Float64
cnt_loans90	Number of loans taken by user in last 90 days	Float64
amnt_loans90	Total amount of loans taken by user in last 90 days	Int64
maxamnt_loans90	maximum amount of loan taken by the user in last 90 days	Int64
medianamnt_loans90	Median of amounts of loan taken by the user in last 90 days	Float64
payback30	Average payback time in days over last 30 days	Float64
payback90	Average payback time in days over last 90 days	Float64
day	telecom circle	Object
month	date	Object

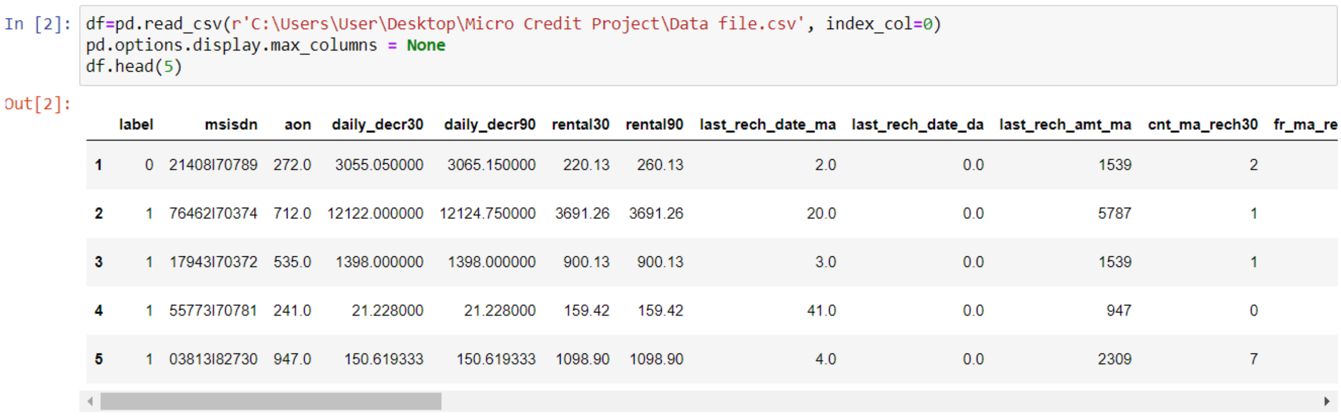
The given dataset is in CSV format, now let's load the dataset and do the analysis.

Importing Libraries



```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns

        import warnings
        warnings.filterwarnings('ignore')
```

Loading the dataset

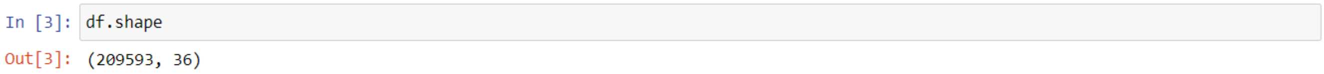


```
In [2]: df=pd.read_csv(r'C:\Users\User\Desktop\Micro Credit Project\Data file.csv', index_col=0)
        pd.options.display.max_columns = None
        df.head(5)
```

Out[2]:

	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_re
1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	2	
2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	5787	1	
3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539	1	
4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	947	0	
5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309	7	

#Checking the shape of dataset

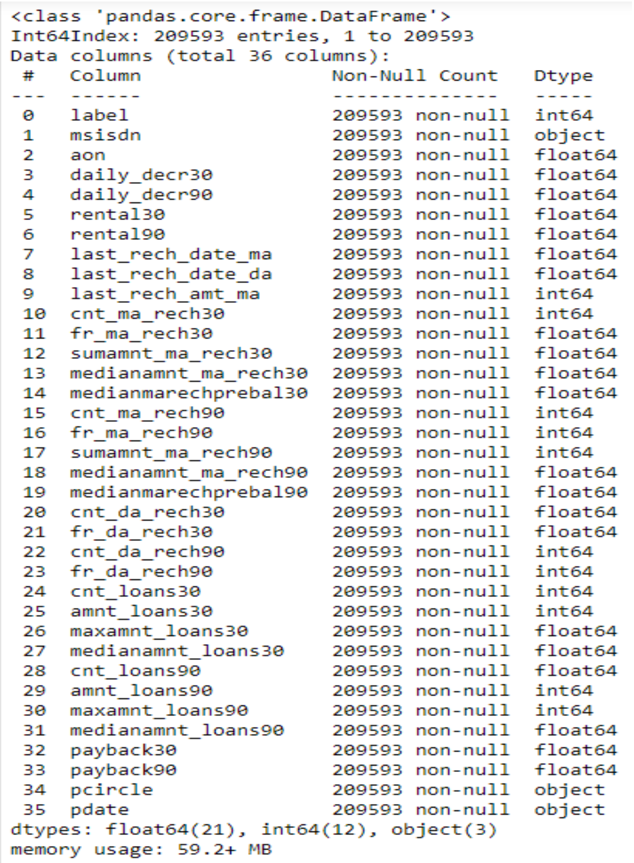


```
In [3]: df.shape
Out[3]: (209593, 36)
```

The dataset has 36 attributes and a total of 2,09,593 of rows of data

Checking the data type of dataset

```
In [4]: df.info()
```



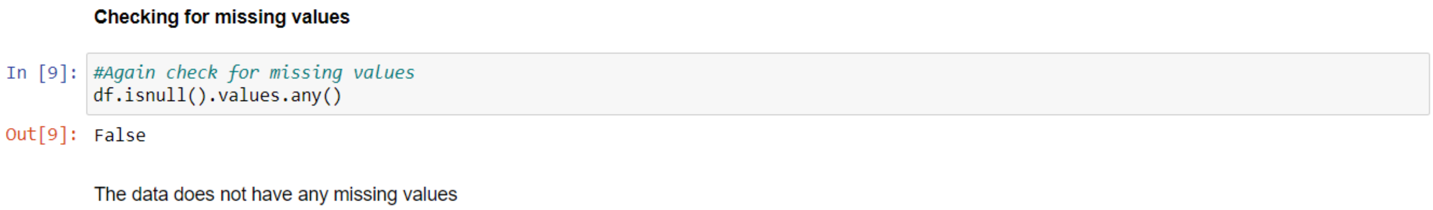
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 209593 entries, 1 to 209593
Data columns (total 36 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   label                 209593 non-null  int64
 1   msisdn                209593 non-null  object
 2   aon                   209593 non-null  float64
 3   daily_decr30          209593 non-null  float64
 4   daily_decr90          209593 non-null  float64
 5   rental30              209593 non-null  float64
 6   rental90              209593 non-null  float64
 7   last_rech_date_ma     209593 non-null  float64
 8   last_rech_date_da     209593 non-null  float64
 9   last_rech_amt_ma      209593 non-null  int64
 10  cnt_ma_rech30         209593 non-null  int64
 11  fr_ma_rech30          209593 non-null  float64
 12  sumamnt_ma_rech30     209593 non-null  float64
 13  medianamnt_ma_rech30  209593 non-null  float64
 14  medianmarechprebal30  209593 non-null  float64
 15  cnt_ma_rech90         209593 non-null  int64
 16  fr_ma_rech90          209593 non-null  int64
 17  sumamnt_ma_rech90     209593 non-null  int64
 18  medianamnt_ma_rech90  209593 non-null  float64
 19  medianmarechprebal90  209593 non-null  float64
 20  cnt_da_rech30         209593 non-null  float64
 21  fr_da_rech30          209593 non-null  float64
 22  cnt_da_rech90         209593 non-null  int64
 23  fr_da_rech90          209593 non-null  int64
 24  cnt_loans30           209593 non-null  int64
 25  amnt_loans30          209593 non-null  int64
 26  maxamnt_loans30       209593 non-null  float64
 27  medianamnt_loans30    209593 non-null  float64
 28  cnt_loans90           209593 non-null  float64
 29  amnt_loans90          209593 non-null  int64
 30  maxamnt_loans90       209593 non-null  int64
 31  medianamnt_loans90    209593 non-null  float64
 32  payback30             209593 non-null  float64
 33  payback90             209593 non-null  float64
 34  pcircle               209593 non-null  object
 35  pdate                 209593 non-null  object
dtypes: float64(21), int64(12), object(3)
memory usage: 59.2+ MB
```

Observations

1. In the "Micro Finance Service" dataset we have 21 float64 data type, 12 int64 data type and 3 object type column.

2. The msisdn, pcircle, pdate columns are object data type columns.

Tracing Null values in data frame.



Checking for missing values

```
In [9]: #Again check for missing values
        df.isnull().values.any()

Out[9]: False
```

The data does not have any missing values

Observation

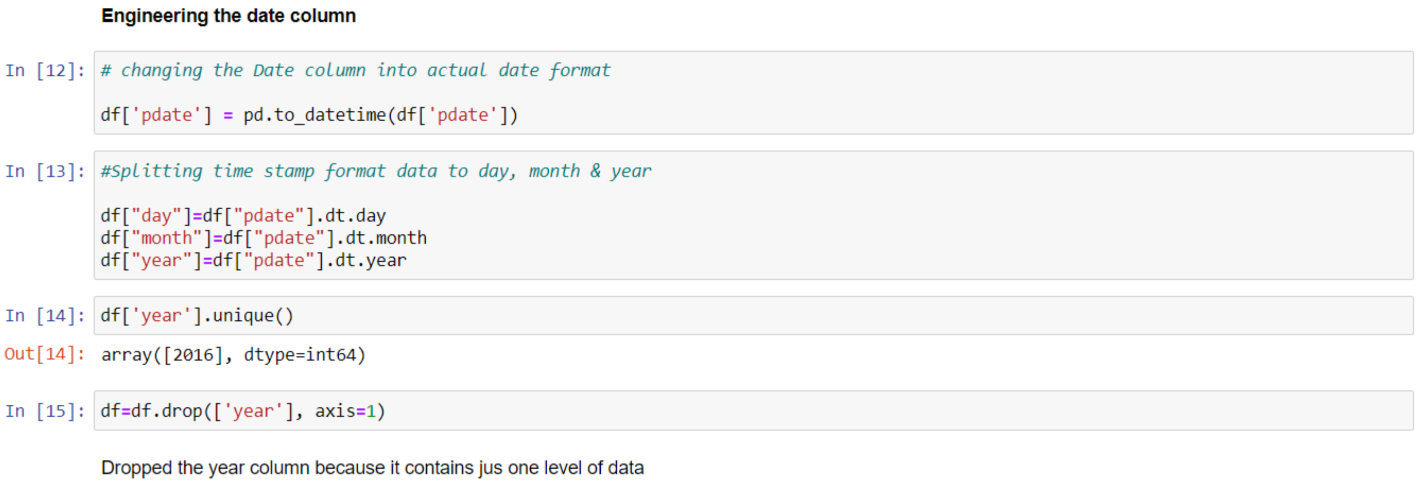
Here we see there are no null values present in the given dataset.

Data Pre-processing

Data pre-processing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. Data pre-processing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we pre-process our data before feeding it into our model.

In the given dataset “msisdn, pcircle, pdate” are object data type columns which should be preprocessed before passing it to the model training.

Let us change the pdate column into actual date, time format.



Engineering the date column

In [12]:
```
# changing the Date column into actual date format

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

In [13]:
```
#Splitting time stamp format data to day, month & year

df["day"]=df["pdate"].dt.day
df["month"]=df["pdate"].dt.month
df["year"]=df["pdate"].dt.year
```

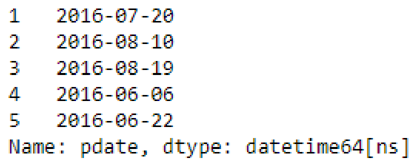
In [14]:
```
df['year'].unique()
```

Out[14]:
```
array([2016], dtype=int64)
```

In [15]:
```
df=df.drop(['year'], axis=1)
```

Dropped the year column because it contains jus one level of data

Output



```
1    2016-07-20
2    2016-08-10
3    2016-08-19
4    2016-06-06
5    2016-06-22
Name: pdate, dtype: datetime64[ns]
```

Now dropping the columns which does not have any significance in the model building and also dropping the similar or duplicate columns which are already present in the dataset.

Now dropping the 'year' column from the dataset as the year is same for all the cases & it doesn't provide any significance for further analysis. Dropping msisdn column as it is just working as ID number/unique telecom number which does not have any effect on the dataset. We should also drop the pdate column as we have already converted the pdate into datetime64 and added date, month separately to the data frame. Dropping the pcircle column as this is a telecom circle column which is same across all the cases.

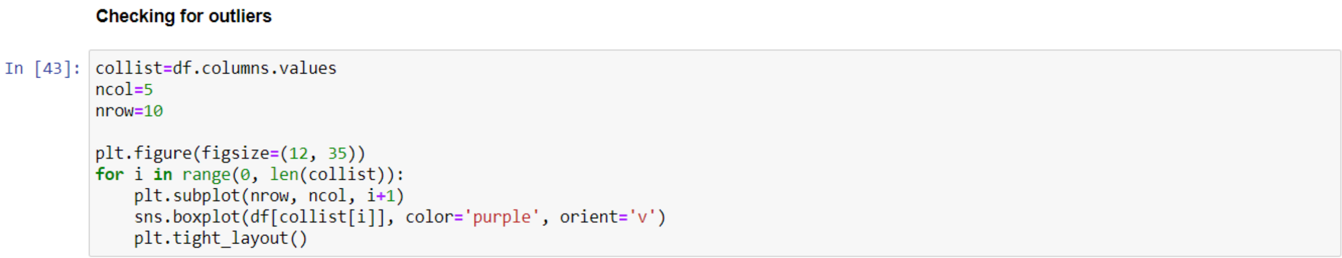
As a pre processing step after the deleting the unnecessary columns from data frame we can see that the number of columns are reduced to 35 which are valuable and provide sufficient inputs to the model building.

Outliers

An outlier is a data point in a data set which is distant or far from all other observations available. It is a data point which lies outside the overall distribution which is available in the dataset. In statistics, an outlier is an observation point that is distant from other observations.

A box plot is a method or a process for graphically representing groups of numerical data through their quartiles. Outliers may also be plotted as an individual point. If there is an outlier it will plotted as point in box plot but other numerical data will be grouped together and displayed as boxes in the diagram. In most cases a threshold of 3 or -3 is used i.e if the Z- score value is higher than or less than 3 or -3 respectively, that particular data point will be identified as outlier.

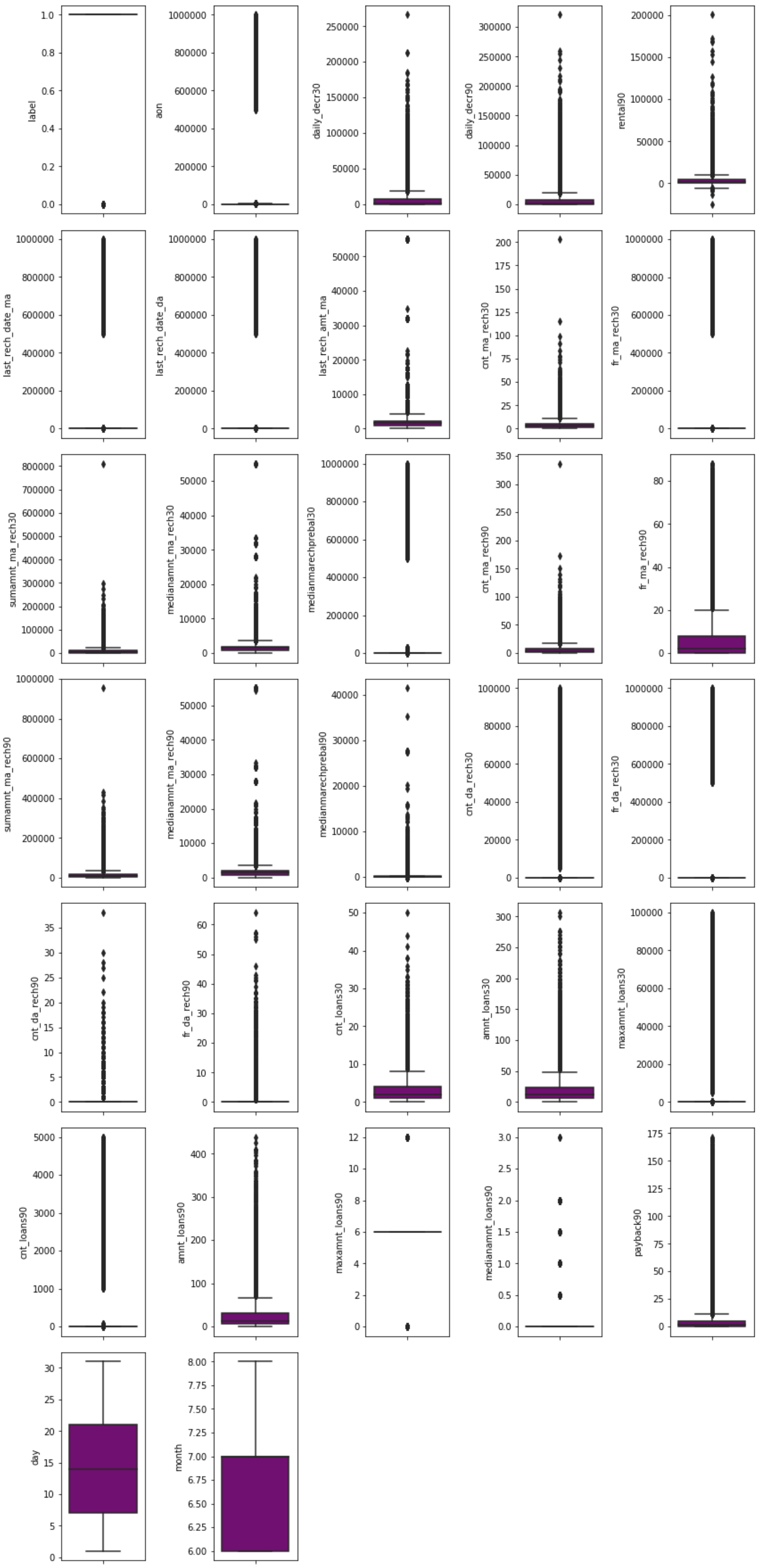
Plotting outliers in the Dataset



Checking for outliers

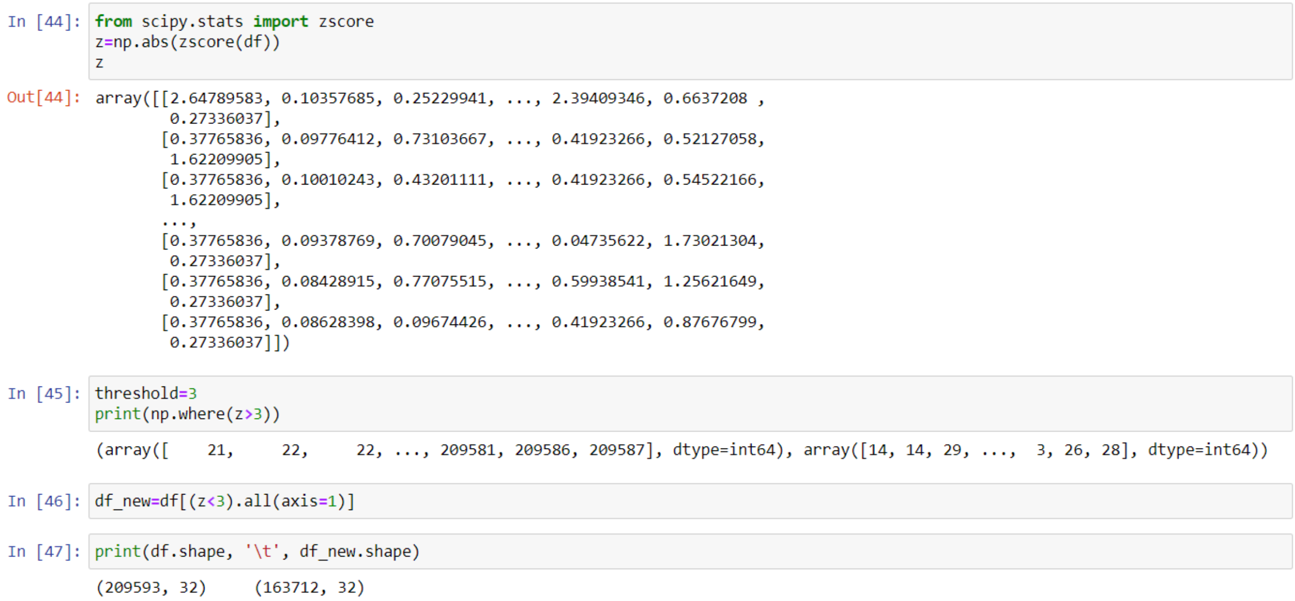
```
In [43]: collist=df.columns.values
         ncol=5
         nrow=10

         plt.figure(figsize=(12, 35))
         for i in range(0, len(collist)):
             plt.subplot(nrow, ncol, i+1)
             sns.boxplot(df[collist[i]], color='purple', orient='v')
             plt.tight_layout()
```

In the box plot graph we can see that columns 'daily_decr30', 'daily_decr90', 'rental30', 'rental90', 'last_rech_amt_ma', 'cnt_ma_rech30', 'fr_ma_rech30', 'sumamnt_ma_rech30', 'medianamnt_ma_rech30', 'medianmarechprebal30', 'cnt_ma_rech90', 'sumamnt_ma_rech90', 'medianamnt_ma_rech90', 'medianmarechprebal90', 'cnt_da_rech30', 'cnt_da_rech90', 'fr_da_rech90', 'amnt_loans30', 'medianamnt_loans30', 'cnt_loans90', 'amnt_loans90', 'medianamnt_loans90' contains in it.


label
0.0 0.2 0.4 0.6 0.8 1.0
aon
0 200000 400000 600000 800000 1000000
daily_decr30
0 50000 100000 150000 200000 250000
daily_decr90
0 50000 100000 150000 200000 250000 300000
rental90
0 50000 100000 150000 200000
last_rech_date_ma
0 200000 400000 600000 800000 1000000
last_rech_date_da
0 200000 400000 600000 800000 1000000
last_rech_amt_ma
0 10000 20000 30000 40000 50000
cnt_ma_rech30
0 25 50 75 100 125 150 175 200
fr_ma_rech30
0 200000 400000 600000 800000 1000000
sumamnt_ma_rech30
0 100000 200000 300000 400000 500000 600000 700000 800000
medianamnt_ma_rech30
0 10000 20000 30000 40000 50000
medianmarechprebal30
0 200000 400000 600000 800000 1000000
cnt_ma_rech90
0 50 100 150 200 250 300 350
fr_ma_rech90
0 20 40 60 80
sumamnt_ma_rech90
0 200000 400000 600000 800000 1000000
medianamnt_ma_rech90
0 10000 20000 30000 40000 50000
medianmarechprebal90
0 10000 20000 30000 40000
cnt_da_rech30
0 20000 40000 60000 80000 100000
fr_da_rech30
0 200000 400000 600000 800000 1000000
cnt_da_rech90
0 5 10 15 20 25 30 35
fr_da_rech90
0 10 20 30 40 50 60
cnt_loans30
0 10 20 30 40 50
amnt_loans30
0 50 100 150 200 250 300
maxamnt_loans30
0 20000 40000 60000 80000 100000
cnt_loans90
0 1000 2000 3000 4000 5000
amnt_loans90
0 100 200 300 400
maxamnt_loans90
0 2 4 6 8 10 12
medianamnt_loans90
0.0 0.5 1.0 1.5 2.0 2.5 3.0
payback90
0 25 50 75 100 125 150 175
day
0 5 10 15 20 25 30
month
6.00 6.25 6.50 6.75 7.00 7.25 7.50 7.75 8.00

Z – Score

A Z-score is a numerical measurement that describes a value's relationship to the mean of a group of values in the dataset. Z-score is measured in terms of standard deviations from the mean.



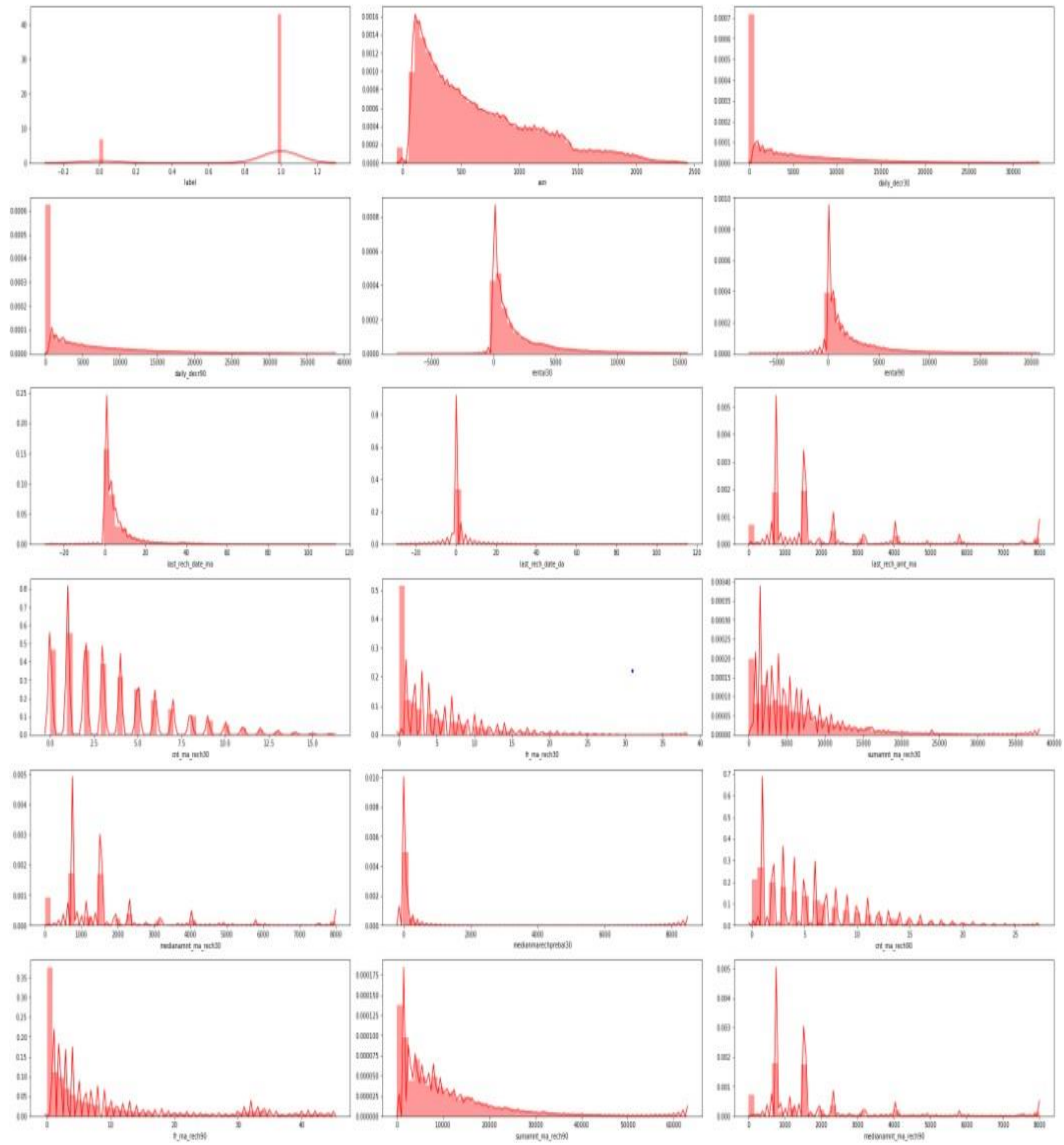
```
In [44]: from scipy.stats import zscore
         z=np.abs(zscore(df))
         z

Out[44]: array([[2.64789583, 0.10357685, 0.25229941, ..., 2.39409346, 0.6637208 ,
                 0.27336037],
                [0.37765836, 0.09776412, 0.73103667, ..., 0.41923266, 0.52127058,
                 1.62209905],
                [0.37765836, 0.10010243, 0.43201111, ..., 0.41923266, 0.54522166,
                 1.62209905],
                ...,
                [0.37765836, 0.09378769, 0.70079045, ..., 0.04735622, 1.73021304,
                 0.27336037],
                [0.37765836, 0.08428915, 0.77075515, ..., 0.59938541, 1.25621649,
                 0.27336037],
                [0.37765836, 0.08628398, 0.09674426, ..., 0.41923266, 0.87676799,
                 0.27336037]])

In [45]: threshold=3
         print(np.where(z>3))

         (array([    21,     22,     22, ..., 209581, 209586, 209587], dtype=int64), array([14, 14, 29, ...,  3, 26, 28], dtype=int64))

In [46]: df_new=df[(z<3).all(axis=1)]

In [47]: print(df.shape, '\t', df_new.shape)

         (209593, 32)     (163712, 32)
```

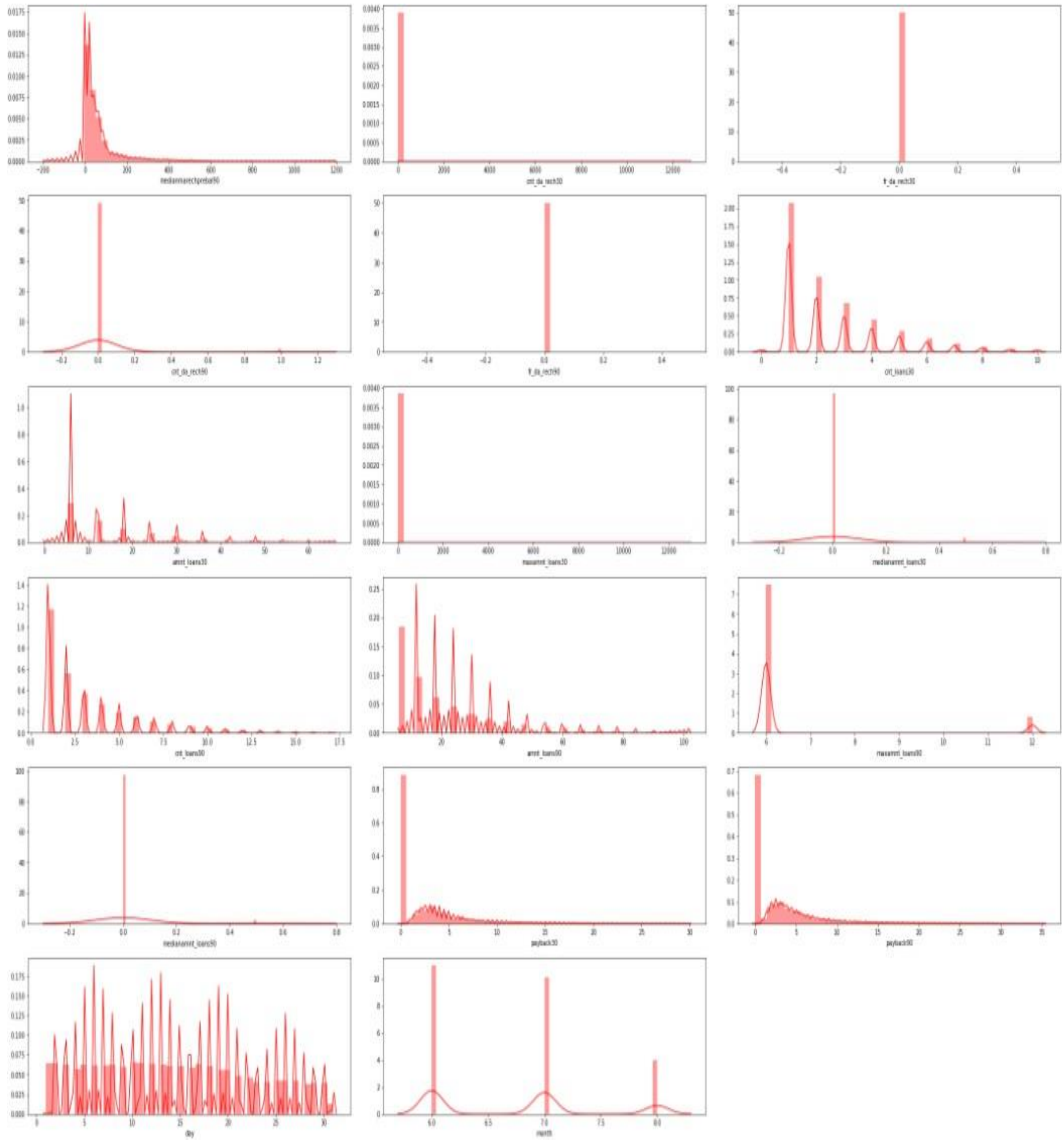
To check distribution of Skewness

Skewness

Skewness refers to distortion or asymmetry in a symmetrical bell curve, or normal distribution in a set of data. Besides positive and negative skew, distributions can also be said to have zero or undefined skew. The skewness value can be positive, zero, negative, or undefined.

Check skewness of dataset with visualization





In the above graphs we can see that columns "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" are right skewed.

Treating Skewness

In the Data Science it is just statistics and many algorithms revolve around the assumption that the data is normalized. So, the more the data is close to normal, the better it is for getting good predictions.

1. Log Transform

Log transformation is a data transformation method in which it replaces each variable x with a log(x). The choice of the logarithm base is usually left up to the analyst and it would depend on the purposes of statistical modeling. The log transformation is, arguably, the most popular among the different types of transformations used to transform skewed data to approximately conform to normality. If the original data follows a log-normal distribution or approximately so, then the log-transformed data follows a normal or near normal distribution.

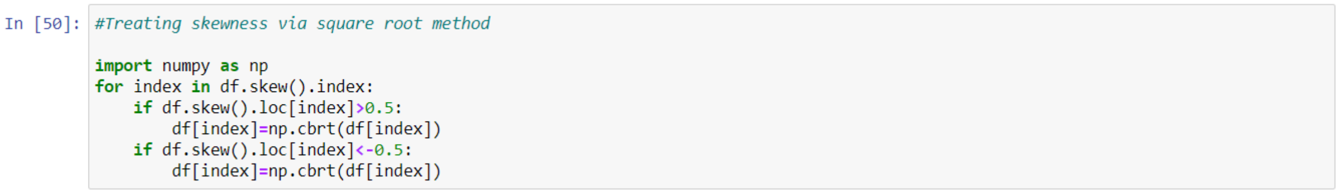
2. Square Root Transform

The square root, x to x^(1/2) = sqrt(x), is a transformation with a moderate effect on distribution shape: it is weaker than the logarithm and the cube root. It is also used for reducing right skewness, and also has the advantage that it can be applied to zero values. So applying a square root transform inflates smaller numbers but stabilises bigger ones. The square root of an area has the units of a length. It is commonly applied to counted data, especially if the values are mostly rather small.

3. Box-Cox Transform

In statistics, a power transform is a family of functions that are applied to create a monotonic transformation of data using power functions. A Box Cox transformation is a transformation of a non-normal dependent variables into a normal shape. This is a useful data transformation technique used to stabilize variance, make the data more normal distribution- like, improve the validity of measures of association such as the Pearson correlation between variables and for other data stabilization procedures.

Treating Skewness in the dataset

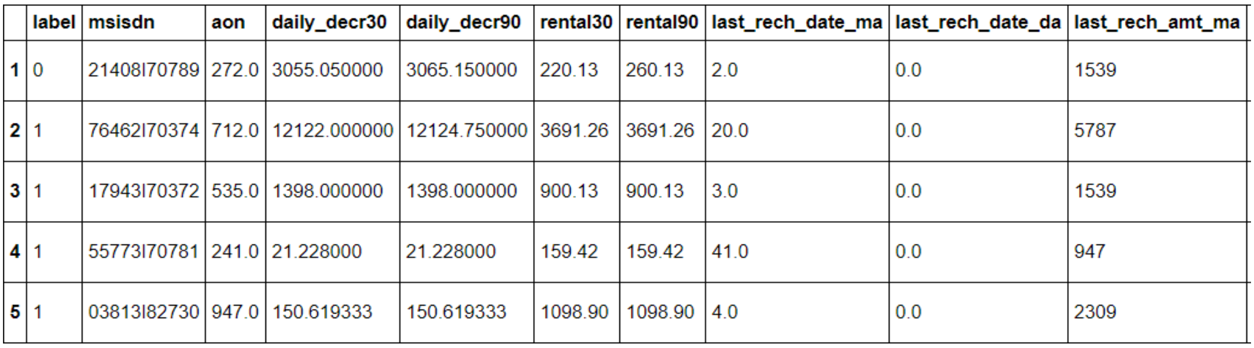


```
In [50]: #Treating skewness via square root method

         import numpy as np
         for index in df.skew().index:
             if df.skew().loc[index]>0.5:
                 df[index]=np.cbrt(df[index])
             if df.skew().loc[index]<-0.5:
                 df[index]=np.cbrt(df[index])
```

Data Inputs- Logic - Output Relationships

The given dataset has 35 columns after pre processing of the data in which the “label” column is an output column. The below analysis describes the relationship behind the data input, its format, the logic in between and the output.

Detailed study of input and output column is as below.

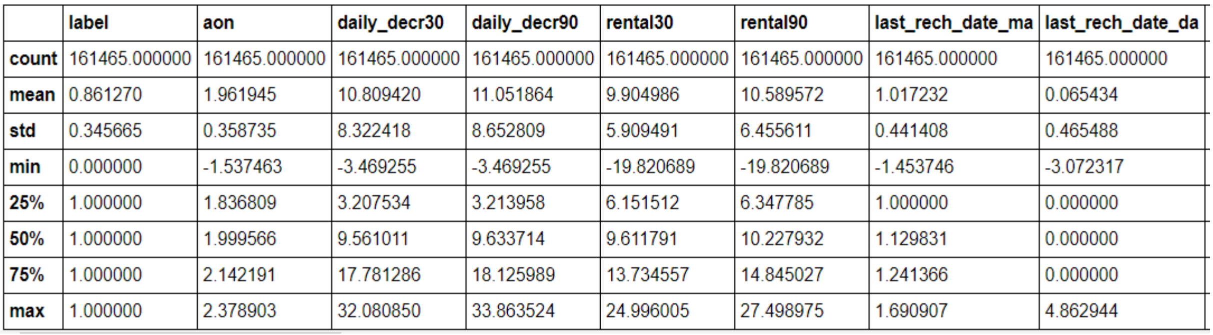


	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma
1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539
2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	5787
3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539
4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	947
5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309

Summary Statistics

In descriptive statistics, summary statistics are used to summarize a set of observations, in order to communicate the largest amount of information as simply as possible. Summary statistics summarize and provide information about your sample data. It tells something about the values in data set. This includes where the average lies and whether the data is skewed.

The describe() function computes a summary of statistics pertaining to the Data Frame columns. This function gives the mean, count, max, standard deviation and IQR values of the dataset in a simple understandable way.



	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da
count	161465.000000	161465.000000	161465.000000	161465.000000	161465.000000	161465.000000	161465.000000	161465.000000
mean	0.861270	1.961945	10.809420	11.051864	9.904986	10.589572	1.017232	0.065434
std	0.345665	0.358735	8.322418	8.652809	5.909491	6.455611	0.441408	0.465488
min	0.000000	-1.537463	-3.469255	-3.469255	-19.820689	-19.820689	-1.453746	-3.072317
25%	1.000000	1.836809	3.207534	3.213958	6.151512	6.347785	1.000000	0.000000
50%	1.000000	1.999566	9.561011	9.633714	9.611791	10.227932	1.129831	0.000000
75%	1.000000	2.142191	17.781286	18.125989	13.734557	14.845027	1.241366	0.000000
max	1.000000	2.378903	32.080850	33.863524	24.996005	27.498975	1.690907	4.862944

Observation

1. The 'label' column is a target variable in the dataset {1:success, 0:failure}

2. Maximum aon (age on cellular network in days) is 999860.755168 and Minimum aon is – 48.000000.

3. Maximum daily_decr30 (Daily amount spent from main account, averaged over last 30 days) is 265926.00 and Min Minimum daily_decr30 is -93.012667

4.Maximum daily_decr90 (Daily amount spent from main account, averaged over last 90 days) is 320630.00 and Min Minimum daily_decr30 is -93.012667

5. Maximum rental30 (Average main account balance over last 30 days) is 198926.110 and Minimum is -23737.140

6. Maximum rental90 (Average main account balance over last 90 days) is 200148.110 and Minimum is -24720.580

7. Maximum cnt_loans30 (Number of loans taken by user in last 30 days) is 50.00 and Minimum is 0

8. Maximum amnt_loans30 (Total amount of loans taken by user in last 30 days) is 306.00 and Minimum is 0

9. Maximum maxamnt_loans30 (maximum amount of loan taken by the user in last 30 days) is 99864.5608 and Minimum is 0

10. Maximum medianamnt_loans30 (Median of amounts of loan taken by the user in last 30 days) is 3.0 and Minimum is 0

11. Maximum cnt_loans90 (Number of loans taken by user in last 90 days) is 4997.51 and Minimum is 0

12. Maximum amnt_loans90 (Total amount of loans taken by user in last 90 days) is 438.00 and Minimum is 0

13. Maximum medianamnt_loans90 (maximum amount of loan taken by the user in last

90 days) is 3 and Minimum is 0

14. Maximum payback30 (Median of amounts of loan taken by the user in last 90 days) is 171.50 and Minimum is 0

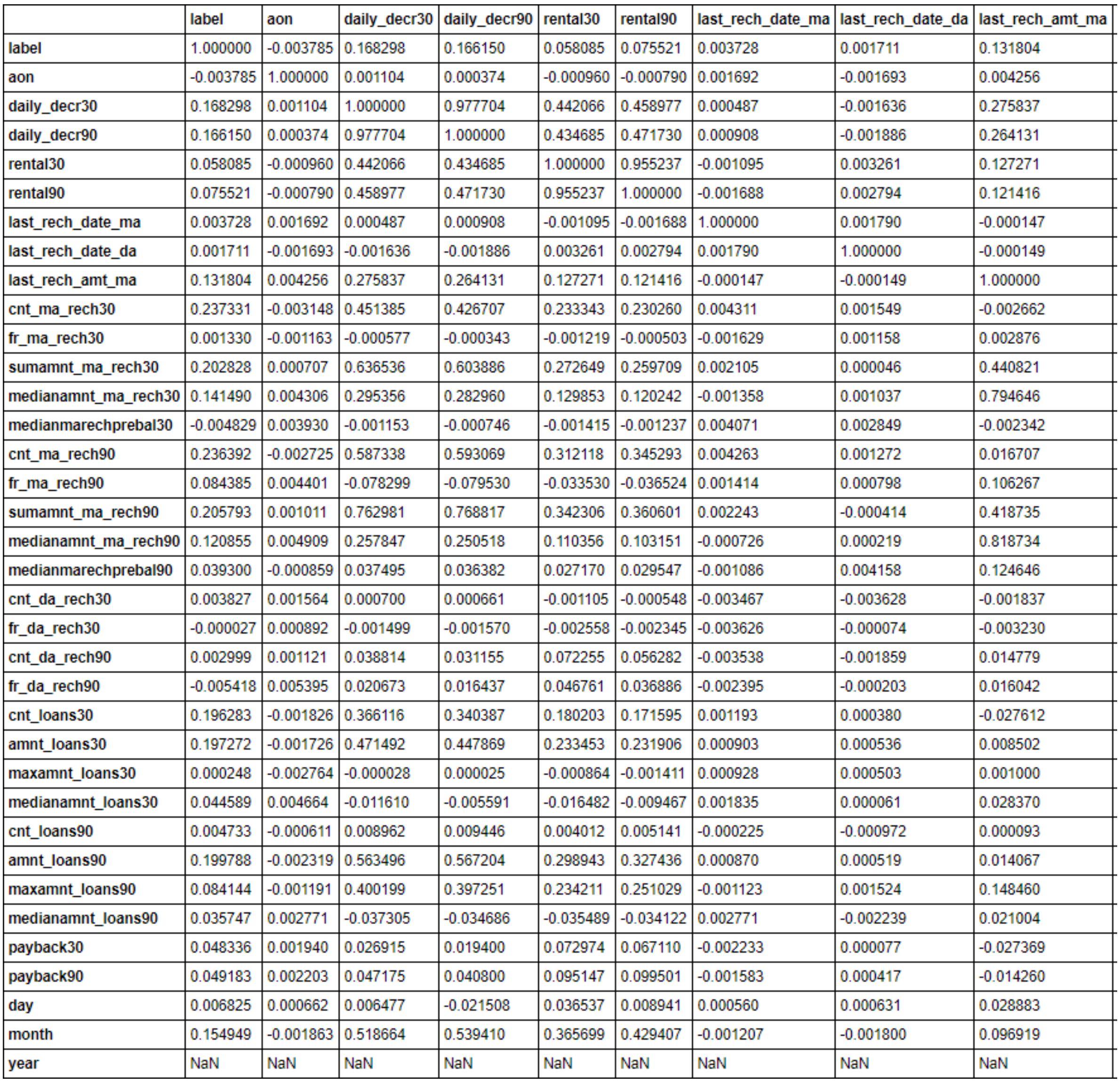
15. Maximum payback30 (Median of amounts of loan taken by the user in last 90 days) is 171.50 and Minimum is 0

16. In the columns "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" are right skewed data because mean is greater than median.

19. The columns in dataset show that Outliers are present in the dataset.

Correlation Factor

The statistical relationship between two variables is referred to as their correlation. The correlation factor represents the relation between columns in a given dataset. A correlation can be positive, meaning both variables are moving in the same direction or it can be negative, meaning that when one variable's value increasing, the other variable’s value is decreasing.

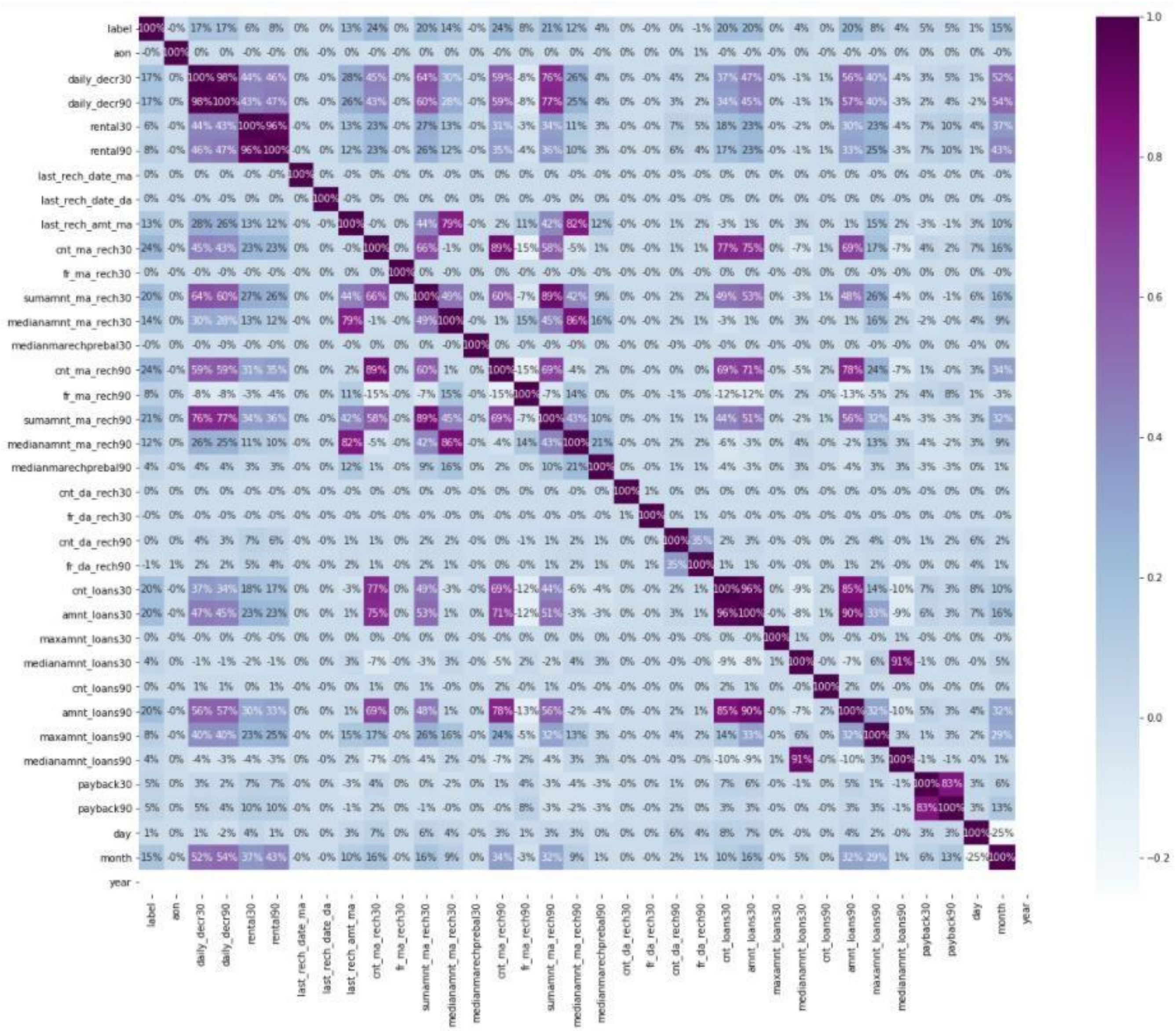


	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma
label	1.000000	-0.003785	0.168298	0.166150	0.058085	0.075521	0.003728	0.001711	0.131804
aon	-0.003785	1.000000	0.001104	0.000374	-0.000960	-0.000790	0.001692	-0.001693	0.004256
daily_decr30	0.168298	0.001104	1.000000	0.977704	0.442066	0.458977	0.000487	-0.001636	0.275837
daily_decr90	0.166150	0.000374	0.977704	1.000000	0.434685	0.471730	0.000908	-0.001886	0.264131
rental30	0.058085	-0.000960	0.442066	0.434685	1.000000	0.955237	-0.001095	0.003261	0.127271
rental90	0.075521	-0.000790	0.458977	0.471730	0.955237	1.000000	-0.001688	0.002794	0.121416
last_rech_date_ma	0.003728	0.001692	0.000487	0.000908	-0.001095	-0.001688	1.000000	0.001790	-0.000147
last_rech_date_da	0.001711	-0.001693	-0.001636	-0.001886	0.003261	0.002794	0.001790	1.000000	-0.000149
last_rech_amt_ma	0.131804	0.004256	0.275837	0.264131	0.127271	0.121416	-0.000147	-0.000149	1.000000
cnt_ma_rech30	0.237331	-0.003148	0.451385	0.426707	0.233343	0.230260	0.004311	0.001549	-0.002662
fr_ma_rech30	0.001330	-0.001163	-0.000577	-0.000343	-0.001219	-0.000503	-0.001629	0.001158	0.002876
sumamnt_ma_rech30	0.202828	0.000707	0.636536	0.603886	0.272649	0.259709	0.002105	0.000046	0.440821
medianamnt_ma_rech30	0.141490	0.004306	0.295356	0.282960	0.129853	0.120242	-0.001358	0.001037	0.794646
medianmarechprebal30	-0.004829	0.003930	-0.001153	-0.000746	-0.001415	-0.001237	0.004071	0.002849	-0.002342
cnt_ma_rech90	0.236392	-0.002725	0.587338	0.593069	0.312118	0.345293	0.004263	0.001272	0.016707
fr_ma_rech90	0.084385	0.004401	-0.078299	-0.079530	-0.033530	-0.036524	0.001414	0.000798	0.106267
sumamnt_ma_rech90	0.205793	0.001011	0.762981	0.768817	0.342306	0.360601	0.002243	-0.000414	0.418735
medianamnt_ma_rech90	0.120855	0.004909	0.257847	0.250518	0.110356	0.103151	-0.000726	0.000219	0.818734
medianmarechprebal90	0.039300	-0.000859	0.037495	0.036382	0.027170	0.029547	-0.001086	0.004158	0.124646
cnt_da_rech30	0.003827	0.001564	0.000700	0.000661	-0.001105	-0.000548	-0.003467	-0.003628	-0.001837
fr_da_rech30	-0.000027	0.000892	-0.001499	-0.001570	-0.002558	-0.002345	-0.003626	-0.000074	-0.003230
cnt_da_rech90	0.002999	0.001121	0.038814	0.031155	0.072255	0.056282	-0.003538	-0.001859	0.014779
fr_da_rech90	-0.005418	0.005395	0.020673	0.016437	0.046761	0.036886	-0.002395	-0.000203	0.016042
cnt_loans30	0.196283	-0.001826	0.366116	0.340387	0.180203	0.171595	0.001193	0.000380	-0.027612
amnt_loans30	0.197272	-0.001726	0.471492	0.447869	0.233453	0.231906	0.000903	0.000536	0.008502
maxamnt_loans30	0.000248	-0.002764	-0.000028	0.000025	-0.000864	-0.001411	0.000928	0.000503	0.001000
medianamnt_loans30	0.044589	0.004664	-0.011610	-0.005591	-0.016482	-0.009467	0.001835	0.000061	0.028370
cnt_loans90	0.004733	-0.000611	0.008962	0.009446	0.004012	0.005141	-0.000225	-0.000972	0.000093
amnt_loans90	0.199788	-0.002319	0.563496	0.567204	0.298943	0.327436	0.000870	0.000519	0.014067
maxamnt_loans90	0.084144	-0.001191	0.400199	0.397251	0.234211	0.251029	-0.001123	0.001524	0.148460
medianamnt_loans90	0.035747	0.002771	-0.037305	-0.034686	-0.035489	-0.034122	0.002771	-0.002239	0.021004
payback30	0.048336	0.001940	0.026915	0.019400	0.072974	0.067110	-0.002233	0.000077	-0.027369
payback90	0.049183	0.002203	0.047175	0.040800	0.095147	0.099501	-0.001583	0.000417	-0.014260
day	0.006825	0.000662	0.006477	-0.021508	0.036537	0.008941	0.000560	0.000631	0.028883
month	0.154949	-0.001863	0.518664	0.539410	0.365699	0.429407	-0.001207	-0.001800	0.096919
year	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Correlation Matrix

A correlation matrix is a tabular data representing the ‘correlations’ between pairs of variables in a given dataset. It is also a very important pre-processing step in Machine Learning pipelines. The Correlation matrix is a data analysis representation that is used to summarize data to understand the relationship between various different variables of the given dataset.

Correlation factor with visualization / Correlation matrix



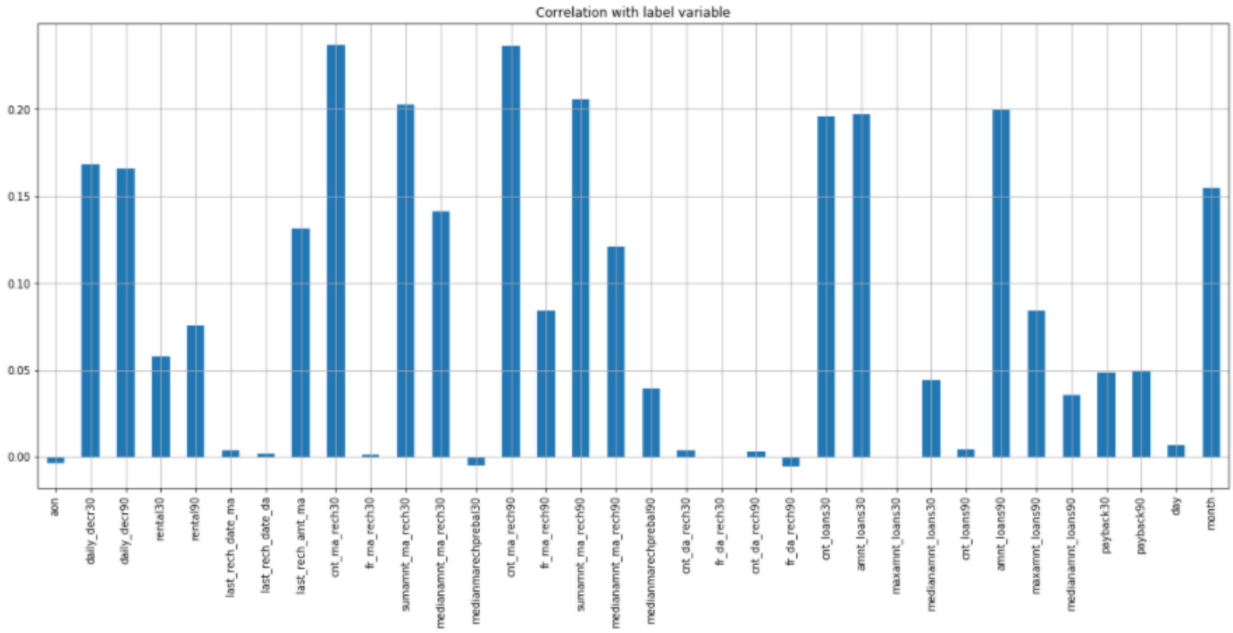
Observation

1. In the Micro Finance Service dataset the correlation graph shows that "daily_decr30, daily_decr90" columns are highly correlated with each other.

2. The columns "cnt_loans30, amnt_loans30" are are highly correlated with each other.

3. The columns "rental30, rental30" are also highly correlated with each other.

Correlation with target column (label)



Observation

1. In the Correlation with output column graph we can see that columns “medianmarechprebal30” (Median of main account balance just before recharge in last 90 days at user level (in Indonesian Rupiah) is negatively related with the Output column.

2. The column “fr_da_rech30” (Frequency of data account recharged in last 30 days) in the dataset is also negatively related with the Output column.

3. All other columns are positively related with the output column and provide significant importance towards the model building.

Hardware and Software Requirements and Tools Used

1. Software Tools used

Python 3.0

MS - Office

Operating System - Windows 10

2. Minimum Hardware Requirement

Processors: Intel Atom® processor or Intel® Core™ i3 processor

Disk space: 2 GB – 3 GB

Operating systems: Windows* 7 or later, macOS, and Linux

Python* versions: 2.7.X, 3.6.X

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

From the given dataset it can be concluded that it is a Classification problem as the output column “label” has binary output “0 & 1”. So for further analysis of the problem we have to import or call out the Classification related libraries in Python work frame.

The different libraries used for the problem solving are

sklearn - Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy.

1. sklearn.linear_model - LogisticRegression

Scikit-Learn. Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.).

Logistic regression - is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).

2. sklearn.naive_bayes - GaussianNB

As the name suggest, Gaussian Naïve Bayes classifier assumes that the data from each label is drawn from a simple Gaussian distribution. The Scikit-learn provides sklearn.naive_bayes.GaussianNB to implement the Gaussian Naïve Bayes algorithm for classification. Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

Gaussian NB algorithm is a special type of NB algorithm. It's specifically used when the features have continuous values. It's also assumed that all the features are following a gaussian distribution i.e, normal distribution. A Gaussian classifier is a generative approach in the sense that it attempts to model class posterior as well as input class-conditional distribution. Therefore, we can generate new samples in input space with a Gaussian classifier.

3. sklearn.svm - SVC

SVM is an exciting algorithm and the concepts are relatively simple. The classifier separates data points using a hyperplane with the largest amount of margin. That's why an SVM classifier is also known as a discriminative classifier. SVM finds an optimal hyperplane which helps in classifying new data points.

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification and regression challenges. It is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving a SVM model sets of labeled training data for each category, they're able to categorize new text.

4. sklearn.tree - DecisionTreeClassifier

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

There are several advantages of using decision trees for predictive analysis:

- Decision trees can be used to predict both continuous and discrete values i.e. they work well for both regression and classification tasks.
- They require relatively less effort for training the algorithm.
- They can be used to classify non-linearly separable data.
- They're very fast and efficient compared to KNN and other classification algorithms.

Decision tree learning is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree to go from observations about an item to conclusions about the item's target value.

5. sklearn.ensemble

The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator. The sklearn.ensemble module includes two averaging algorithms based on randomized decision trees: the RandomForest algorithm and the Extra-

Trees method. Both algorithms are perturb-and-combine techniques specifically designed for trees. This means a diverse set of classifiers is created by introducing randomness in the classifier construction. The prediction of the ensemble is given as the averaged prediction of the individual classifiers.

Boosting ensemble algorithms creates a sequence of models that attempt to correct the mistakes of the models before them in the sequence. Once created, the models make predictions which may be weighted by their demonstrated accuracy and the results are combined to create a final output prediction.

The different types of ensemble techniques are

i. Random Forest Classifier - Random Forest uses multiple decision trees as base learning models in the dataset. Random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting in the dataset. The main concept of Random Forest is to combine multiple decision trees in determining the final result rather than relying on individual decision trees.

ii. AdaBoost Classifier - AdaBoost is best used to boost the performance of decision trees on binary classification problems. AdaBoost was originally called AdaBoost. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers.

iii. Gradient boosting classifiers - are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting.

iv. Bagging classifier - is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Bagging is used when the goal is to reduce the variance of a decision tree classifier.

v. Extra Trees Classifier - is an ensemble learning method fundamentally based on decision trees. Extra Trees Classifier, like Random Forest, randomizes certain decisions and subsets of data to minimize over-learning from the data and over fitting. This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

6. sklearn.metrics - The sklearn. metrics module implements several loss, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decisions values.

Important sklearn.metrics modules are

i. classification_report - A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report.

It shows the main classification metrics precision, recall and f1-score on a per-class basis. The metrics are calculated by using true and false positives, true and false negatives. Positive and negative in this case are generic names for the predicted classes. There are four ways to check if the predictions are right or wrong:

TN / True Negative: when a case was negative and predicted

negative TP / True Positive: when a case was positive and predicted

positive FN / False Negative: when a case was positive but predicted

negative FP / False Positive: when a case was negative but predicted

positive

ii. confusion_matrix - a confusion matrix is a tabular summary of the number of correct and incorrect predictions made by a classifier. It can be used to evaluate the performance of a classification model through the calculation of performance metrics like accuracy, precision, recall, and F1-score.

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

iii. accuracy_score - Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.

Accuracy = TP+TN/TP+FP+FN+TN

iv. roc_curve - An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters i.e. True Positive Rate & False Positive Rate. ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

v. auc (Area Under the Curve) - AUC stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve from (0,0) to (1,1). AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example.

7. sklearn.model_selection –

i. GridSearchCV - It is a library function that is a member of sklearn's model_selection package. It helps to loop through predefined hyper parameters and fit your estimator (model) on your training set. So, in the end, you can select the best parameters from the listed hyperparameters. GridSearchCV combines an estimator with a grid search preamble to tune hyper-parameters. The method picks the optimal parameter from the grid search and uses it with the estimator selected by the user.

ii. cross_val_score - Cross validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part being 20% of full dataset. While running the Cross validation the 1st part (20%) of the 5 parts will be kept out as a hold out set for validation and everything else is used for training data. This way we will get the first estimate of the model quality of the dataset. In the similar way further iterations are made for the second 20% of the dataset is held as a hold out set and remaining 4 parts are used for training data during process. This way we will get the second estimate of the model quality of the dataset. These steps are repeated during the cross validation process to get the remaining estimate of the model quality.

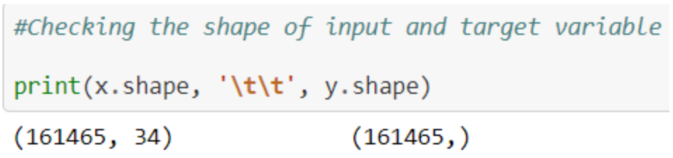
cross_val_score estimates the expected accuracy of the model on out-of-training data (pulled from the same underlying process as the training data). The benefit is that one need not set aside any data to obtain this metric, and we can still train the model on all of the available data.

Testing of Identified Approaches (Algorithms)

Algorithm used for training and testing.

After completing the required pre processing techniques for the model building data is separated as input and output columns before passing it to the train_test_split.

Checking the shape of input and target variable



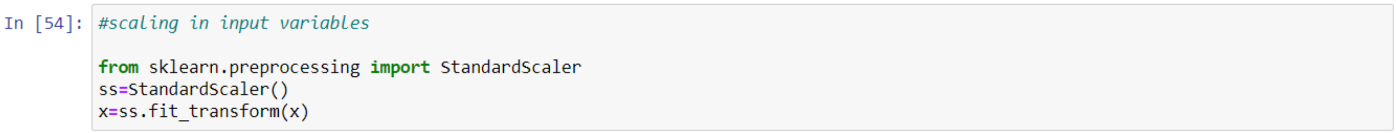
```
#Checking the shape of input and target variable

print(x.shape, '\t\t', y.shape)
```

```
(161465, 34)          (161465,)
```

Scaling the input variables

Standard Scaler - The idea behind StandardScaler is that it will transform the data such that its distribution will have a mean value 0 and standard deviation of 1. In case of multivariate data, this is done feature-wise (in other words independently for each column of the data). Given the distribution of the data, each value in the dataset will have the mean value subtracted, and then divided by the standard deviation of the whole dataset (or feature in the multivariate case).

It transforms the data in such a manner that it has mean as 0 and standard deviation as 1. In short, it standardizes the data. Standardization is useful for data which has negative values. It arranges the data in a standard normal distribution.



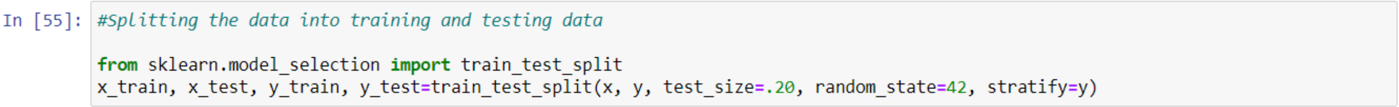
```
In [54]: #scaling in input variables

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

Splitting the data into training and testing data

Train Test Split

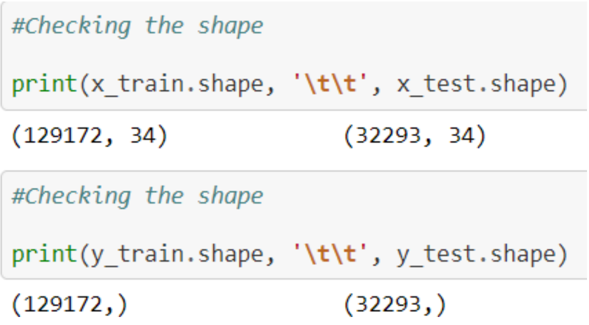
Scikit-learn is a Python library that offers various features for data processing that can be used for classification, clustering, and model selection. Model_selection is a method for setting a blueprint to analyze data and then using it to measure new data. Selecting a proper model allows you to generate accurate results when making a prediction. If we have one dataset, then it needs to be split by using the Sklearn train_test_split function first. By default, Sklearn train_test_split will make random partitions for the two subsets.

The train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, we don't need to divide the dataset manually. The train_test_split function is for splitting a single dataset for two different purposes: training and testing. The testing subset is for building your model. The testing subset is for using the model on unknown data to evaluate the performance of the model.



```
In [55]: #Splitting the data into training and testing data

         from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=.20, random_state=42, stratify=y)
```

Checking the shape of the x_train, x_test & y_train, y_test



```
#Checking the shape

print(x_train.shape, '\t\t', x_test.shape)
```

```
(129172, 34)             (32293, 34)
```

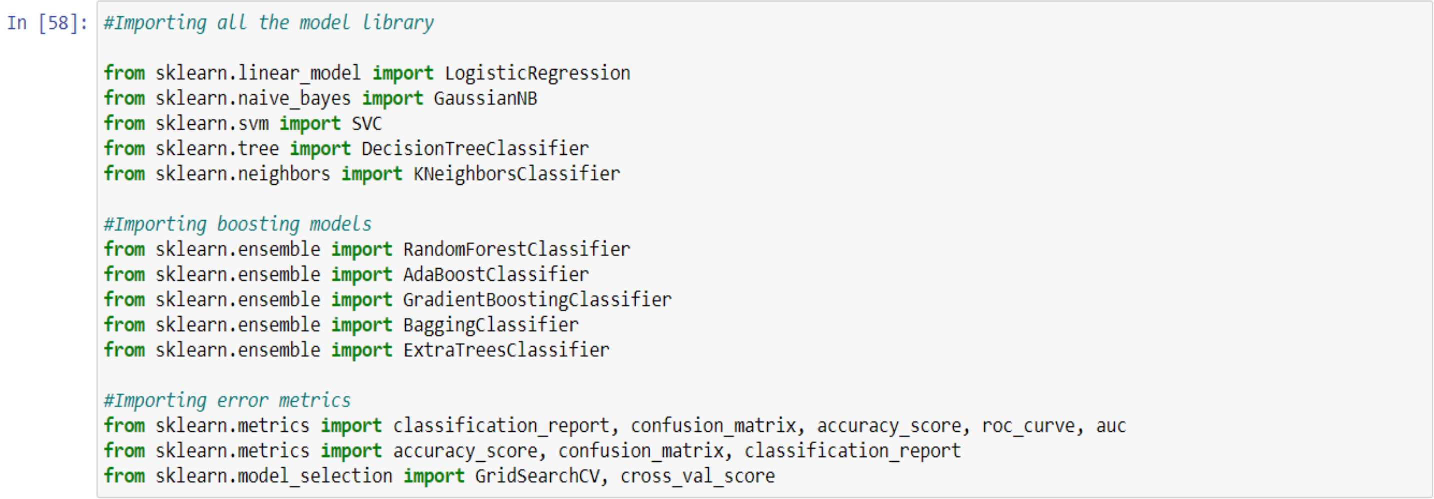
```
#Checking the shape

print(y_train.shape, '\t\t', y_test.shape)
```

```
(129172,)                (32293,)
```

During the train_test_split process with the test_size=.20 & random_state=42 the data is split in x_train, x_test, y_train, y_test sections. Where x_train has 129172 rows and 34 columns, x-test has 32293 rows and 34 columns in their shape. The y_train has 129172 rows & y_test has 32293 rown in its shape.

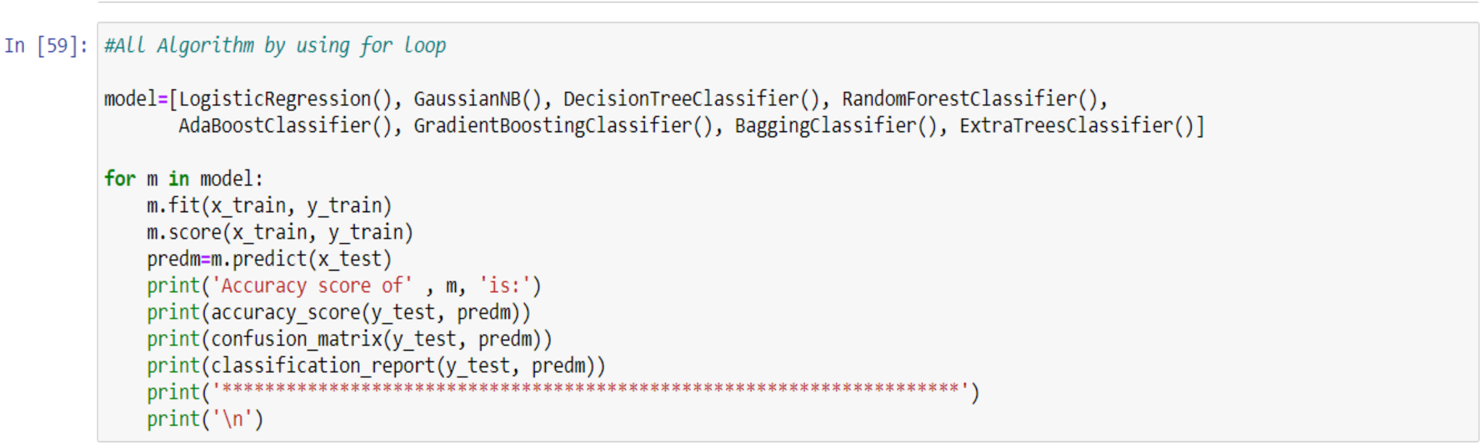
Run and Evaluate selected models

Before running the model & evaluating them, first we need to import all the necessary libraries which are required for the problem solving.



```
In [58]: #Importing all the model library

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier

#Importing boosting models
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier

#Importing error metrics
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_curve, auc
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.model_selection import GridSearchCV, cross_val_score
```

Now using for loop running all the models such as “Logistic Regression, GaussianNB, Decision Tree Classifier, Random Forest Classifier, Ada Boost Classifier, Gradient Boosting Classifier, Bagging Classifier, Extra Trees Classifier”.

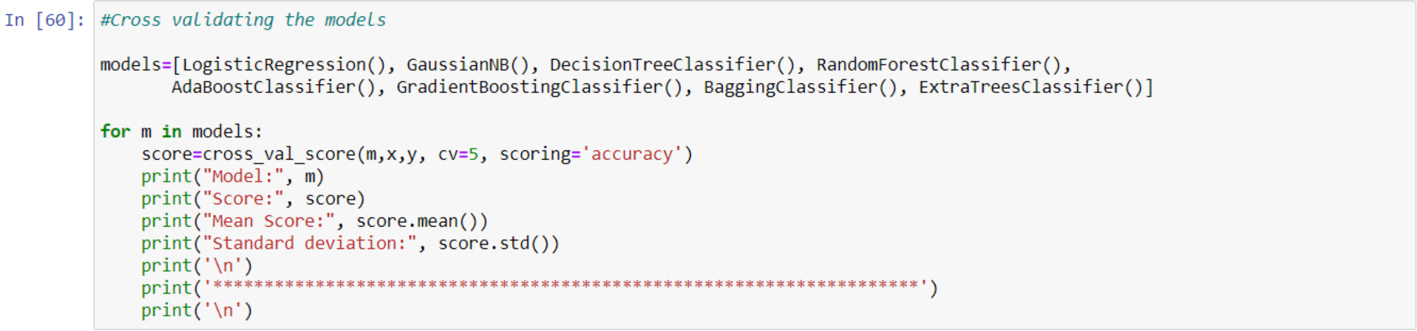


```
In [59]: #All Algorithm by using for loop

model=[LogisticRegression(), GaussianNB(), DecisionTreeClassifier(), RandomForestClassifier(),
       AdaBoostClassifier(), GradientBoostingClassifier(), BaggingClassifier(), ExtraTreesClassifier()]

for m in model:
    m.fit(x_train, y_train)
    m.score(x_train, y_train)
    predm=m.predict(x_test)
    print('Accuracy score of' , m, 'is:')
    print(accuracy_score(y_test, predm))
    print(confusion_matrix(y_test, predm))
    print(classification_report(y_test, predm))
    print('**********************************************************************')
    print('\n')
```

Output

Cross Validation

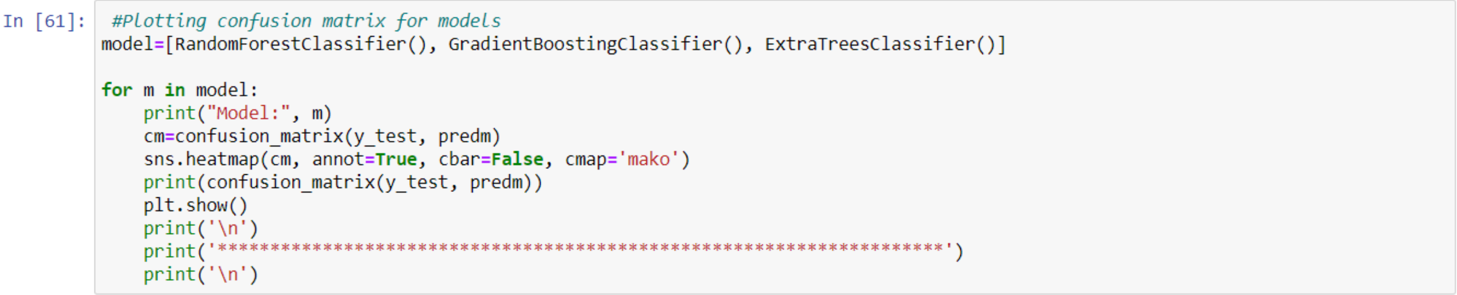
Now using for loop checking cross_val_score for all the models i.e. “Logistic Regression, GaussianNB, Decision Tree Classifier, Random Forest Classifier, Ada Boost Classifier, Gradient Boosting Classifier, Bagging Classifier, Extra Trees Classifier”.



```
In [60]: #Cross validating the models

         models=[LogisticRegression(), GaussianNB(), DecisionTreeClassifier(), RandomForestClassifier(),
                AdaBoostClassifier(), GradientBoostingClassifier(), BaggingClassifier(), ExtraTreesClassifier()]

         for m in models:
             score=cross_val_score(m,x,y, cv=5, scoring='accuracy')
             print("Model:", m)
             print("Score:", score)
             print("Mean Score:", score.mean())
             print("Standard deviation:", score.std())
             print('\n')
             print('*********************************************************************')
             print('\n')
```

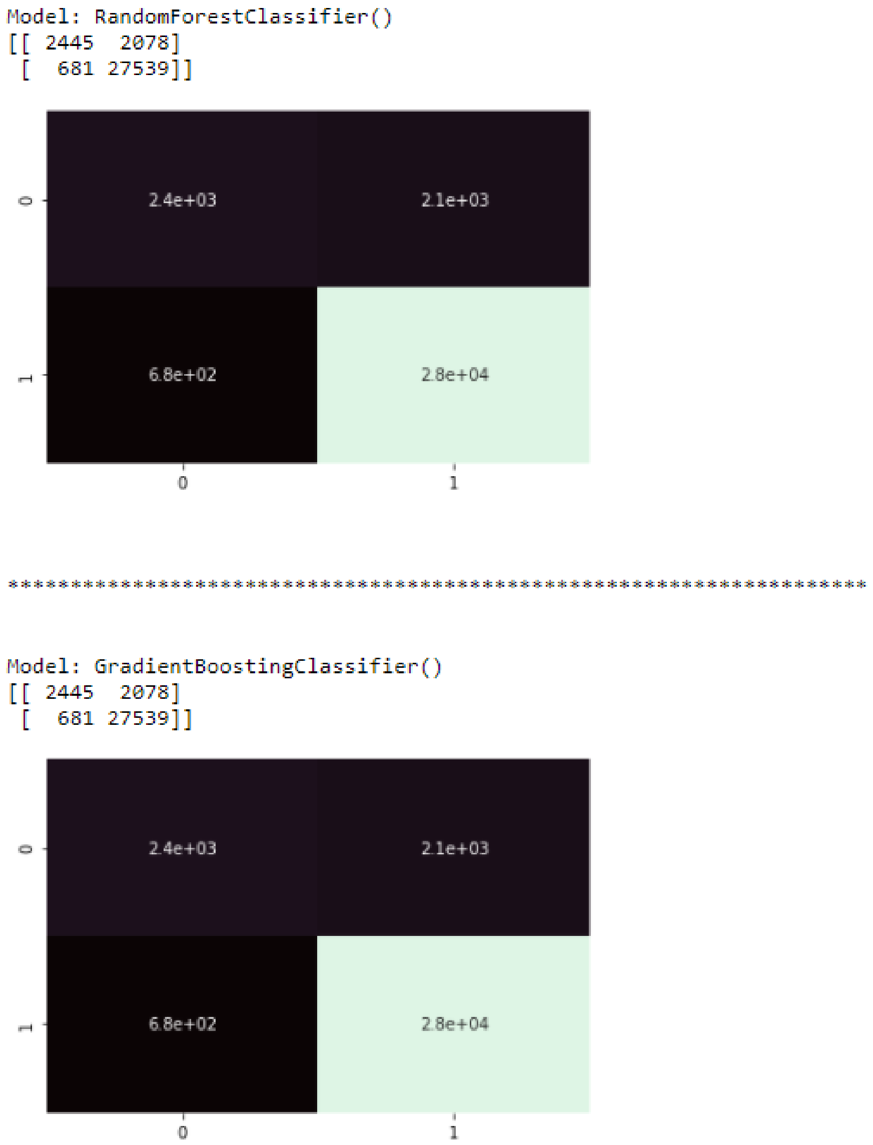
Confusion Matrix for all Models

Now using for loop plotting Confusion Matrix for all the models i.e. “Logistic Regression, GaussianNB, Decision Tree Classifier, Random Forest Classifier, Ada Boost Classifier, Gradient Boosting Classifier, Bagging Classifier, Extra Trees Classifier”.



```
In [61]:  #Plotting confusion matrix for models
         model=[RandomForestClassifier(), GradientBoostingClassifier(), ExtraTreesClassifier()]

         for m in model:
             print("Model:", m)
             cm=confusion_matrix(y_test, predm)
             sns.heatmap(cm, annot=True, cbar=False, cmap='mako')
             print(confusion_matrix(y_test, predm))
             plt.show()
             print('\n')
             print('*********************************************************************')
             print('\n')
```

Output



```
Model: RandomForestClassifier()
[[ 2445  2078]
 [  681 27539]]
```

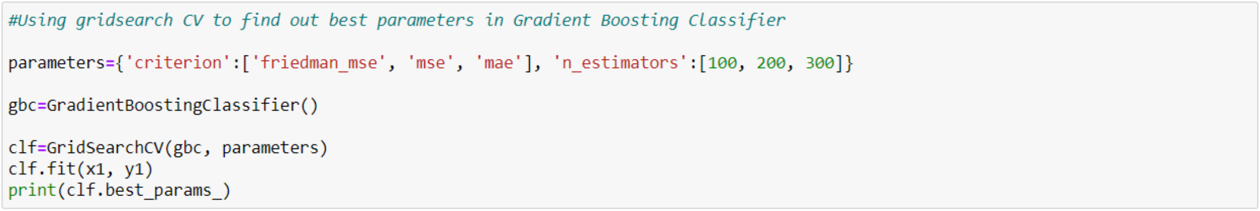
```
*********************************************************************

Model: GradientBoostingClassifier()
[[ 2445  2078]
 [  681 27539]]
```

Observation

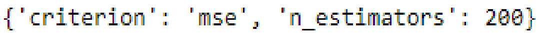
In the above models the Logistic Regression, Gradient Boosting Classifier, Random Forest Classifier, and Extra Trees Classifier models are working better with high accuracy score which is above 90%. For further analysis these models are further evaluated under GridSearchCV model.

Key Metrics for success in solving problem under consideration

Using gridsearch CV to find out best parameters in Gradient Boosting Classifier

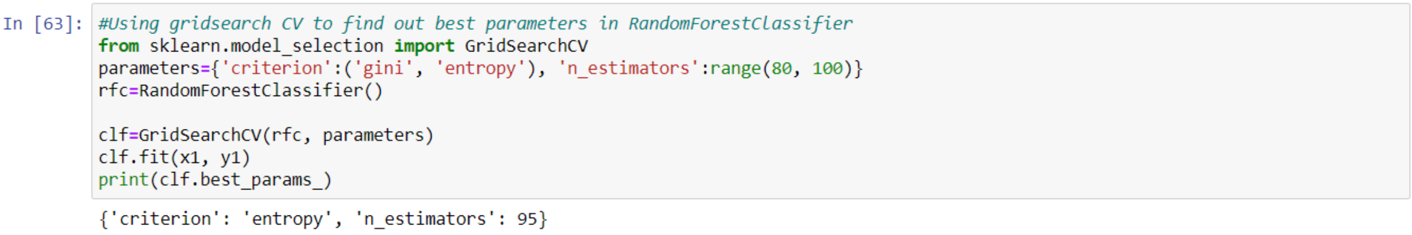


```
#Using gridsearch CV to find out best parameters in Gradient Boosting Classifier

parameters={'criterion':['friedman_mse', 'mse', 'mae'], 'n_estimators':[100, 200, 300]}

gbc=GradientBoostingClassifier()

clf=GridSearchCV(gbc, parameters)
clf.fit(x1, y1)
print(clf.best_params_)
```



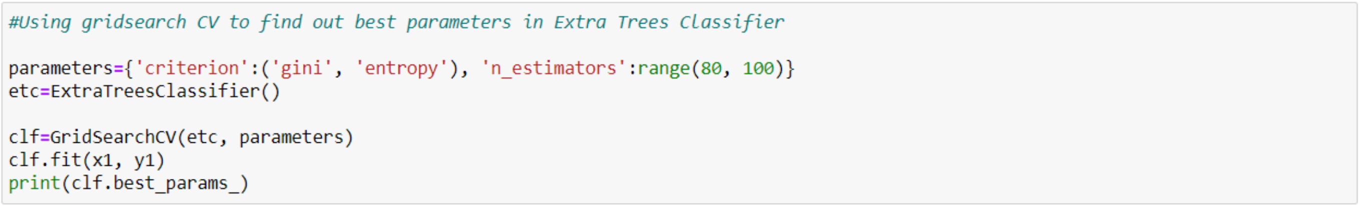
```
{'criterion': 'mse', 'n_estimators': 200}
```

Using gridsearch CV to find out best parameters in RandomForestClassifier

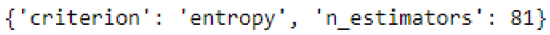


```
In [63]: #Using gridsearch CV to find out best parameters in RandomForestClassifier
         from sklearn.model_selection import GridSearchCV
         parameters={'criterion':('gini', 'entropy'), 'n_estimators':range(80, 100)}
         rfc=RandomForestClassifier()

         clf=GridSearchCV(rfc, parameters)
         clf.fit(x1, y1)
         print(clf.best_params_)

         {'criterion': 'entropy', 'n_estimators': 95}
```

Using gridsearch CV to find out best parameters in Extra Trees Classifier

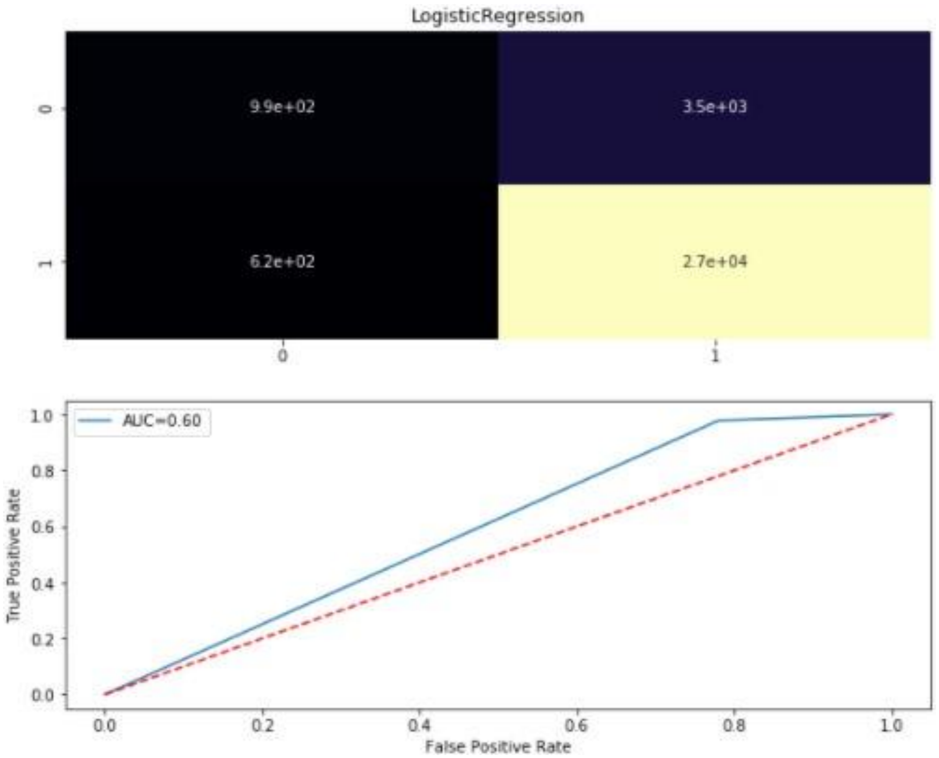


```
#Using gridsearch CV to find out best parameters in Extra Trees Classifier

parameters={'criterion':('gini', 'entropy'), 'n_estimators':range(80, 100)}
etc=ExtraTreesClassifier()

clf=GridSearchCV(etc, parameters)
clf.fit(x1, y1)
print(clf.best_params_)
```

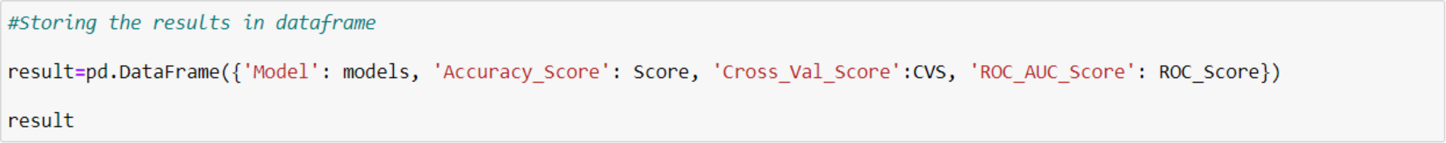


```
{'criterion': 'entropy', 'n_estimators': 81}
```

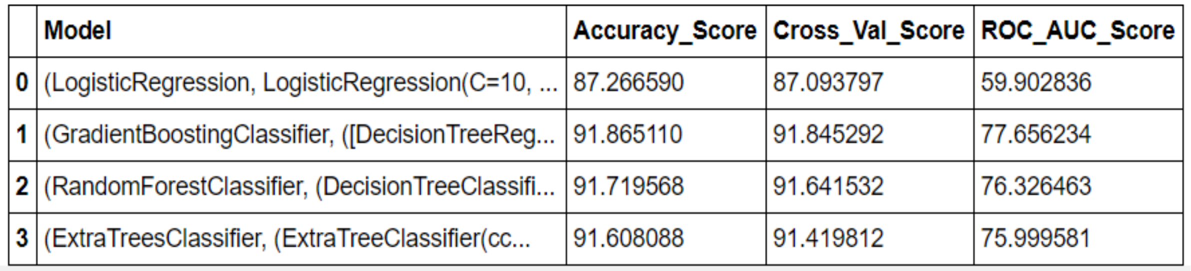
Predictive modeling



Storing the final results in data frame



```
#Storing the results in dataframe

result=pd.DataFrame({'Model': models, 'Accuracy_Score': Score, 'Cross_Val_Score':CVS, 'ROC_AUC_Score': ROC_Score})

result
```



	Model	Accuracy_Score	Cross_Val_Score	ROC_AUC_Score
0	(LogisticRegression, LogisticRegression(C=10, ...	87.266590	87.093797	59.902836
1	(GradientBoostingClassifier, ([DecisionTreeReg...	91.865110	91.845292	77.656234
2	(RandomForestClassifier, (DecisionTreeClassifi...	91.719568	91.641532	76.326463
3	(ExtraTreesClassifier, (ExtraTreeClassifier(cc...	91.608088	91.419812	75.999581

Observation

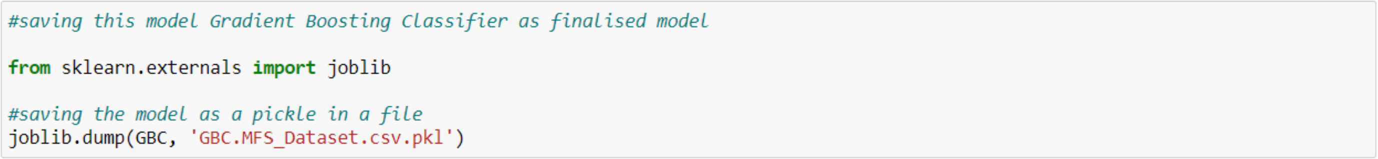
1. In the above Table it shows that Gradient Boosting Classifier model is working with highest accuracy score of 91.865110

2. The cross_val_score of model is 91.845292 which again show that cross val score is better compared to other models in the table.

3. The ROC_AUC_score of the model is showing as 77.656234 which show that model is learning with highest accuracy in the loop compared to other models.

4. All these points proves that Gradient Boosting Classifier model is working best and can be considered as finalised model.

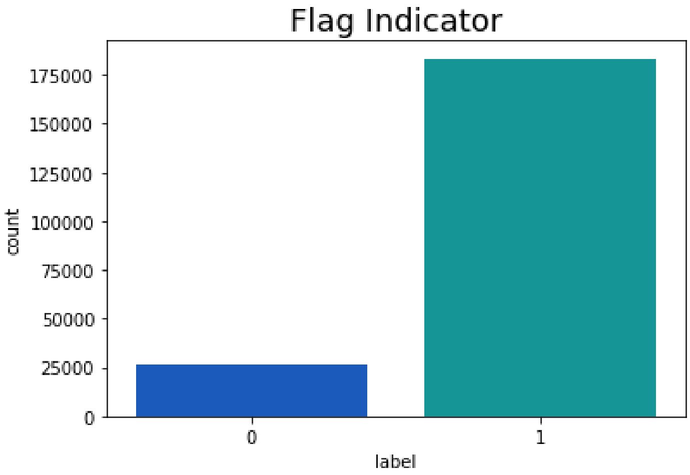
Choosing the finalised model



```
#saving this model Gradient Boosting Classifier as finalised model

from sklearn.externals import joblib

#saving the model as a pickle in a file
joblib.dump(GBC, 'GBC.MFS_Dataset.csv.pkl')
```

Visualizations & Interpretation of the Results

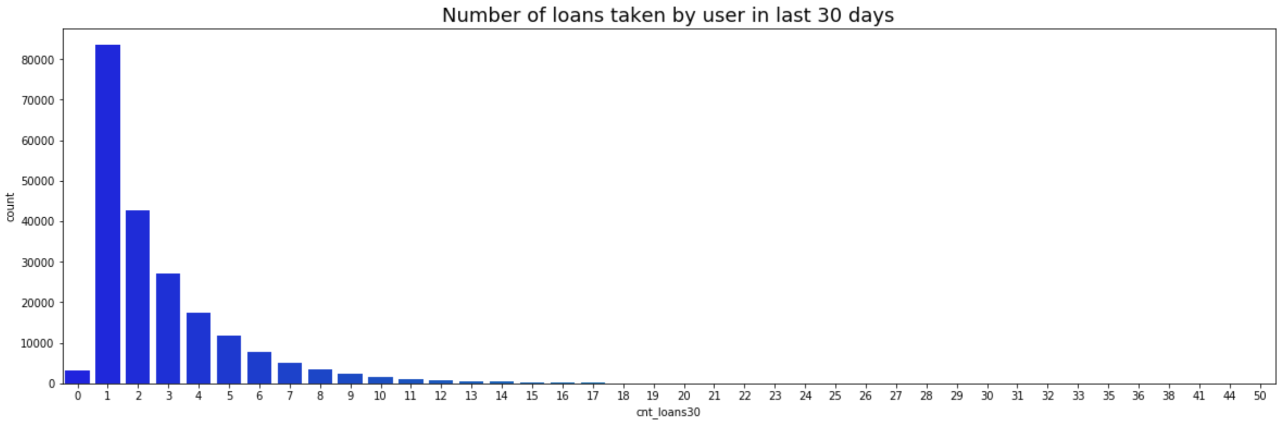
Now we can analyze all the plots, graphs in the dataset and the inferences and observations obtained from them.



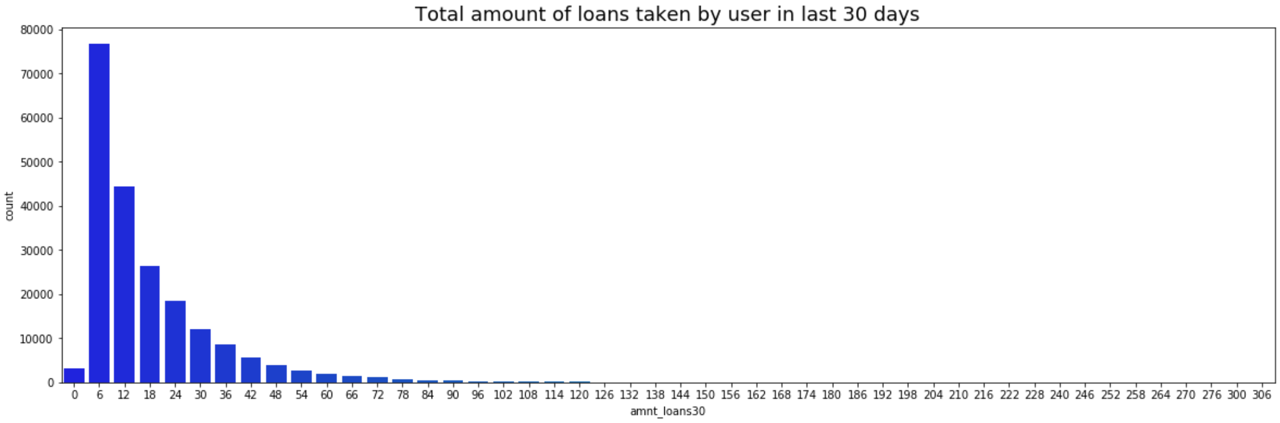
Observation

1. Out of 209593 cases 183431 are success i.e. user paid back the credit amount within 5 days of issuing the loan.
2. Out of 209593 cases 26162 are failure cases i.e. failed to user pay back the credit amount within 5 days of issuing the loan.

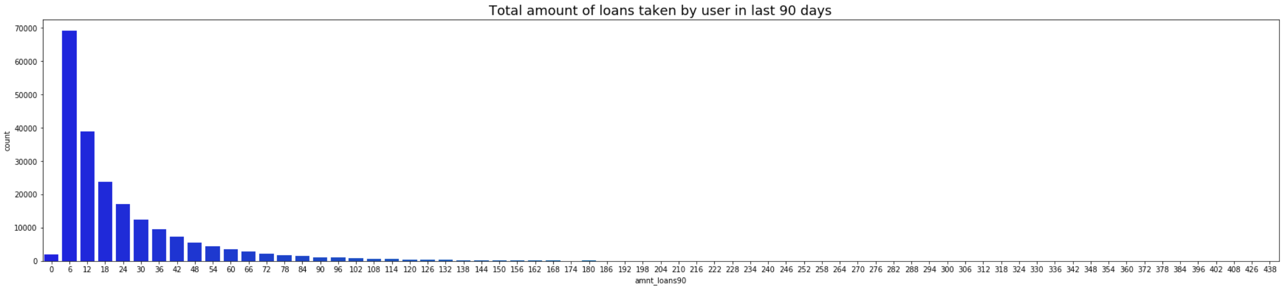
<u>Checking the Number of loans taken by user in last 30 days</u>



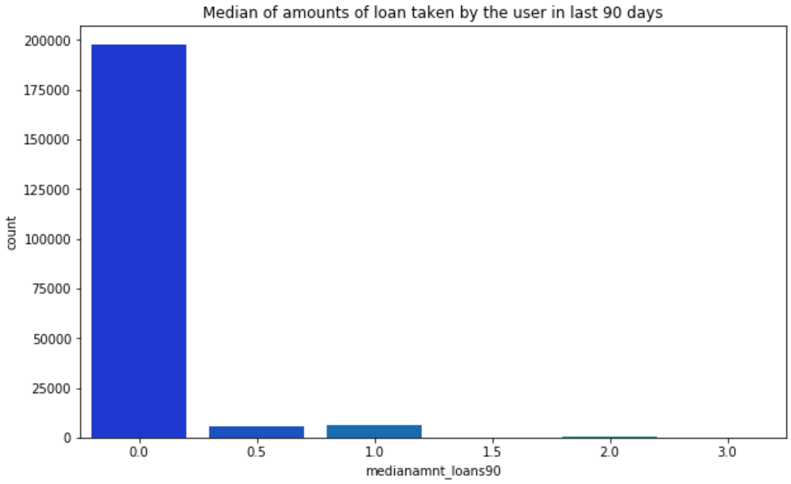
Checking the Number of Total amount of loans taken by user in last 30 days



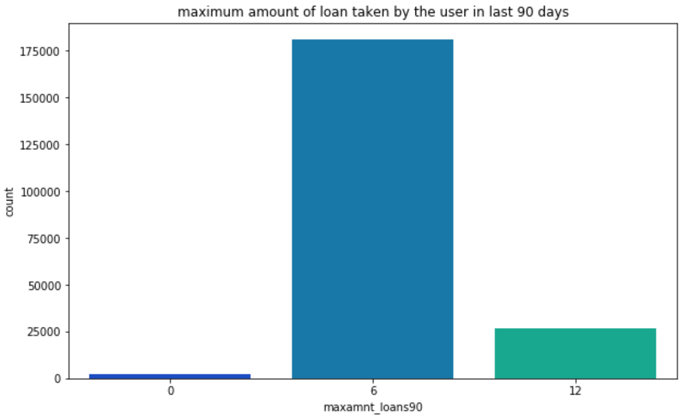
Checking the Number of Total amount of loans taken by user in last 90 days



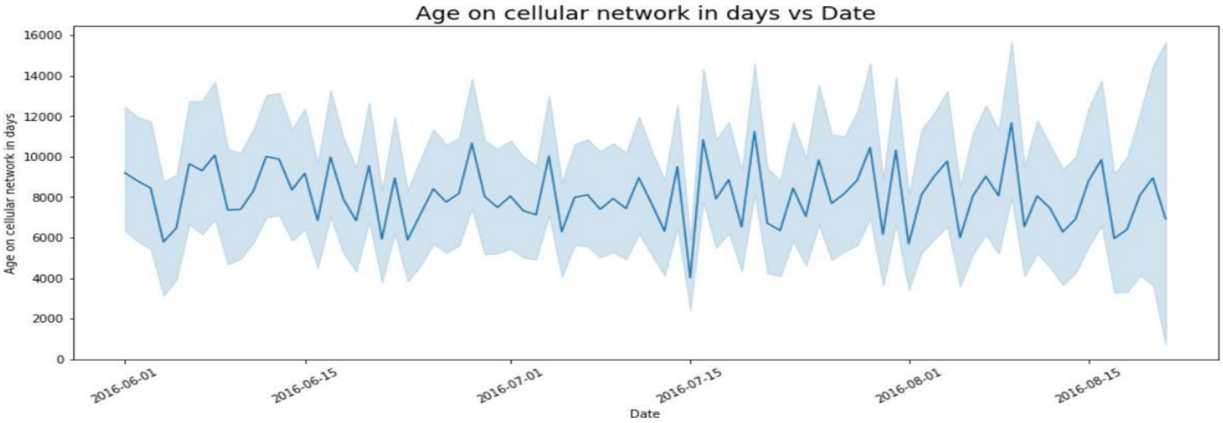
Checking the Number of Total amount of loans taken by user in last 90 days



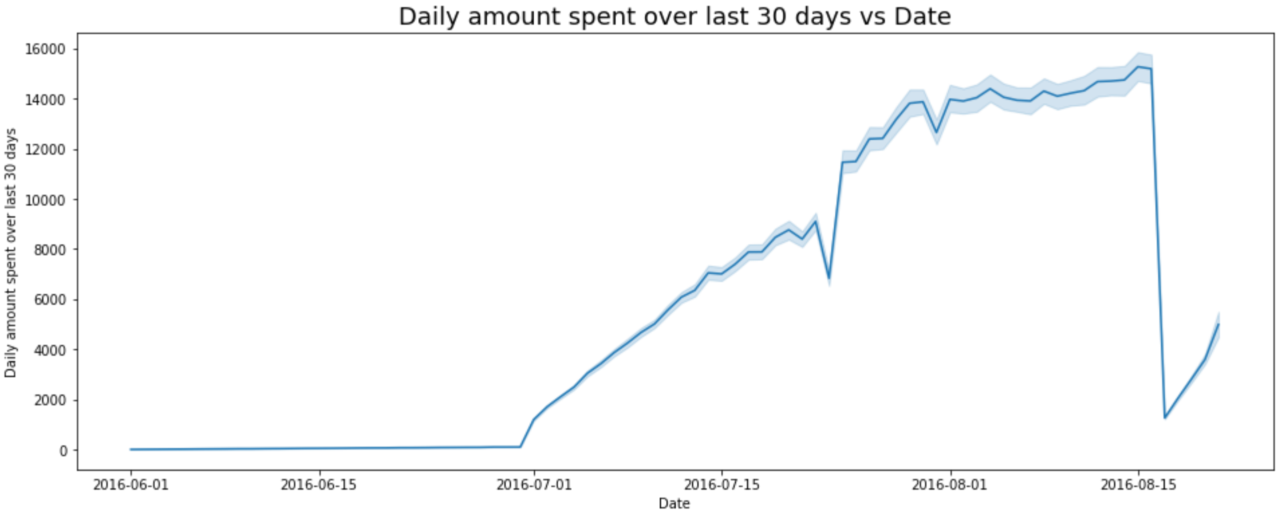
Checking the maximum amount of loan taken by the user in last 90 days



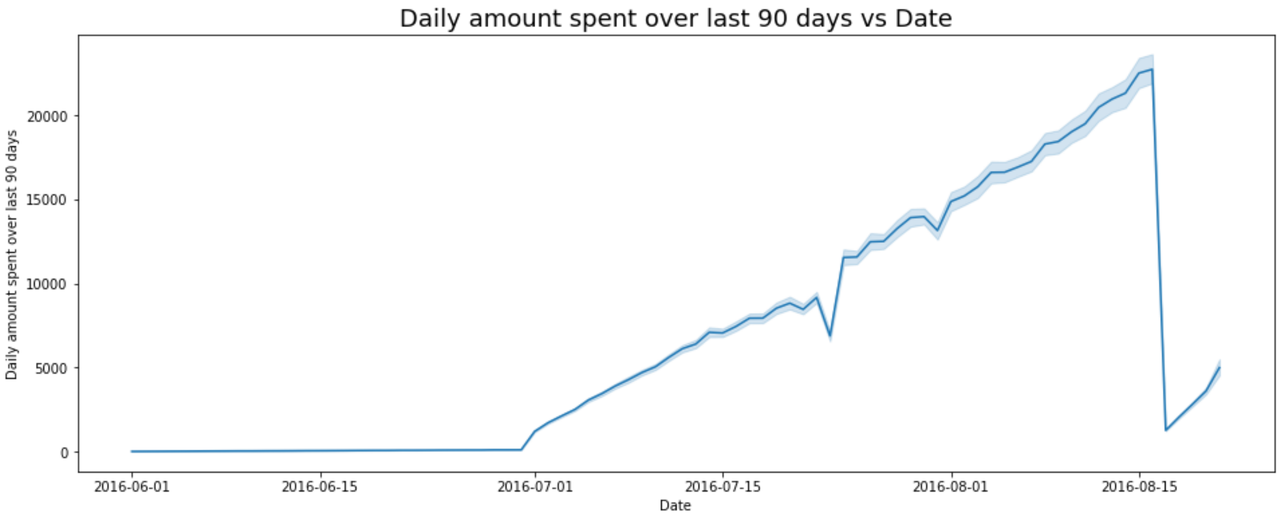
Month wise Flag Indicator i.e user paid back the credit amount within 5 days of issuing the loan



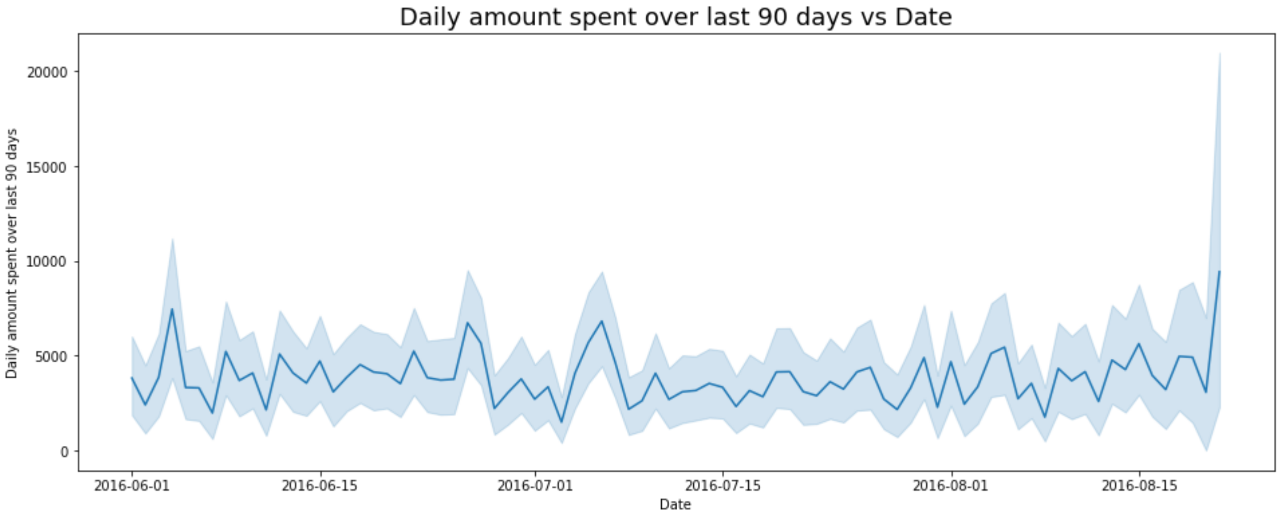
Using lineplot from seaborn to check Daily amount spent from main account, averaged over last 30 days vs pdate



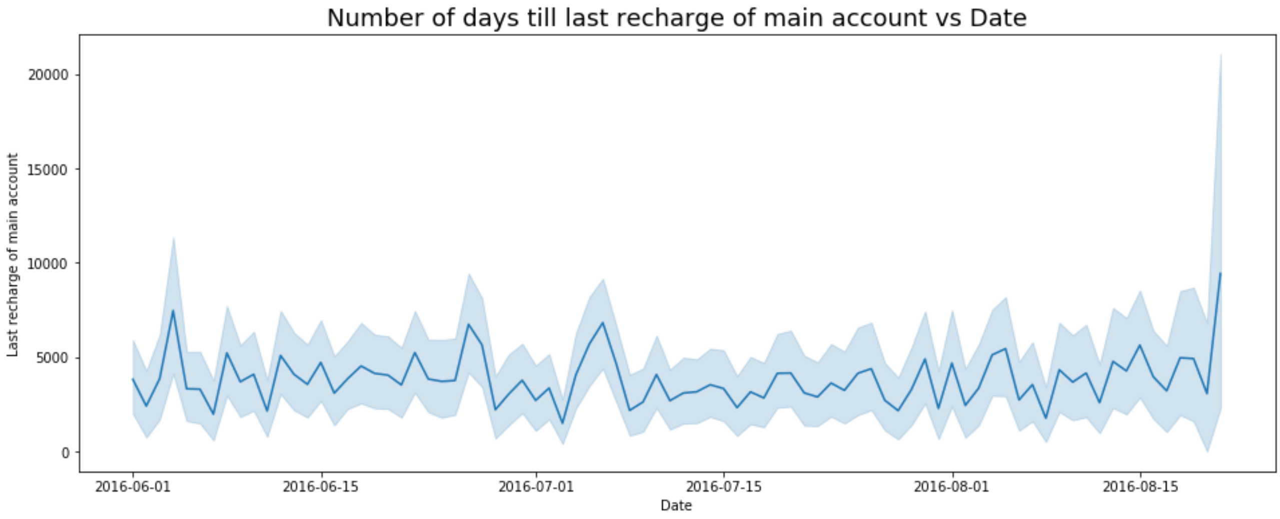
Using lineplot from seaborn to check Daily amount spent from main account, averaged over last 90 days vs pdate



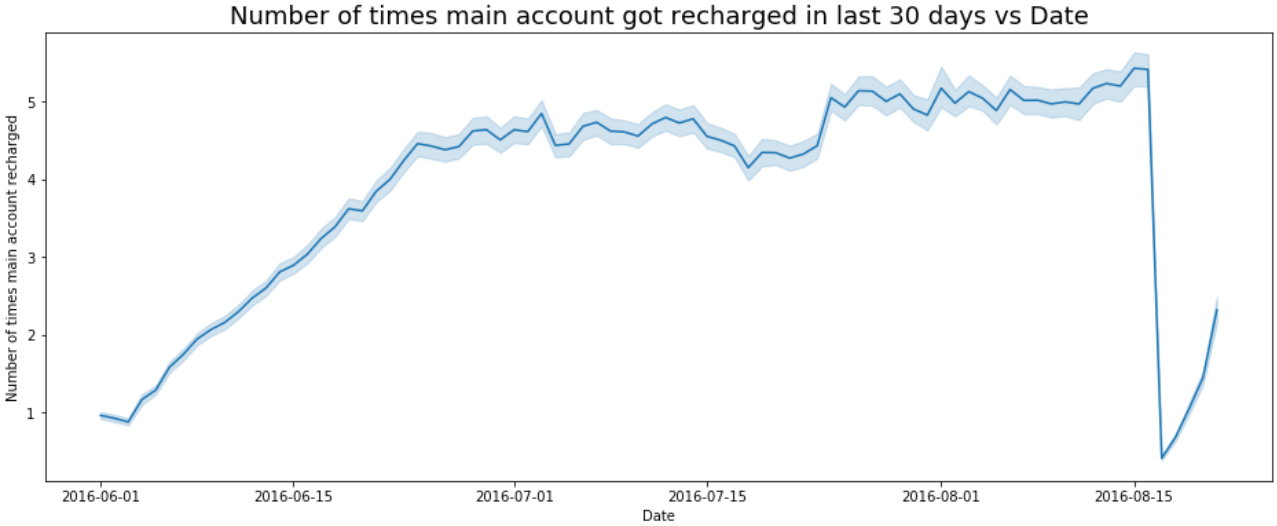
Using lineplot from seaborn to check Daily amount spent from main account, averaged over last 90 days vs pdate



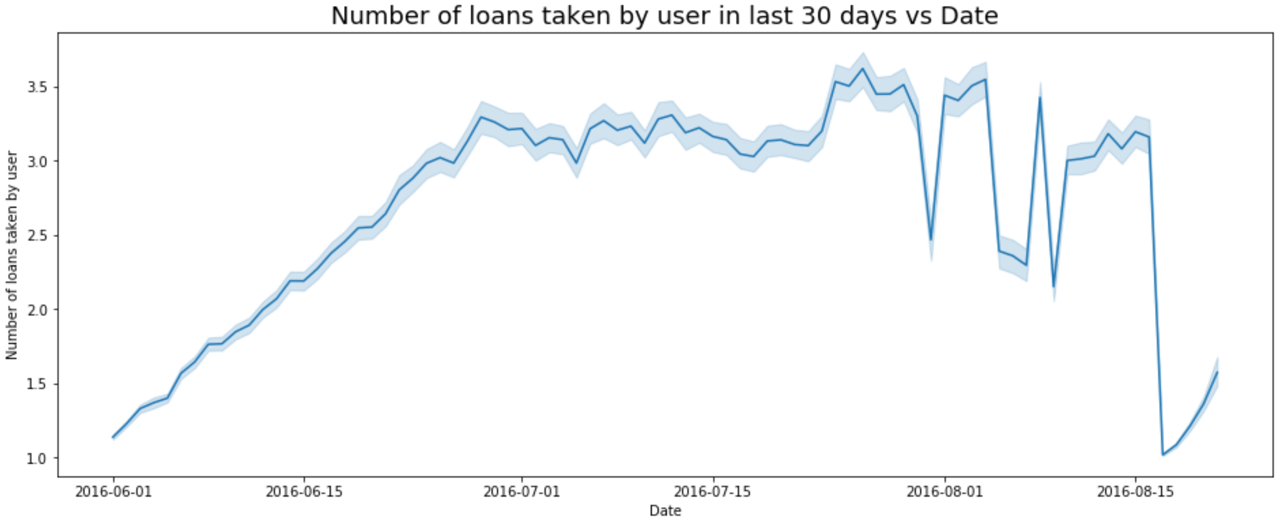
Using lineplot from seaborn to check Number of times main account got recharged in last 30 days vs pdate



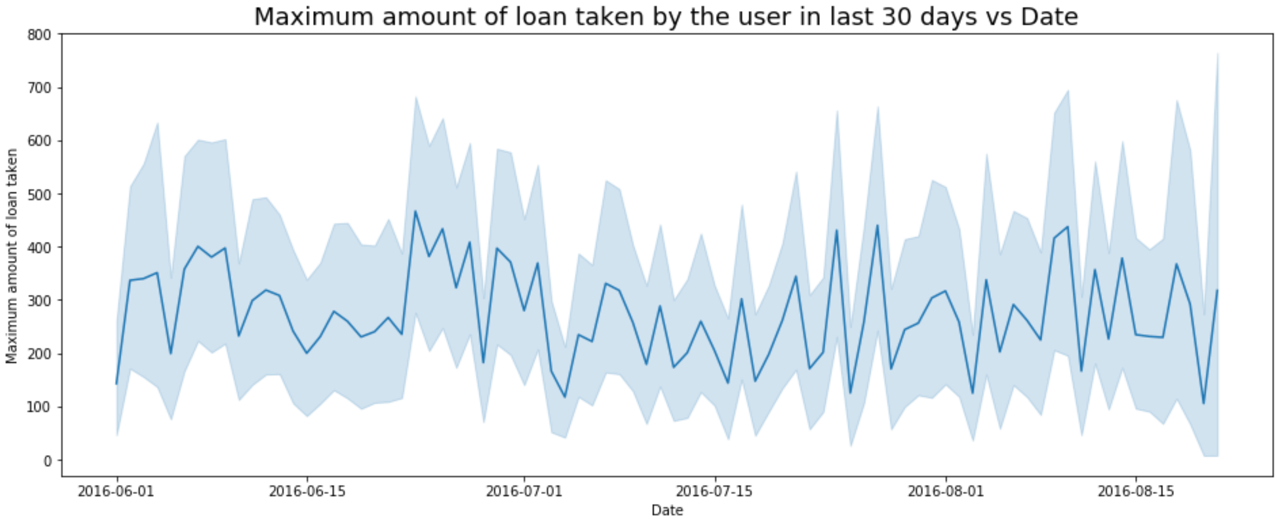
Using lineplot from seaborn to check Number of loans taken by user in last 30 days vs pdate



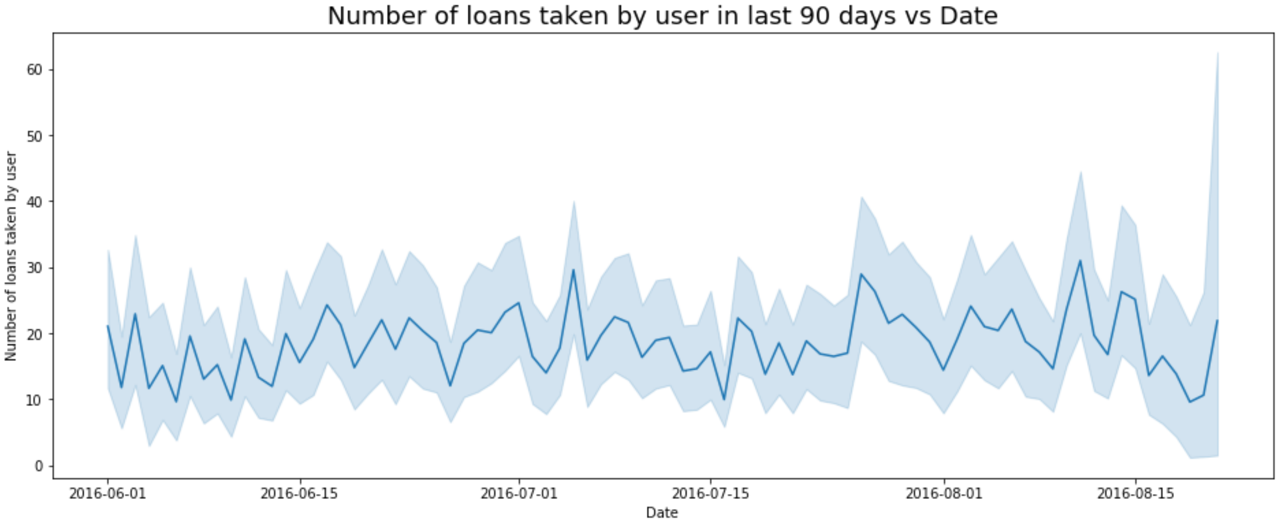
using lineplot from seaborn to check Total amount of loans taken by user in last 30 days vs pdate



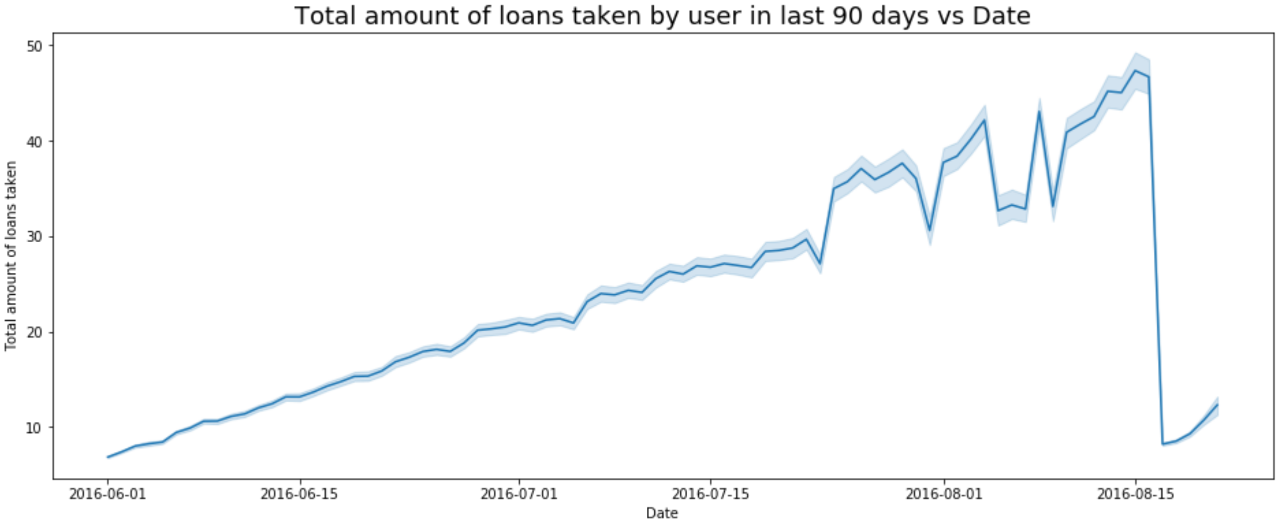
Using lineplot from seaborn to check maximum amount of loan taken by the user in last 30 days vs pdate



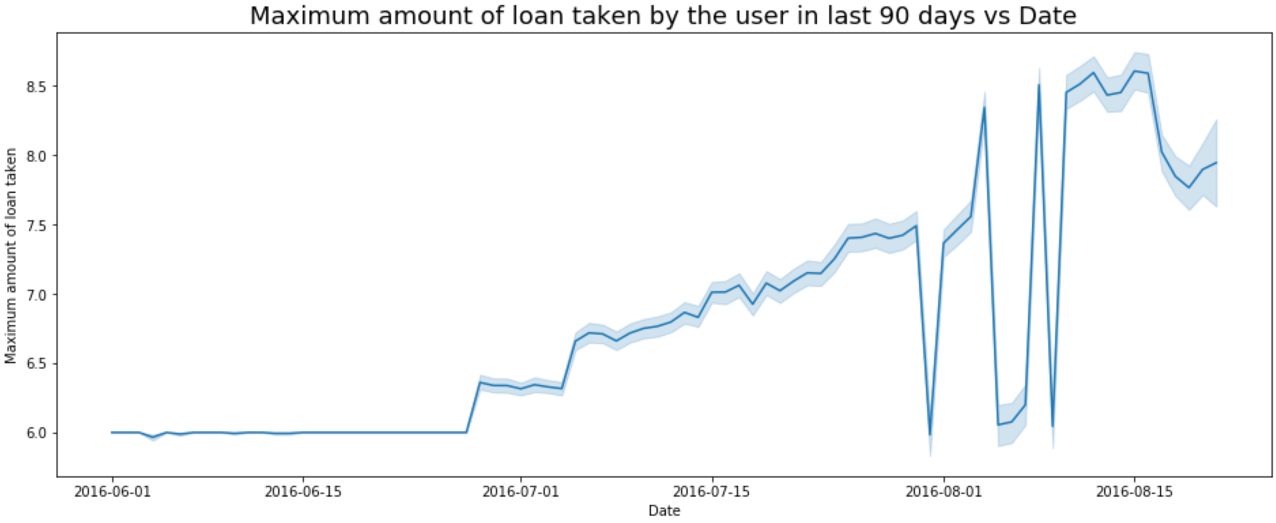
Using lineplot from seaborn to check Number of loans taken by user in last 90 days vs pdate



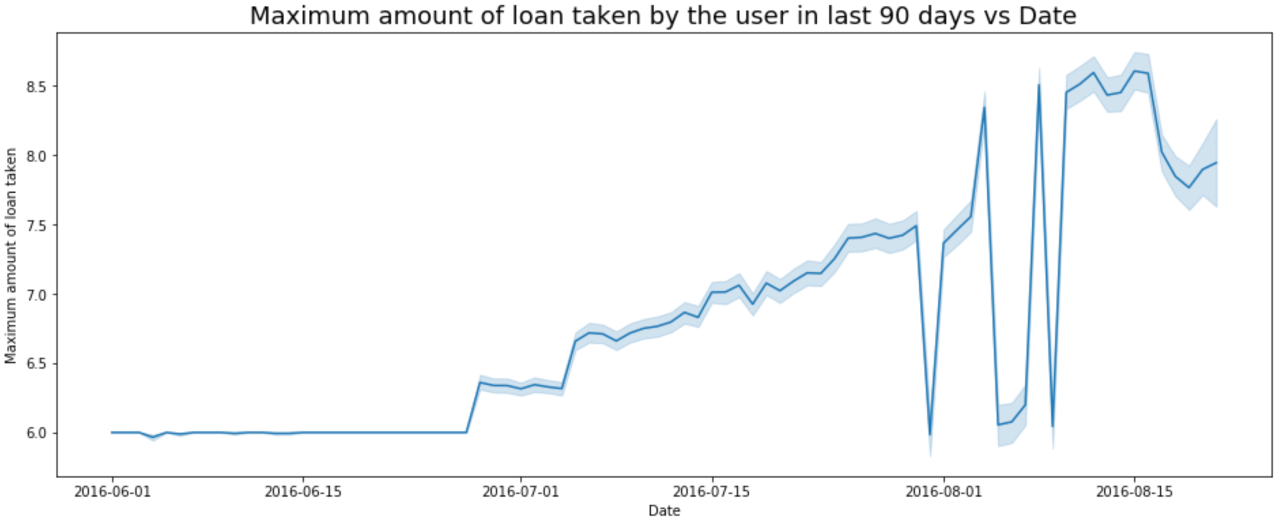
Using lineplot from seaborn to check Total amount of loans taken by user in last 90 days vs pdate



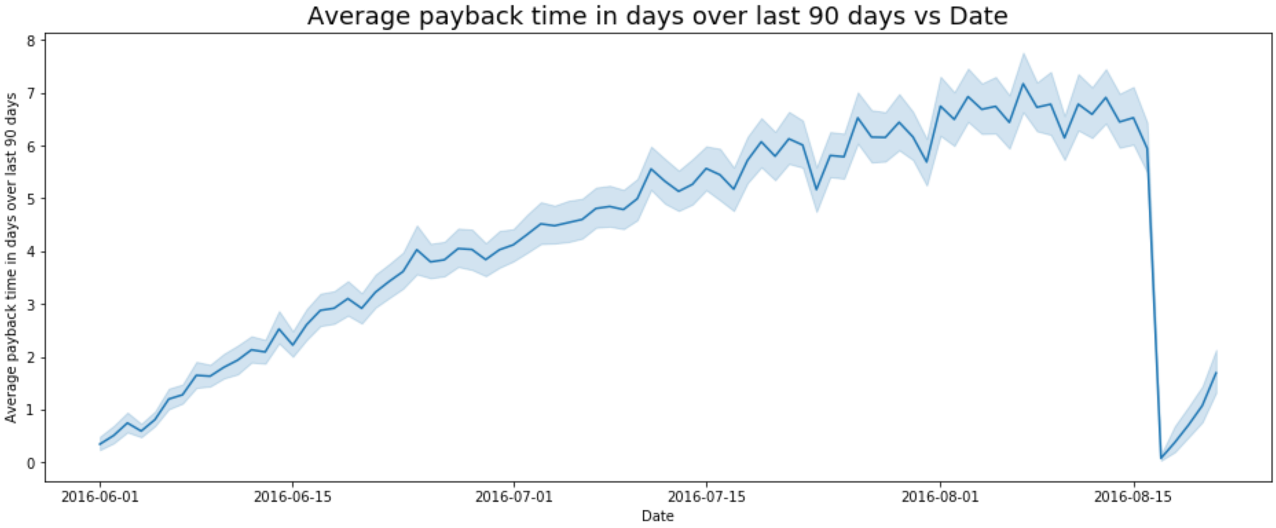
Using lineplot from seaborn to check maximum amount of loan taken by the user in last 90 days vs pdate



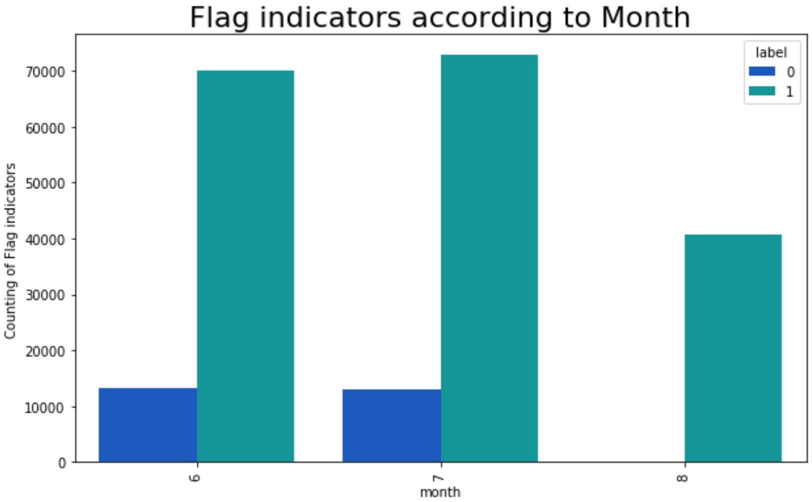
Using lineplot from seaborn to check Average payback time in days over last 30 days vs pdate



Using lineplot from seaborn to check Average payback time in days over last 90 days vs pdate

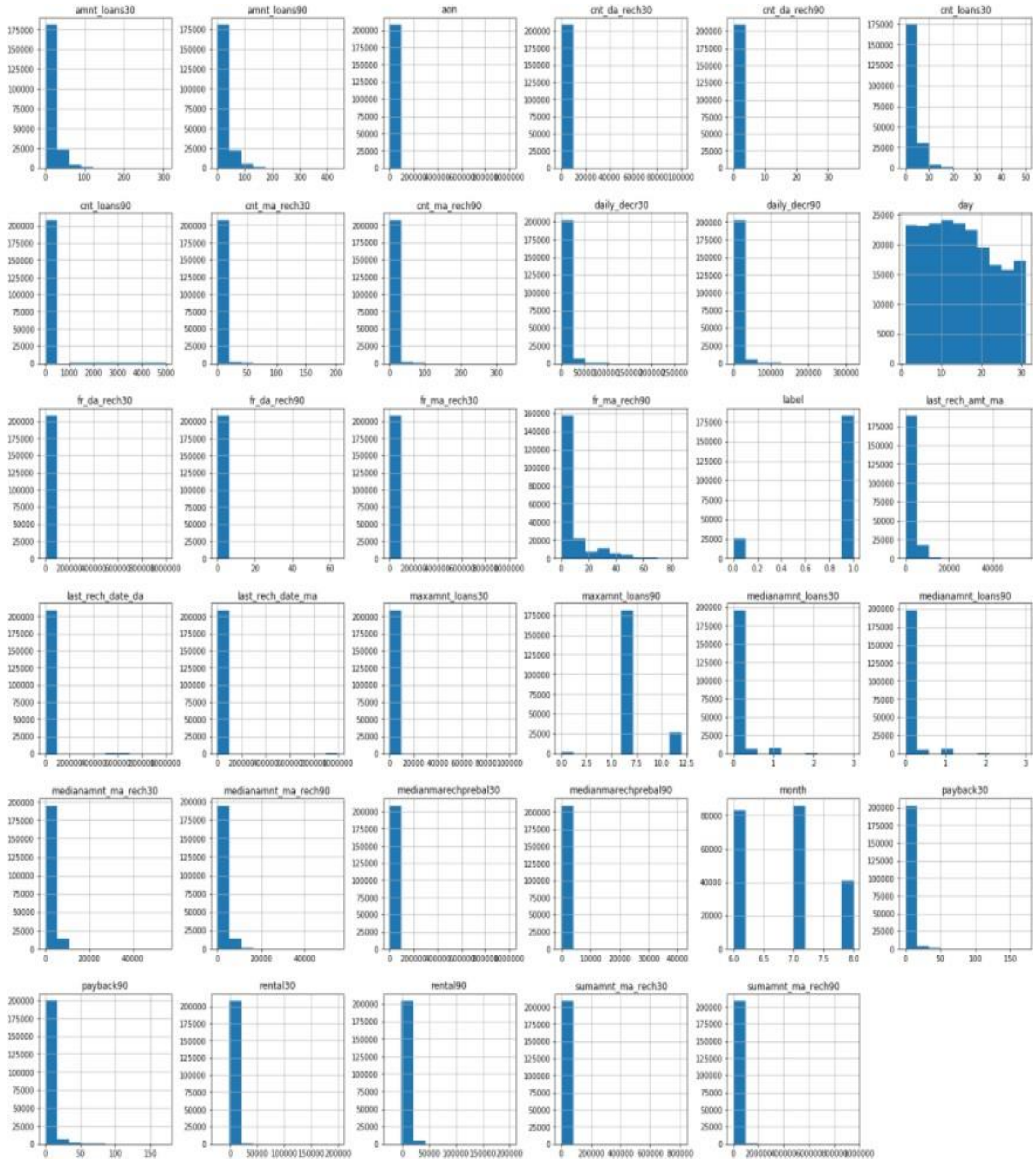


Flag indicators according to Month



Plotting Histogram

A histogram is used to represent data provided in a form of certain groups using various columns of the dataset. It is accurate method for the graphical representation of numerical data distribution in the given dataset.



CONCLUSION

- The Gradient Boosting Classifier model is working with highest accuracy score of 91.865110
- The cross_val_score of model is 91.845292 which again show that cross val score is better compared to other models in the table.
- The ROC_AUC_score of the model is showing as 77.656234 which show that model is learning with highest accuracy in the loop compared to other models.
- All these points proves that Gradient Boosting Classifier model is working best and can be considered as finalised model.

Learning Outcomes of the Study in respect of Data Science

1. **Credit risk modeling** – This allows banks to predict how their loans are going to be repaid and to foresee a defaulter based on history and credit report.

2. **Prediction of Customer Lifetime Value (CLV) –** Banks & MFI need to predict future revenues based on inputs from the past. This is best done using predictive data analytics to calculate the future values of each customer. This helps in segregating customers, identifying the ones with high future value, and investing more resources on them in terms of customer service, offers, and discounted pricing.

3. **Deployment of ML models** – The Machine learning models can also predict which banking tools individual members might use and recommend them so customers can make better financial decisions.

4. **Fraud Detection** – Fraud can be identified by taking measures and then implementing these measures to prevent it from happening again. Ideally, MFI & Banks want to find ways to prevent fraud from taking place, or, if that’s not possible, to detect it before significant damage is done. In the event that they are unable to prevent it in a timely fashion, however, fraud detection is the best bet for eradicating it from the environment and preventing a recurrence. The fraud detection can be implemented to safeguard the business proposal of the MFI’s.

Limitations of this work and Scope for Future Work

- The production & maintenance of artificial intelligence requires high costs as they are very complex machines, AI consists of advanced software programs that require regular updates to meet the needs of the changing environment in the Banking/MFI sector, In the case of critical failures, the procedure to reinstate the system and recover lost codes may require enormous time & cost.
- Although Artificial Intelligence & ML models can learn & improve, it still can’t make judgment calls, Humans can take individual circumstances and judgment calls into account when making decisions. So even though the model can help in the Business outcome but it cannot provide an accurate prediction for individuals.

References

1. Puja Padhi, “Information and Communication Technology in Microfinance Sector: Case Study of Three Indian MFIs”, IIM Kozhikode Society & Management Review, SAGE Journals, December 10, 2015

2. Shanthi Elizabeth Senthe, “Transformative Technology in Microfinance: Delivering Hope Electronically”, Research gate, Pittsburgh Journal of Technology Law and Policy, December 13, 2012

3. Hamed Rahimi Nohooji, “The Classification of the Applicable Machine Learning Methods in Robot Manipulators”, International Journal of Machine Learning and Computing, Vol. 2, No. 5, October 2012

4. Robert. J. Kauffman, “Information and Communication Technology and the Sustainability of Microfinance”, Electronic Commerce Research and Applications, September 2012

5. Saon Ray, “The Changing Role of Technological Factors in Explaining Efficiency in Indian Firms”, The Journal of Developing Areas, April 2011

6. M.Chandani, “Usage of Data Science in Fraud Detection” Henryharvin blog post, June 23, 2020