MICRO-CREDIT DEFAULTER PROJECT

Submitted by:

UTKARSH VARDHAN

ACKNOWLEDGMENT

I would like to express my deepest appreciation to the Team of Flip Robo for giving a full length description of the project and also handling queries at the same time. My SME or mentor Astha Mishra has helped me in the formation of this project where I was stuck with some problems and it was cured by my SME , therefore I would like to extend my sincere thanks to Astha Mishra.

I very much appreciate Data-Trained Education for its valuable advice, suggestion and experience they gave me during the training period, because of that only I was able to complete this Project.

There were some errors and problems occurred in between the project solution where I was able to rectify with the help of the internet or webs like kaggle and github etc.

INTRODUCTION

- Business Problem Framing

Companies having problems in selecting their customers as if they can pay back the loan amount in 5 days of issuance of loan. Loan amount = 5 and 10 Indonesian Rupiah on mobile balance, payback amount 6 & 12. This is a problem which is faced by various general companies as sometimes the loan amount is not recovered because these companies do not choose their customers properly likewise if they are or will be able to repay their loan amount.

Microfinance Companies give loans of small amounts, usually normal people forget to don't repay their small amount of loan back to the company as nothing will happen to them with such a small amount. Because of these things today our world has more than $ 70 Billion Outstanding Loans.

- Conceptual Background of the Domain Problem

A Microfinance Institution (MFI) is an organization that offers financial services to low income populations. Microfinance services (MFS) becomes very useful when targeting especially the unbanked poor families living in remote areas with not much sources of income. The 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 micro 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.

- Review of Literature

According to my analysis and understanding of the project it has been elevated that the organization has been going through with a problem of predicting their customers as there has been almost 12.5 % of the the data we have of defaulters which are not able to pay the loan amount of 6 or 12 with in 5 days of f the issuance. This research is done on python language which is able to predict the nature of the customers if they will be able to pay the loan amount back. Most of our models are predicting above 88% of accuracy score which is going to help the clients in predicting their customers for the organization.

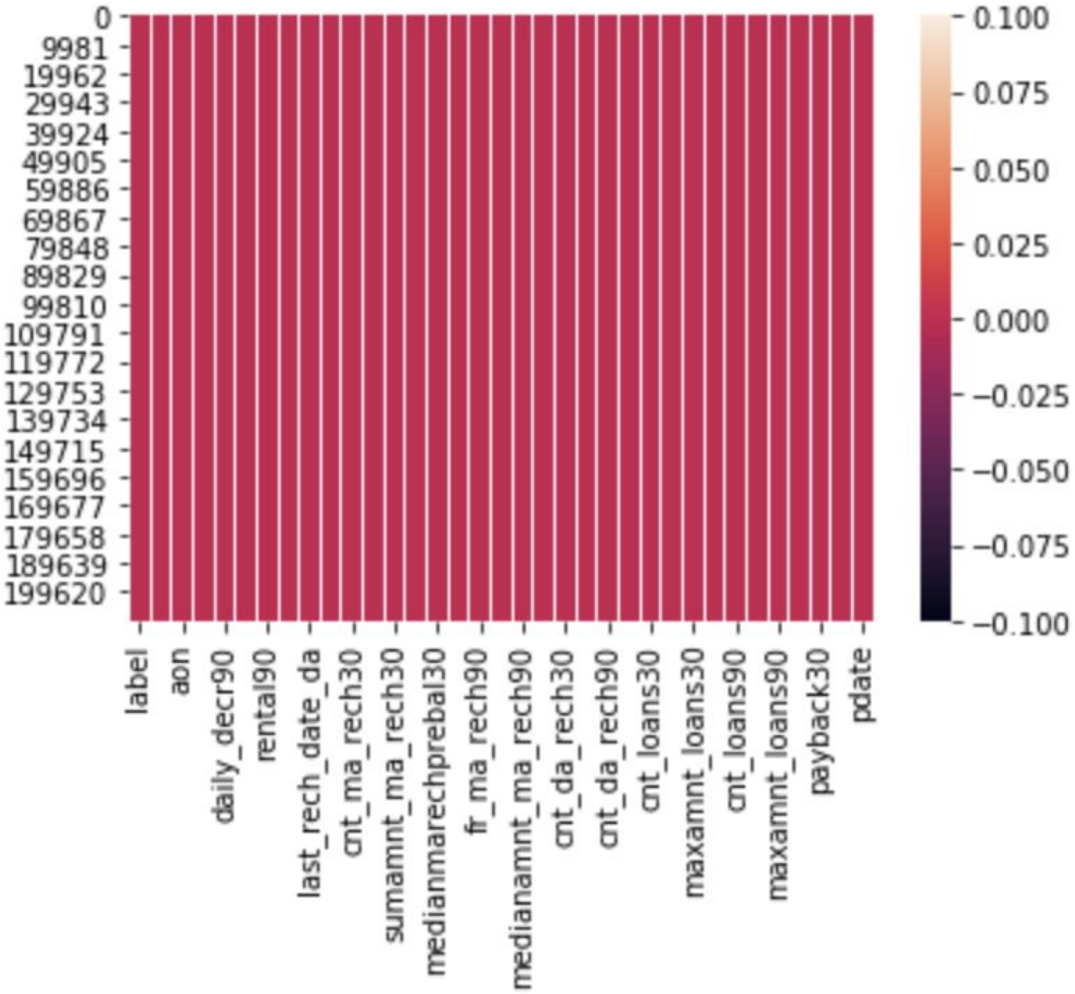
- Motivation for the Problem Undertaken

The motivation and the objective behind this project is to work on the data with all the techniques and enhance the result of the models so that our model predicts a higher accuracy score. This is because our clients will be able to predict the customers will be a defaulters of non- defaulters. This project focuses on providing the services and products to lower income families or poor customers, thus it helps in defining these customer preferences of paying back the loan or not.

Analytical Problem Framing

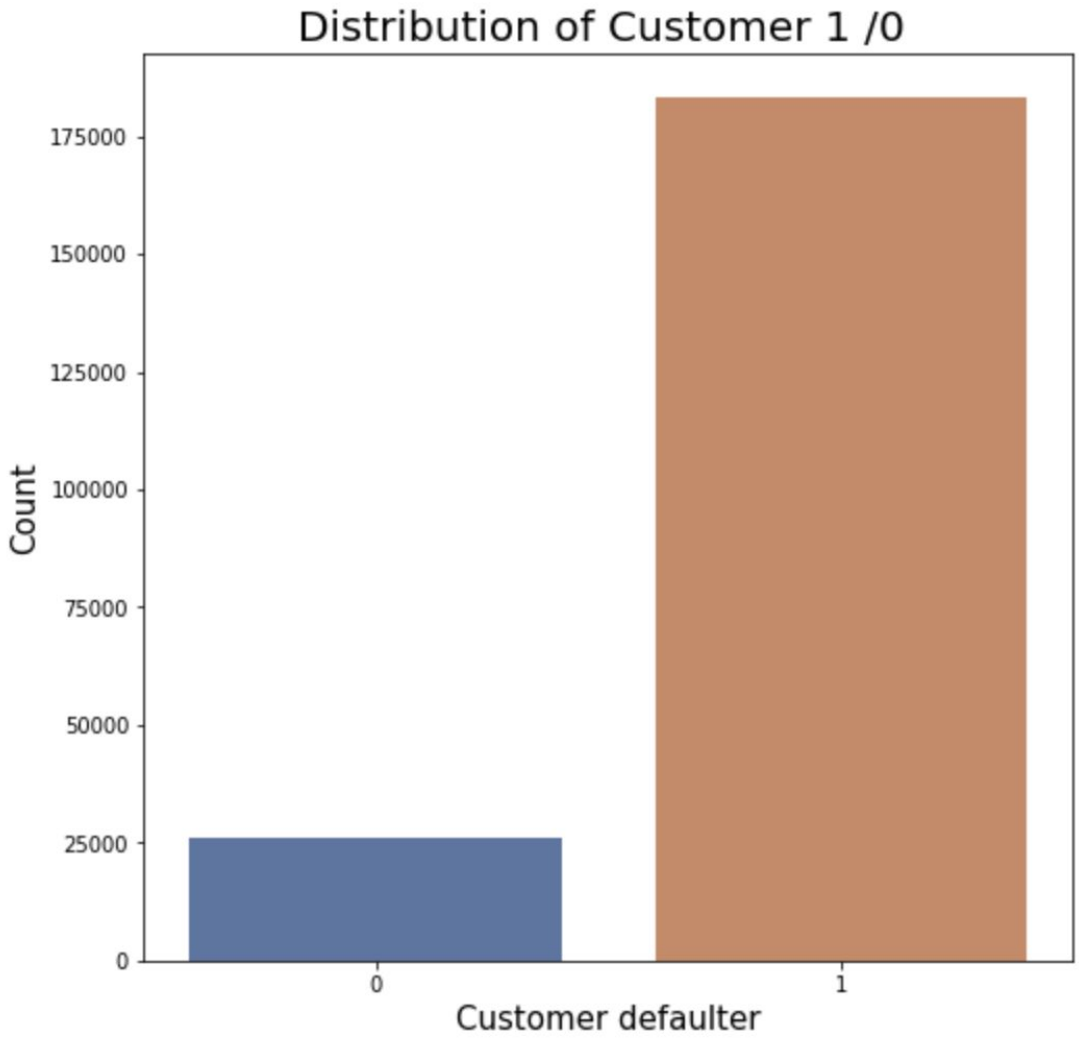
- Mathematical/ Analytical Modeling of the Problem
    1. Client - Telecom Industry from Indonesia.
    2. MFI is an organization which provides financial services to low income groups.
    3. They are collaborating with MFI to provide the micro credit on mobile balances.
    4. These credits to be paid back in 5 days otherwise he or she is a defaulter.
    5. There are two loan amounts : 5 and 10 Indonesian Rupiah which is having a pay back amount of 6 and 12 Rupiah.
    6. Data set has a shape of 209593 rows and 37 columns under which there is a column with a feature name of Unnamed: 0 which is removed as it is of no use in predicting the target variable.

7. There are no null values present in the dataset.



The above graph is here to show that if there are any null values in the above dataset, I have found out that the red color shows the '0' value here, it means that none of the value is empty(Null Values).

8. Target variable is “Label” 1 = Non Defaulters with 87.5% records , 0= Defaulters with 12.5% records.



9. As the data is expensive , we are not allowed to lose more than 8 to 10 percent of the data.

```
from scipy.stats import zscore
z_score=abs(zscore(df))
print(df.shape)
```

```
(209593, 35)
```

```
df1=df.loc[(z_score<4.5).all(axis=1)]
print(df1.shape)
```

```
(181809, 35)
```

After treating the dataset with z score in order to remove the outliers present in our dataset we have successfully removed almost less than 10% of the data from the original dataset, without disturbing the distribution of the Target variable which is same as before represented through the graph below :

- Data Sources and their formats

1) Data types are int64(12), object(1), float(21), datetime64(1).

```
<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   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  pdate                  209593 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(21), int64(12), object(1)
memory usage: 56.0+ MB
```

2) Data Description:

```
df.describe()
```

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	c
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	
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	55000.000000	

8 rows × 33 columns

The above description states the total count of the entry made that is rows, it also shows the

mean , min , max, std 25% and 75% such as , taking in consideration of the aon that is the age on cellular network in days which has 209593 count where mean or average days is 8112. Minimum days by the customer is -48 where maximum is 999860. It also states the 25% which is 246 and 75% is 982. As we could see that the 3rd quartile is greater than the mean of the data that shows the dataset is containing outliers which needs to be removed. 50% is nothing but the median of the data feature which is at 527 , likewise we could detect other variables too, that is rental 30 or last_rech_date_da etc

- Data Preprocessing Done

1) Synthesizing the date column which was done while loading the data file in python:

   #loading the given datasets:

   df1=pd.read_csv('Data file.csv',parse_dates=[-1])

   df1

2) Checking if there is any Unique data set, In such that WE encountered that “pcircle” feature has only one unique object array that is ‘UPW’ thus we have removed the column as it is not going to affect the target variable that is Label Data.
3) I have Dropped Unnamed: 0 column as it is used for numbering the rows and such data would not be used for predicting the label dataset.
4) Checking Missing Values :
   sns.heatmap(df.isnull())
   I have found out that the red color in the graph shows the '0' value here, it means that none of the value is empty(Null Values).

5) df.isnull().sum()

   In order to get more clarity we have taken out the sum of the total Null Values down which is also giving us the same output that is , ' 0 ' .
6) Enhancing Date column:

   #making seperate columns of year, month and day:

   df['year'] = pd.DatetimeIndex(df['pdate']).year

   df['month'] = pd.DatetimeIndex(df['pdate']).month

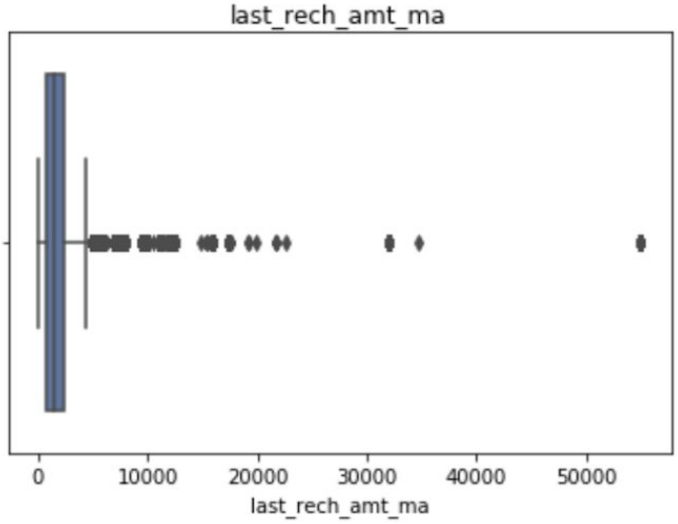
   df['Day'] = pd.DatetimeIndex(df['pdate']).day

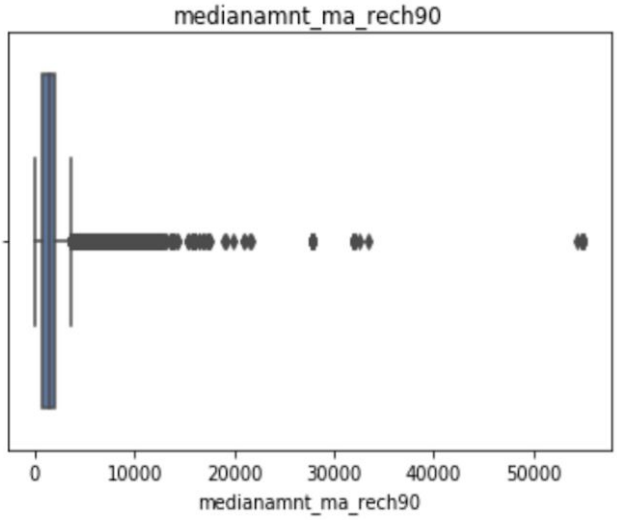
   df.head()

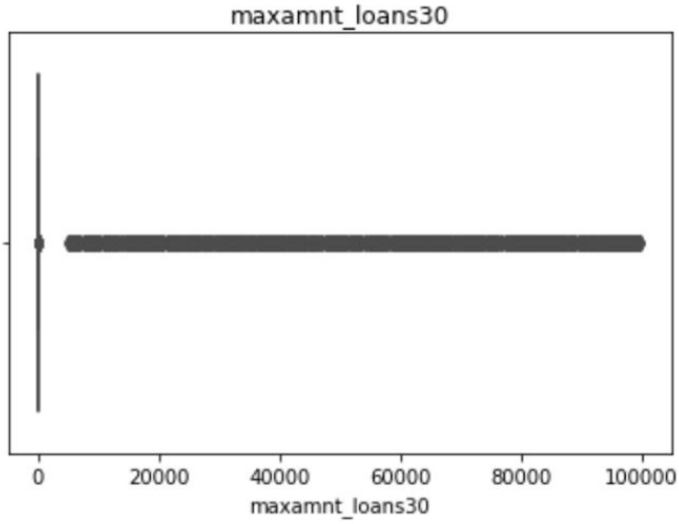
   With this technique I was able to add three different columns of day , month and year but as we could see that the year is same in all the rows so we will remove the columns for “year” and “pdate” which is of no use now.

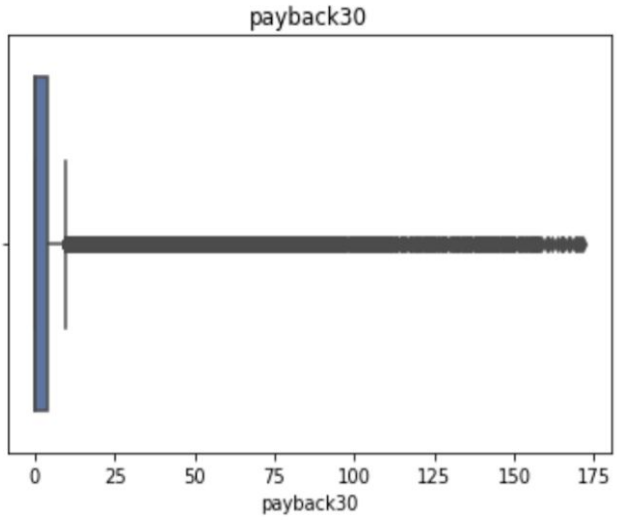
7) Removing Outliers from the data: The plot boxes situated below are representing the minimum value , maximum value and showing the mean value for the variables. In most of the variables. The graph shows some pointers above or below the graph which is used to detect if there are any outliers present in the datasets or not. If the detection is

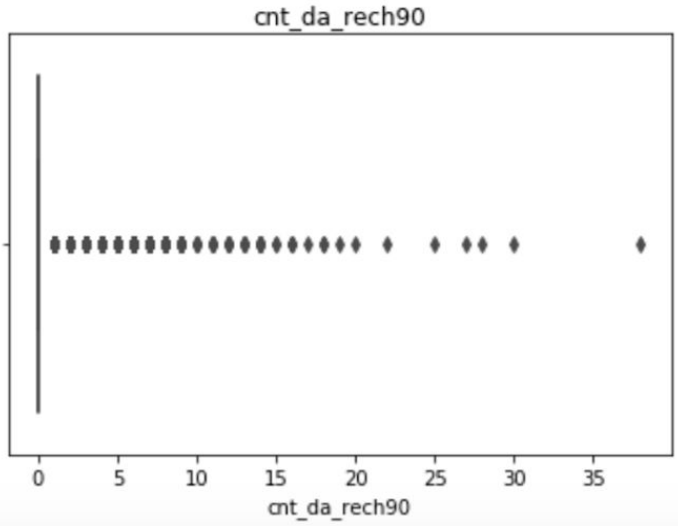
accurate then we should be able to treat them in order to get the perfect outcome or result.Outliers here are the observations that we find at the abnormal distance from other values.

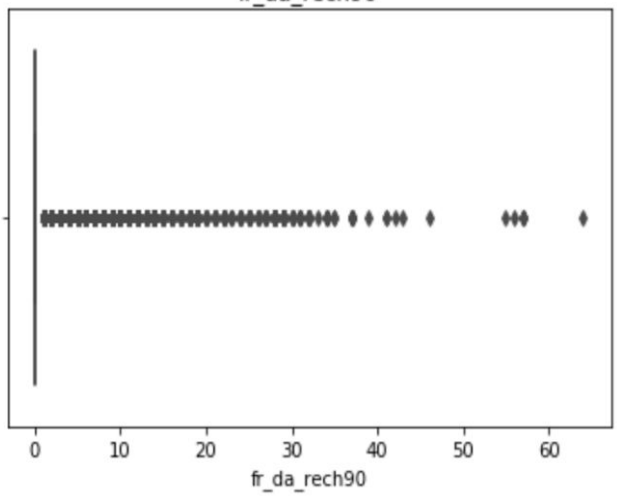












Treatment of the Outliers Data:

```
one_hot.label.value_counts(normalize=True)
```

```
1    0.875177
0    0.124823
Name: label, dtype: float64
```

```
print("number of erroneous values in th columns daily_dec30 and daily_decr90 ===>>", len(one_hot.loc[(one_hot.daily_decr30<0) | (one_hot.daily_decr90<0) | (one_hot.daily_decr30>100000) | (one_hot.daily_decr90>100000)]))
```

```
number of erroneous values in th columns daily_dec30 and daily_decr90 ===>> 2041
```

```
#Removing all the erroneous values from the data above:
dff=one_hot.loc[~((one_hot.daily_decr30<0) | (one_hot.daily_decr90<0) | (one_hot.daily_decr30>100000) | (one_hot.daily_decr90>100000) )]
dff.shape
```

```
1    0.881461
0    0.118539
Name: label, dtype: float64
```

```
print("number of erroneous values in the columns rental30 and rental90 ===>>", len(dff.loc[(dff.rental30>20000) | (dff.rental90>20000)]))
```

```
number of erroneous values in the columns rental30 and rental90 ===>> 5022
```

We could see that as the -tive value is more than the 10% of the data so we could loose more than 10% of the data through this. thats why we will not be removing the values here, but we will be treating the values which are above 20,000 mark.

```
dff=dff.loc[~((dff.rental30>20000) | (dff.rental90>20000))]
dff.shape
```

```
1    0.879855
0    0.120145
Name: label, dtype: float64
```

```
print("number of erroneous values in the columns last_rech_date_ma and last_rech_date_da ===>>", len(dff.loc[(dff.last_rech_date_ma<0) | (dff.last_rech_date_da<0) | (dff.last_rech_date_ma>9000) | (dff.last_rech_date_da>9000)]))
```

number of erroneous values in the columns last_rech_date_ma and last_rech_date_da ===>> 3317

```
dff=dff.loc[~((dff.last_rech_date_ma<0) | (dff.last_rech_date_da<0) | (dff.last_rech_date_ma>9000) | (dff.last_rech_date_da>9000))]
```

```
dff.shape
```

1 0.879029

0 0.120971

Name: label, dtype: float64

```
print("number of erroneous values in the columns medianmarechprebal30 and medianmarechprebal90 ===>>", len(dff.loc[(dff.medianmarechprebal30<0) | (dff.medianmarechprebal90<0) | (dff.medianmarechprebal30>18000)| (dff.medianmarechprebal90>18000)]))
```

number of erroneous values in the columns medianmarechprebal30 and medianmarechprebal90 ===>> 2747

```
dff=dff.loc[~((dff.medianmarechprebal30<0) | (dff.medianmarechprebal90<0) | (dff.medianmarechprebal30>18000) | (dff.medianmarechprebal90>18000))]
```

```
dff.shape
```

1 0.879933

0 0.120067

Name: label, dtype: float64

```
print("number of erroneous values in the column maxamnt_loans90 ===>>", len(dff.loc[(dff.maxamnt_loans90<=0) | (dff.maxamnt_loans30<=0)]))
```

number of erroneous values in the column maxamnt_loans90 ===>> 2951

```
dff=dff.loc[~((dff.maxamnt_loans90<=0)| (dff.maxamnt_loans30<=0))]
```

```
dff.shape
```

1 0.878102

0 0.121898

Name: label, dtype: float64

```
print("number of erroneous values in the columns medianamnt_ma_rech30 and medianamnt_ma_rech90 ===>>", len(dff.loc[(dff.medianamnt_ma_rech30<0) | (dff.medianamnt_ma_rech90<0) | (dff.medianamnt_ma_rech30>15000) | (dff.medianamnt_ma_rech90>15000)]))
```

number of erroneous values in the columns medianamnt_ma_rech30 and medianamnt_ma_rech90 ===>> 133

```
dff=dff.loc[~((dff.medianamnt_ma_rech30<0) | (dff.medianamnt_ma_rech90<0) | (dff.medianamnt_ma_rech30>15000) | (dff.medianamnt_ma_rech90>15000))]
```

dff.shape

1 0.878158

0 0.121842

Name: label, dtype: float64

print("number of erroneous values in the column aon ===>>",
len(dff.loc[(dff.aon<0) | (dff.aon>3000)]))

number of erroneous values in the column aon ===>> 3398

dff=dff.loc[~((dff.aon<0) | (dff.aon>3000))]

dff.shape

(189984, 37)

1 0.879858

0 0.120142

Name: label, dtype: float64

8) **MinMaxScaler**:

from sklearn.preprocessing import MinMaxScaler

scaler= MinMaxScaler (feature_range=(0,1),copy=True).fit(x)

df1_scaler = scaler.transform(x)

x = pd.DataFrame(df1_scaler)

x.head(10)

The **MinMaxScaler** is the probably the most famous scaling algorithm, and follows the following formula for each feature: xi–min(x)max(x)–min(x) It essentially shrinks the range such that the range is now between 0 and 1 (or -1 to 1 if there are negative values).

9) **PCA :**(PRINCIPAL COMPONENT ANALYSIS) As we can see in our final dataset that there are 35 columns present in order to transform the data without affecting any fields data lowering the columns number. x was assigned with 34 columns now after transforming it has become 15 columns.

```
from sklearn import decomposition
from sklearn.decomposition import PCA
# Create a Covariance Matrix
covar_matrix = PCA(n_components = 34) #we have 34 features
#Calculate Eigenvalues
covar_matrix.fit(x)
variance = covar_matrix.explained_variance_ratio_ #calculate variance ratios
```

```
var=np.cumsum(np.round(covar_matrix.explained_variance_ratio_, decimals=3)*100)
var #cumulative sum of variance explained with [n] features
array([42.4, 67. , 76.9, 83.5, 87.8, 91.3, 93.3, 94.8, 95.7, 96.5, 97.1,
       97.6, 98.1, 98.5, 98.8, 99. , 99.2, 99.4, 99.6, 99.7, 99.8, 99.8,
       99.8, 99.8, 99.8, 99.8, 99.8, 99.8, 99.8, 99.8, 99.8, 99.8, 99.8,
       99.8])
plt.ylabel('% Variance Explained')
plt.xlabel('# of Features')
plt.title('PCA Analysis')
plt.ylim(34,100.5)
plt.style.context('seaborn-whitegrid')
plt.plot(var)
```

Based on the plot above it's clear we should pick 15 features at 99% variance.

```
pca=PCA(n_components=15)
x=pca.fit_transform(x)
x.shape
```

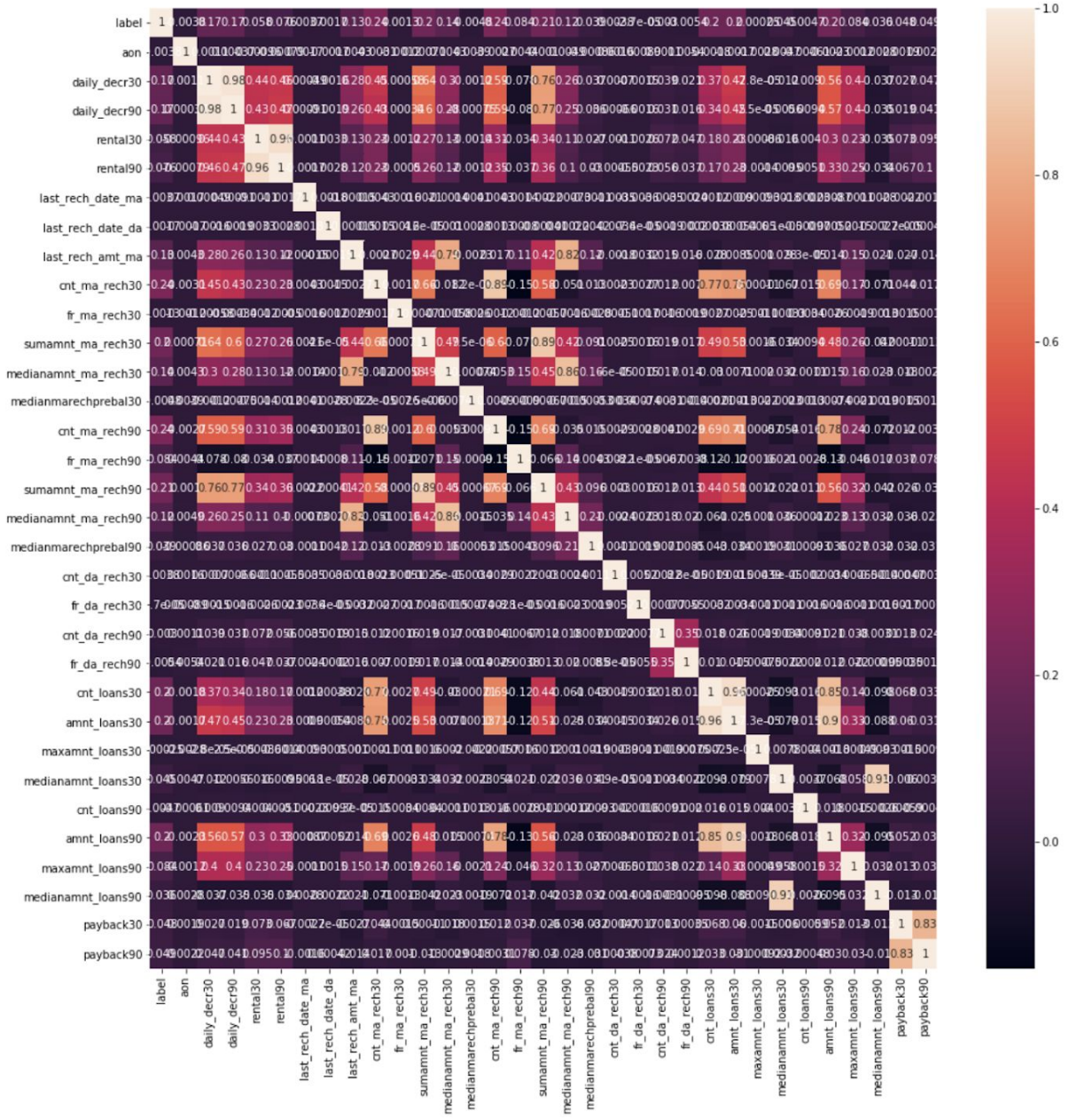
Final shape of the Data = (189984, 15)

- Data Inputs- Logic- Output Relationships

Data input are the x variable which are independent variable through which we will be able to predict the output data that is the targeted variable or Lable data or y variable :

To find the relationship between the output and input data :

```
corr_hmap=df.corr()
plt.figure(figsize=(16,16))
sns.heatmap(corr_hmap,annot=True)
plt.show()
```



With the help of the above graph we will be able to see if the x variables are correlating with the y or targeted variable or not. It has been represented by a heat map with numbering from 1 to -1 where 1 represents high correlation and -1 represents the least correlation between the variables.

In order to get more clarity of the above data it has been represented via correlation matrix which is sorting values in descending order that is :

```
corr_matrix = df.corr()
print(corr_matrix["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
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
cnt_loans90              0.004733
cnt_da_rech30            0.003827
last_rech_date_ma        0.003728
cnt_da_rech90            0.002999
last_rech_date_da        0.001711
fr_ma_rech30             0.001330
maxamnt_loans30          0.000248
fr_da_rech30            -0.000027
aon                     -0.003785
medianmarechprebal30    -0.004829
fr_da_rech90            -0.005418
Name: label, dtype: float64
```

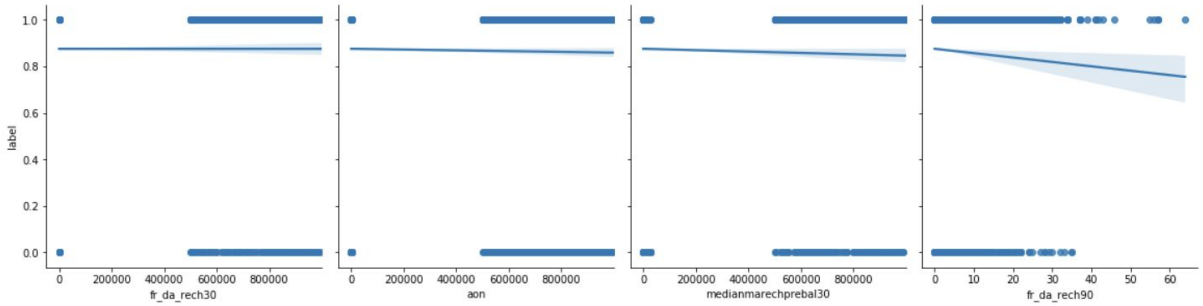
In the above presentation it is shown that the all the features showing positive correlation with respect to the targeted variable or Label data except the last 4 features which is showing somewhat negative correlation thus we will still be taking them in to our consideration as we are not allowed to remove or lose data more than 10 % hene we would be removing only two columns where the negative relation is high that is :

dff.drop("fr_da_rech90", axis=1,inplace=True)

dff.drop("fr_da_rech30", axis=1,inplace=True)

```
sns.pairplot(df, x_vars=['fr_da_rech30','aon','medianmarechprebal30','fr_da_rech90'],
y_vars='label', size=4, aspect=1, kind='reg')
```

```
<seaborn.axisgrid.PairGrid at 0x7fc600d0f190>
```



- Hardware and Software Requirements and Tools Used

import numpy as np: numpy is used in the dataset for working with the arrays, working in domain of linear algebra, fourier transform, and matrices
import pandas as pd : Pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.
import matplotlib.pyplot as plt: It is used to check the missing values in our dataset also used for the histogram which is to detect the count of various features lying in different groups .
from sklearn.linear_model import LogisticRegression: It helps in predicting the model with logistic regression where , 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).
from sklearn.metrics import classification_report: A Classification report is used to measure the quality of predictions from a classification algorithm. ... The report 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.

from sklearn.tree import DecisionTreeClassifier: The decision tree classifier (Pang-Ning et al., 2006) creates the classification model by building a decision tree. Each node in the tree specifies a test on an attribute, each branch descending from that node corresponds to one of the possible values for that attribute.
from sklearn.metrics import confusion_matrix: A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. ... The classifier made a total of 165 predictions (e.g., 165 patients were being tested for the presence of that disease).
from sklearn.metrics import accuracy_score : In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true.
from sklearn.metrics import roc_curve: ROC is a plot of signal (True Positive Rate) against noise (False Positive Rate). ... The model performance is determined by looking at the area under the ROC curve (or AUC). The best possible AUC is 1 while the worst is 0.5 (the 45 degrees random line).
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score: ROC stands for curves receiver or operating characteristic curve. It illustrates in a binary classifier system the discrimination threshold created by plotting the true positive rate vs false positive rate. ... The roc_auc_score always runs from 0 to 1, and is sorting predictive possibilities.
import seaborn as sns : Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
from sklearn.model_selection import GridSearchCV: GridSearchCV is a library function that is a member of sklearn's model_selection package. It helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, you can select the best parameters from the listed hyperparameters.
from sklearn.preprocessing import StandardScaler: Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance). It is a standardized feature which is extracted by subtracting the mean and then scaling to unit variance.Unit variance means dividing all the values by the standard deviation. StandardScaler makes the mean of the distribution 0. About 68% of the values will lie be between -1 and 1
from sklearn.model_selection import train_test_split: train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. ... By default, Sklearn train_test_split will make random partitions for the two subsets. However, you can also specify a random state for the operation.
from sklearn.model_selection import cross_val_score: It takes the features df and target y , splits into k-folds (which is the cv parameter), fits on the (k-1) folds and evaluates on the last fold. It does this k times, which is why you get k values in your output array.
from sklearn.naive_bayes import GaussianNB: A 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.

from sklearn.svm import SVC: The objective of a Linear SVC (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes, your data. From there, after getting the hyperplane, you can then feed some features to your classifier to see what the "predicted" class is.

from scipy.stats import z score: The standard score (more commonly referred to as a z-score) is a very useful statistic because it (a) allows us to calculate the probability of a score occurring within our normal distribution and (b) enables us to compare two scores that are from different normal distributions.

from sklearn.neighbors import KNeighborsClassifier: The K-nearest neighbors (KNN) algorithm is a type of supervised machine learning algorithms. KNN is extremely easy to implement in its most basic form, and yet performs quite complex classification tasks. ... KNN is a non-parametric learning algorithm, which means that it doesn't assume anything about the underlying data.

from sklearn.model_selection import GridSearchCV : GridSearchCV is a library function that is a member of sklearn's model_selection package. It helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, you can select the best parameters from the listed hyperparameters

from sklearn.ensemble import AdaBoostRegressor : There are primarily three hyperparameters that you can tune to improve the performance of AdaBoost: The number or estimators, learning rate and maximum number of splits. It's really hard to give general guidelines for optimal values for these parameters, as it always depends on the problem and the data. Default it uses Decision tree classifier.

from sklearn.ensemble import RandomForestClassifier : A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True(default), otherwise the whole dataset is used to build each tree.

from sklearn.metrics import r2_score: A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0. ... Parameters y_truearray-like of shape (n_samples,) or (n_samples, n_outputs) Ground truth (correct) target values.

import warnings

warnings.filterwarnings('ignore') : The warn() function defined in the ' warning ' module is used to show warning messages. The warning module is actually a subclass of Exception which is a built-in class in Python. filter_none. # program to display a warning message. import warnings.

from sklearn.externals import joblib: To save the file in object format.

Listing down the hardware and software requirements along with the tools, libraries and packages used. Describe all the software tools used along with a detailed description of tasks done with those tools.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

With the Statistical method it is derived that the 4 features are having somewhat negativity in correlation thus we will still be taking them into consideration as we could not lose more data. It is also seen that with the help of the bar graph there are no defaulters in the month of 8th and therefore in the month of 6th and 7th we could see that the defaulters of the loan is ranging from 10000 to 12000 in both months. Here after we could see that most of the customers pay their loan back on the 5th , 6th , 7th , and 8th day of the month. Maximum defaulters which have not paid the loan or the total amount of loans taken by the users in last 90 days is maximum for 6 Rupiah i.e. The amount of 6 Rupiah is the amount where maximum number of users have paid and not paid the loan.It has also come to our notice that the maximum amount of loan in the last 90 days is maximum for 6 Rupiah with respect to the 12 Rupiah.

If we see the relation between the aon (age on cellular network in days) and maximum amount of loan in 90 days mostly above 6000 or 8000 days whatever the loan amount is it has been paid back that is they are non defaulters on the other hand mostly below 6000 of aon people are defaulting in the payment of the loans. There may be few defaulters ranging in between 8000 to 10000 but they will be negligible, so Statistically with respect to aon we could identify the customers who will be able to repay their loan amount.

Analytical approach is done by predicting the target variable that is Label data before that we have to Assign the x variable and y variable from the data given to us. Now we going to put the test size and random state for the data : x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=.20,random_state=42,stratify=y)

print(x_train.shape,x_test.shape)

(145447, 34) (36362, 34)

print(y_train.shape,y_test.shape)

(145447, 1) (36362, 1)

Model Testing in finding the Accuracy score, Recall score of target variable = “0” list of defaulters , RocAuc Score and the Cross val score:

Model	Accuracy Score	Recall score “0”	Roc Auc Score	Cross Val Score
LogisticRegression	71.87%	84%	77.13%	71.63%
GaussianNB	65.67%	79%	71.45%	65.11%
DecisionTreeClassifier	84.77%	40%	65.45%	84.82%
RandomForestRegressor	89.84%	--	--	26.35%
SVC	**76.39%**	**84%**	**79.86%**	**75%**
KNeighborsClassifier	89%	30%	63.66%	96.96%
AdaBoostClassifier	89.26%	28%	62.62%	97.66%
RandomForestClassifier	**77.49%**	**82%**	**79.44%**	**76.98%**

- Testing of Identified Approaches (Algorithms)

Listing down all the algorithms used for the training and testing:

```
from sklearn.metrics import
classification_report,confusion_matrix,accuracy_score,roc_curve,auc
from sklearn.model_selection import train_test_split,cross_val_score
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=.30,random_state=42,stratify=y)


print(x_train.shape,x_test.shape)
print(y_train.shape,y_test.shape)
```

Logistic Regression

Gaussian NB

Decision Tree Classifier

```
LOR=LogisticRegression(class_weight="balanced")
GNB=GaussianNB()
DTC=DecisionTreeClassifier(random_state=10)

models= [ ]
models.append(('LogisticRegression',LOR))
models.append(('GaussianNB',GNB))
models.append(('DecisionTreeClassifier',DTC))

Model = []
score = []
cvs = []
rocscore = []
for name,model in models:
    print('****************',name,'****************')
    print('\n')
    Model.append(name)
    model.fit(x_train,y_train)
    print(model)
    pred=model.predict(x_test)
    print('\n')
    AS= accuracy_score(y_test,pred)
    print('ACCURACY SCORE IS = ',AS)
    score.append(AS*100)
    print('\n')
    sc=cross_val_score(model,x,y,cv=10,scoring='accuracy').mean()
    print('CROSS_VAL_SCORE = ',sc)
```

```
    cvs.append(sc*100)
    print('\n')
    false_positive_rate,true_positive_rate,thresholds=roc_curve(y_test,pred)
    roc_auc= auc(false_positive_rate,true_positive_rate)
    print('ROC_AUC_SCORE = ',roc_auc)
    rocscore.append(roc_auc*100)
    print('\n')
    print('CLASSIFICATION REPORT = ',classification_report(y_test,pred))
    print('\n')
    cm=confusion_matrix(y_test,pred)
    print('CONFUSION MATRIX',cm)
    print('\n')
    plt.figure(figsize=(10,40))
    plt.subplot(911)
    plt.title(name)
    print(sns.heatmap(cm,annot=True))
    plt.subplot(912)
    plt.title(name)
    plt.plot([0,1],[0,1],'k--')
    plt.plot(false_positive_rate,true_positive_rate,label='AUC= %.2f'% roc_auc)
    plt.xlabel('false positive rate')
    plt.ylabel('true positive rate')
    plt.title(name)
    plt.show()
    print('\n\n')

from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import roc_auc_score
```

RandomForestRegressor

```
from sklearn.ensemble import RandomForestRegressor
rr=RandomForestRegressor(bootstrap=True,max_features='auto',min_samples_split=2,n_
estimators=300)
rr.fit(x_train, y_train)
rr.score(x_train,y_train)
pred=rr.predict(x_test)

from math import sqrt
print("Test Results for Random Forest Regressor Model:")
print(50 * '-')
print("Root mean squared error: ", sqrt(mean_squared_error(y_test,pred)))
print("R-squared: ", r2_score(y_test,pred))
```

SVC

```
svc=SVC(kernel='rbf',class_weight="balanced")
svc.fit(x_train,y_train)
svc.score(x_train,y_train)
predsvc=svc.predict(x_test)
print(accuracy_score(y_test,predsvc))
print(confusion_matrix(y_test,predsvc))
print(classification_report(y_test,predsvc))
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, predsvc)
print('Accuracy: %f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, predsvc)
print('Precision: %f' % precision)
# recall: tp / (tp + fn)
```

```
recall = recall_score(y_test, predsvc)
print('Recall: %f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, predsvc)
print('F1 score: %f' % f1)
# ROC AUC
auc = roc_auc_score(y_test, predsvc)
print('ROC AUC: %f' % auc)
```

KNeighborsClassifier

```
KNN=KNeighborsClassifier(n_neighbors=14,algorithm='auto',weights="distance")
KNN.fit(x_train,y_train)
KNN.score(x_train,y_train)
predKNN=KNN.predict(x_test)
print(accuracy_score(y_test,predKNN))
print(confusion_matrix(y_test,predKNN))
print(classification_report(y_test,predKNN))
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, predKNN)
print('Accuracy: %f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, predKNN)
print('Precision: %f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, predKNN)
print('Recall: %f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, predKNN)
print('F1 score: %f' % f1)
```

```
# ROC AUC
auc = roc_auc_score(y_test, predKNN)
print('ROC AUC: %f' % auc)
```

K-Fold -

```
from sklearn.model_selection import KFold
from sklearn import tree
crossvalidation=KFold(n_splits=10,shuffle=True,random_state=10)
for depth in range (1,15):
    tree_classifier=tree.DecisionTreeClassifier(max_depth=depth,random_state=10)
    if tree_classifier.fit(x,y).tree_.max_depth<depth:
        break
    score=np.mean(cross_val_score(tree_classifier,x,y,scoring='accuracy',
cv=crossvalidation,n_jobs=1))
    print(depth, score)

from sklearn.ensemble import AdaBoostClassifier
ada=AdaBoostClassifier()
search_grid={'n_estimators':[100,200,500,1000],'learning_rate':[0.1,1]}
search=GridSearchCV(estimator=ada,param_grid=search_grid,scoring='accuracy',n_jobs
=1,cv=crossvalidation)
search.fit(x,y)
search.best_params_

score=np.mean(cross_val_score(ada,x,y,scoring='accuracy',cv=crossvalidation,n_jobs=1))
score

search.best_score_
```

AdaBoostClassifier - Decision tree Classifier

```
# base estimator = Decision tree Classifier
from sklearn.ensemble import AdaBoostClassifier
ADA=AdaBoostClassifier(n_estimators=500 , learning_rate=1)
ADA.fit(x_train,y_train)
ADA.score(x_train,y_train)
predADA=ADA.predict(x_test)
print(accuracy_score(y_test,predADA))
print(confusion_matrix(y_test,predADA))
print(classification_report(y_test,predADA))
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, predADA)
print('Accuracy: %f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, predADA)
print('Precision: %f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, predADA)
print('Recall: %f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, predADA)
print('F1 score: %f' % f1)
# ROC AUC
auc = roc_auc_score(y_test, predADA)
print('ROC AUC: %f' % auc)
```

GridSearchCv:

```
rfc=RandomForestClassifier(random_state=52)
param_grid = {
```

```
    'n_estimators': [100, 200, 500],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : [6,8,10]
}
CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv=3)
CV_rfc.fit(x_train, y_train)
CV_rfc.best_params_
```

RandomForestClassifier

```
from sklearn.metrics import f1_score
from sklearn.ensemble import RandomForestClassifier
R_forest= RandomForestClassifier(criterion='gini',
n_estimators=200, max_features=
'auto',max_depth=10,random_state=52,class_weight="balanced")
modelR= R_forest.fit(x_train, y_train)

# Predictions
pred_2 = modelR.predict(x_test)

print ("The accuracy of RandomForestClassifier : ",accuracy_score(y_test, pred_2))
print ("The f1 score of RandomForestClassifier : ", f1_score(y_test, pred_2, average =
'binary'))



print(confusion_matrix(y_test,pred_2))
print(classification_report(y_test,pred_2))

# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, pred_2)
print('Accuracy: %f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, pred_2)
```

```
print('Precision: %f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, pred_2)
print('Recall: %f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, pred_2)
print('F1 score: %f' % f1)
# ROC AUC
auc = roc_auc_score(y_test, pred_2)
print('ROC AUC: %f' % auc)
```

#Cross val score :

```
print("Mean accuracy for R_forest classifier
",cross_val_score(R_forest,x,y,cv=3,scoring="recall").mean())
print("Standard Deviation accuracy for R_forst classifier
",cross_val_score(R_forest,x,y,cv=3,scoring="recall").std())

print("Mean accuracy score for AdaBoostClassifier
",cross_val_score(ADA,x,y,cv=3,scoring="recall").mean())
print("Standard Deviation accuracy for AdaBoostClassifier
",cross_val_score(ADA,x,y,cv=3,scoring="recall").std())
print()
print()
print("Mean accuracy for KNeighborsClassifier
",cross_val_score(KNN,x,y,cv=3,scoring="recall").mean())
print("Standard Deviation accuracy for KNeighborsClassifier
",cross_val_score(KNN,x,y,cv=3,scoring="recall").std())
print()
print()
print("Mean  for RandomForestRegressor ",cross_val_score(rr,x,y,cv=3).mean())
print("Standard Deviation  for RandomForestRegressor
",cross_val_score(rr,x,y,cv=3).std())
```

```
        print()
        print()
        print("Mean accuracy score for SVC
",cross_val_score(svc,x,y,cv=3,scoring="recall").mean())
        print("Standard Deviation accuracy score for SVC
",cross_val_score(svc,x,y,cv=3,scoring="recall").std())
```

- Run and Evaluate selected models

Results observed over different evaluation metrics:

********** **LogisticRegression** **********

LogisticRegression(class_weight='balanced')

ACCURACY SCORE IS = 0.7187872833181276

CROSS_VAL_SCORE = 0.716365600572026

ROC_AUC_SCORE = 0.7713494924183296

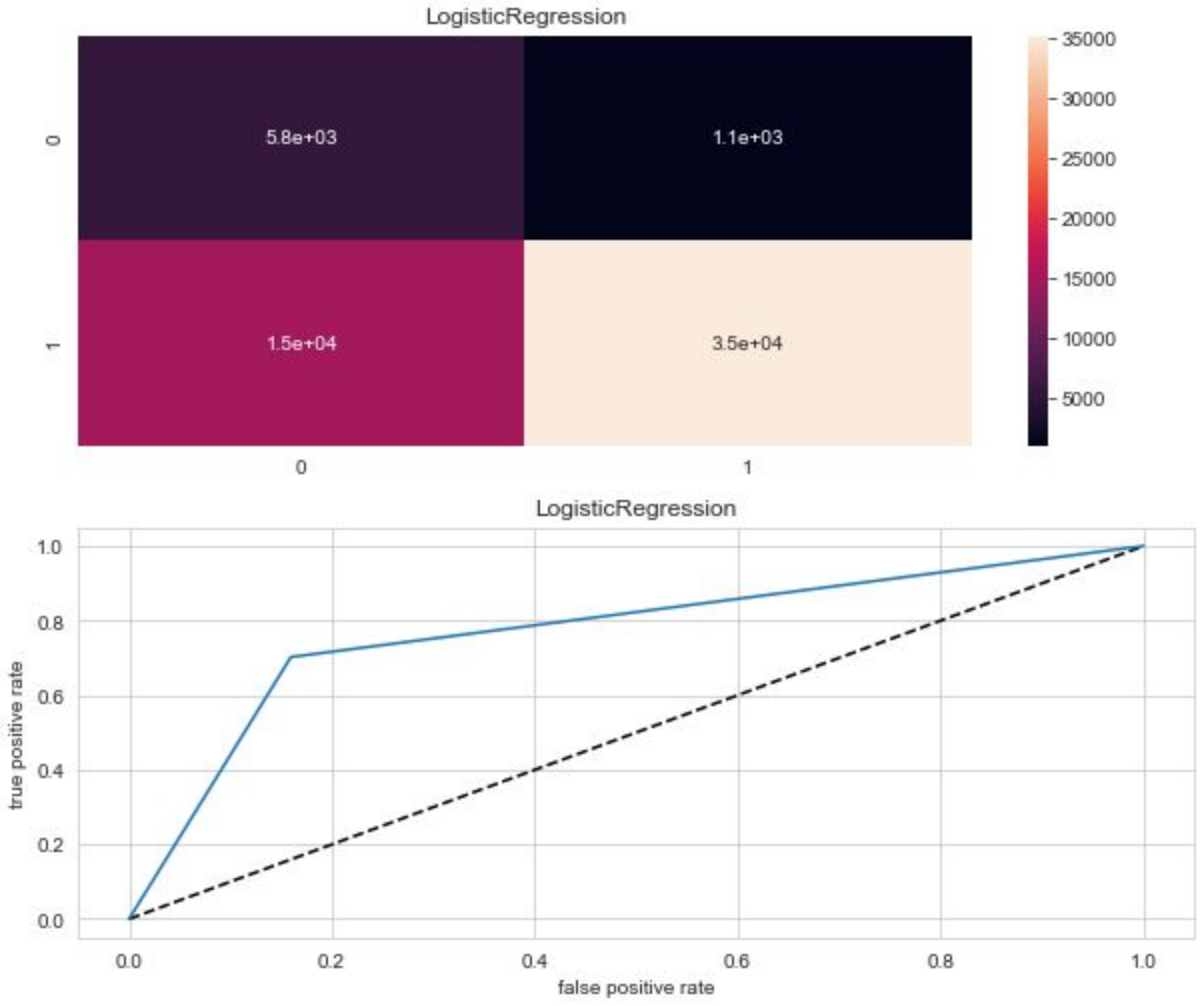
CLASSIFICATION REPORT = precision recall f1-score support

0	0.28	0.84	0.42	6848
1	0.97	0.70	0.81	50148
accuracy			0.72	56996

macro avg	0.62	0.77	0.62	56996
weighted avg	0.89	0.72	0.77	56996

CONFUSION MATRIX [[5756 1092]
 [14936 35212]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)



********** **GaussianNB** **********

GaussianNB()

ACCURACY SCORE IS = 0.6567302968629377

CROSS_VAL_SCORE = 0.6511180117020248

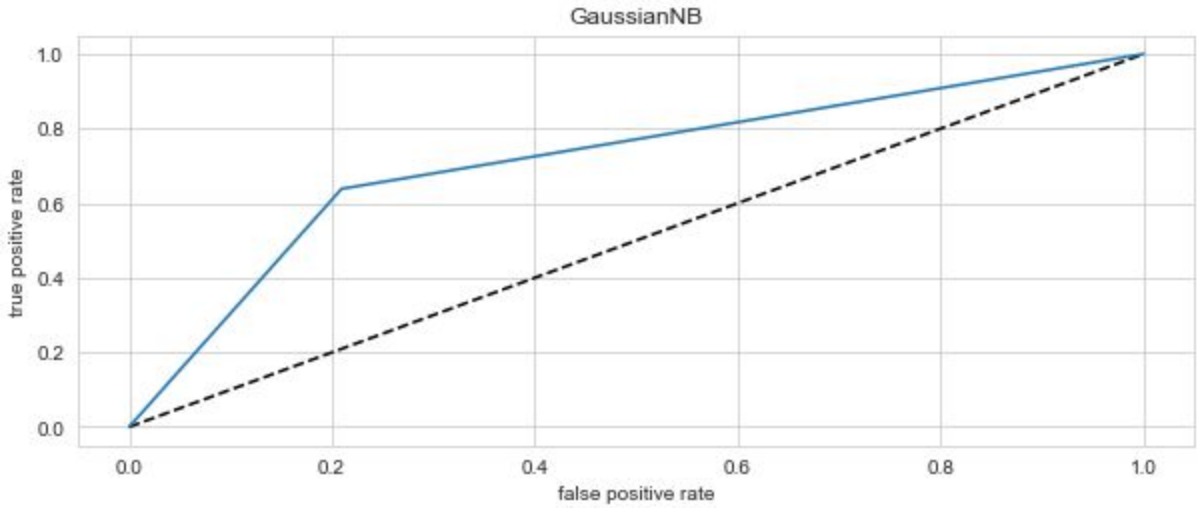
ROC_AUC_SCORE = 0.7145229909188429

CLASSIFICATION REPORT = precision recall f1-score support

0	0.23	0.79	0.36	6848
1	0.96	0.64	0.77	50148
accuracy			0.66	56996
macro avg	0.59	0.71	0.56	56996
weighted avg	0.87	0.66	0.72	56996

CONFUSION MATRIX [[5414 1434]
[18131 32017]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)



********** **DecisionTreeClassifier** **********

DecisionTreeClassifier(random_state=10)

ACCURACY SCORE IS = 0.8477261562214893

CROSS_VAL_SCORE = 0.8482082747236641

ROC_AUC_SCORE = 0.654798531160848

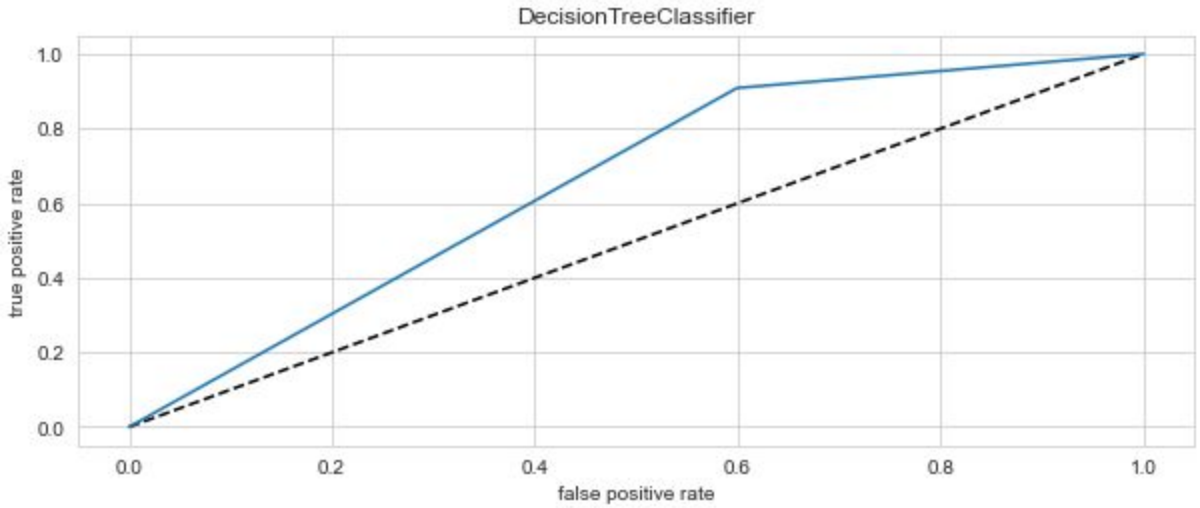
CLASSIFICATION REPORT = precision recall f1-score support

0	0.37	0.40	0.39	6848
1	0.92	0.91	0.91	50148
accuracy			0.85	56996
macro avg	0.65	0.65	0.65	56996
weighted avg	0.85	0.85	0.85	56996

CONFUSION MATRIX [[2745 4103]
 [4576 45572]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)


DecisionTreeClassifier
2.7e+03
4.1e+03
4.6e+03
4.6e+04
0
1
0
1
45000
40000
35000
30000
25000
20000
15000
10000
5000


DecisionTreeClassifier
true positive rate
false positive rate
0.0
0.2
0.4
0.6
0.8
1.0

RandomForestRegressor

Accuracy score = 0.8984368903433945

Test Results for Random Forest Regressor Model:

Root mean squared error: 0.27785398052823945
R-squared: 0.2696943972877993

SVC

0.7639834374342059

[[5781 1067]

[12385 37763]]

	precision	recall	f1-score	support
0	0.32	0.84	0.46	6848
1	0.97	0.75	0.85	50148
accuracy			0.76	56996
macro avg	0.65	0.80	0.66	56996
weighted avg	0.89	0.76	0.80	56996

Accuracy: 0.763983

Precision: 0.972521

Recall: 0.753031

F1 score: 0.848817

ROC AUC: 0.798610

KNeighborsClassifier

0.8899747350691276

[[2077 4771]

[1500 48648]]

	precision	recall	f1-score	support
0	0.58	0.30	0.40	6848
1	0.91	0.97	0.94	50148
accuracy			0.89	56996
macro avg	0.75	0.64	0.67	56996
weighted avg	0.87	0.89	0.87	56996

Accuracy: 0.889975

Precision: 0.910687

Recall: 0.970089

F1 score: 0.939450

ROC AUC: 0.636694

KFold

1 0.8798583395169967

2 0.8798583395169967

3 0.8798583395169967

4 0.8798899206806065

5 0.8803794309329758

6 0.885985164771251

7 0.886690490816054

8 0.8873115814003232

9 0.8886169459520371

10 0.8880116271100335

11 0.8873168295973779

12 0.8853798348471651

13 0.883348091896238

14 0.8808215633447855

You can see that the most accurate decision tree had a depth of 9. Before that there was a general decline in accuracy all the above and below depths.

We now can determine if the adaBoost model is better based on whether the accuracy is above 88.86 %. Before we develop the AdaBoost model, we need to tune several hyperparameters in order to develop the most accurate model possible.

{'learning_rate': 1, 'n_estimators': 500}

0.8887801182525765

0.8919698534562528

AdaBoostClassifier - Decision tree Classifier

0.8926942241560811

[[1887 4961]

 [1155 48993]]

	precision	recall	f1-score	support
0	0.62	0.28	0.38	6848
1	0.91	0.98	0.94	50148
accuracy			0.89	56996
macro avg	0.76	0.63	0.66	56996
weighted avg	0.87	0.89	0.87	56996

Accuracy: 0.892694

Precision: 0.908051

Recall: 0.976968

F1 score: 0.941250

ROC AUC: 0.626262

GridSearchCv:

{'max_depth': 10, 'max_features': 'auto', 'n_estimators': 200}

RandomForestClassifier

The accuracy of RandomForestClassifier : 0.7852129974033265
The f1 score of RandomForestClassifier : 0.8649769483599143
[[5542 1306]
[10936 39212]]

	precision	recall	f1-score	support
0	0.34	0.81	0.48	6848
1	0.97	0.78	0.86	50148
accuracy			0.79	56996
macro avg	0.65	0.80	0.67	56996
weighted avg	0.89	0.79	0.82	56996

Accuracy: 0.785213
Precision: 0.967767
Recall: 0.781926
F1 score: 0.864977
ROC AUC: 0.795606

#Cross val score :

Mean accuracy for R_forest classifier 0.7698658121746954

Standard Deviation accuracy for R_forst classifier 0.0008792163255697287

Mean accuracy score for AdaBoostClassifier 0.9766031144699644

Standard Deviation accuracy for AdaBoostClassifier 0.0007913355863261584

Mean accuracy for KNeighborsClassifier 0.9696277169580734

Standard Deviation accuracy for KNeighborsClassifier 0.0006015328210649589

Mean for RandomForestRegressor 0.26356631312811424

Standard Deviation for RandomForestRegressor 0.003755424796178326

Mean accuracy score for SVC 0.7493823203363271

Standard Deviation accuracy score for SVC 0.0010589492947497474

- Key Metrics for success in solving problem under consideration

There are few metrics which have been used to identify and predicting any of the best model are :

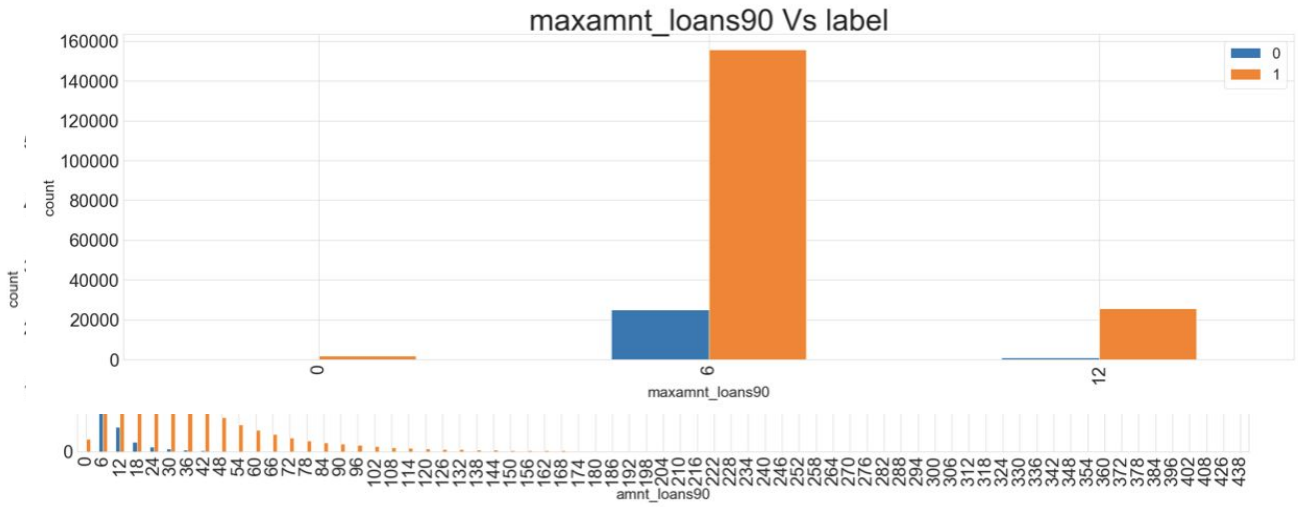
1) Accuracy score
2) Cross_val_score
3) Roc_AUC_score
4) Classification report (Recall)
5) Confusion Matrix
6) True positive rate and False positive rate Statistical graph.
7) Heat map Axes Subplot plot([0,1],[0,1]
8) F1 - Score

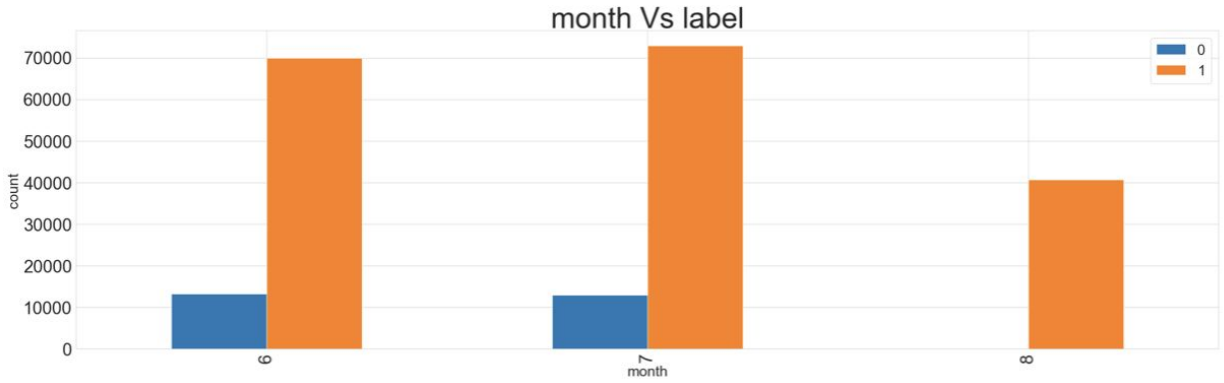
- Visualizations

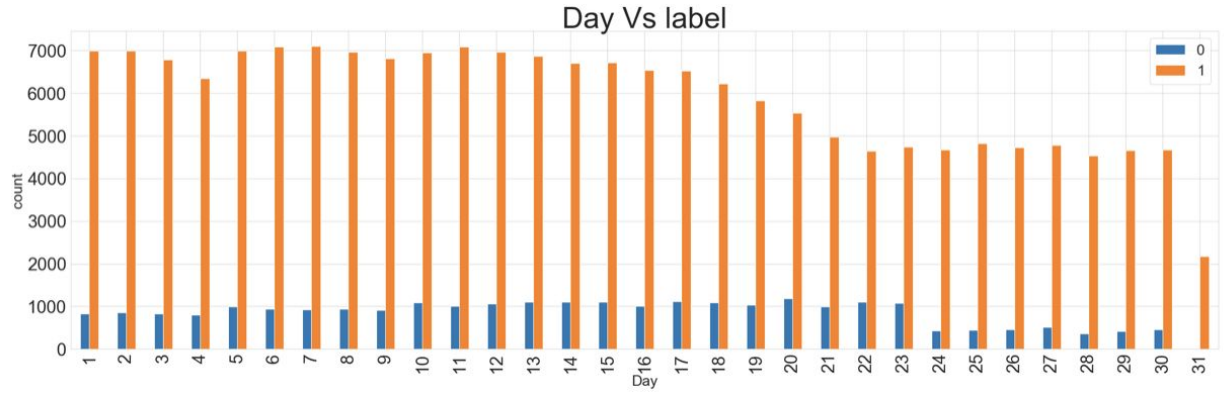
```
def dis_bar(x,y):

    df.groupby([x,y]).size().unstack(level=-1).plot(kind='bar', figsize=(35,10))
    plt.xlabel(x,fontsize= 25)
    plt.ylabel('count',fontsize= 25)
    plt.legend(loc=0,fontsize= 25)
    plt.xticks(fontsize=30)
    plt.yticks(fontsize=30)
    plt.title("{X} Vs {Y}".format(X=x,Y=y),fontsize = 50)
    plt.show()
```

Maximum defaulters which have not paid the loan or the total amount of loans taken by the users in last 90 days is maximum for 6 Rupiah i.e. The amount of 6 Rupiah is the amount where maximum number of users have paid and not paid the loan.It has also come to our notice that the maximum amount of loan in the last 90 days is maximum for 6 Rupiah with respect to the 12 Rupiah.

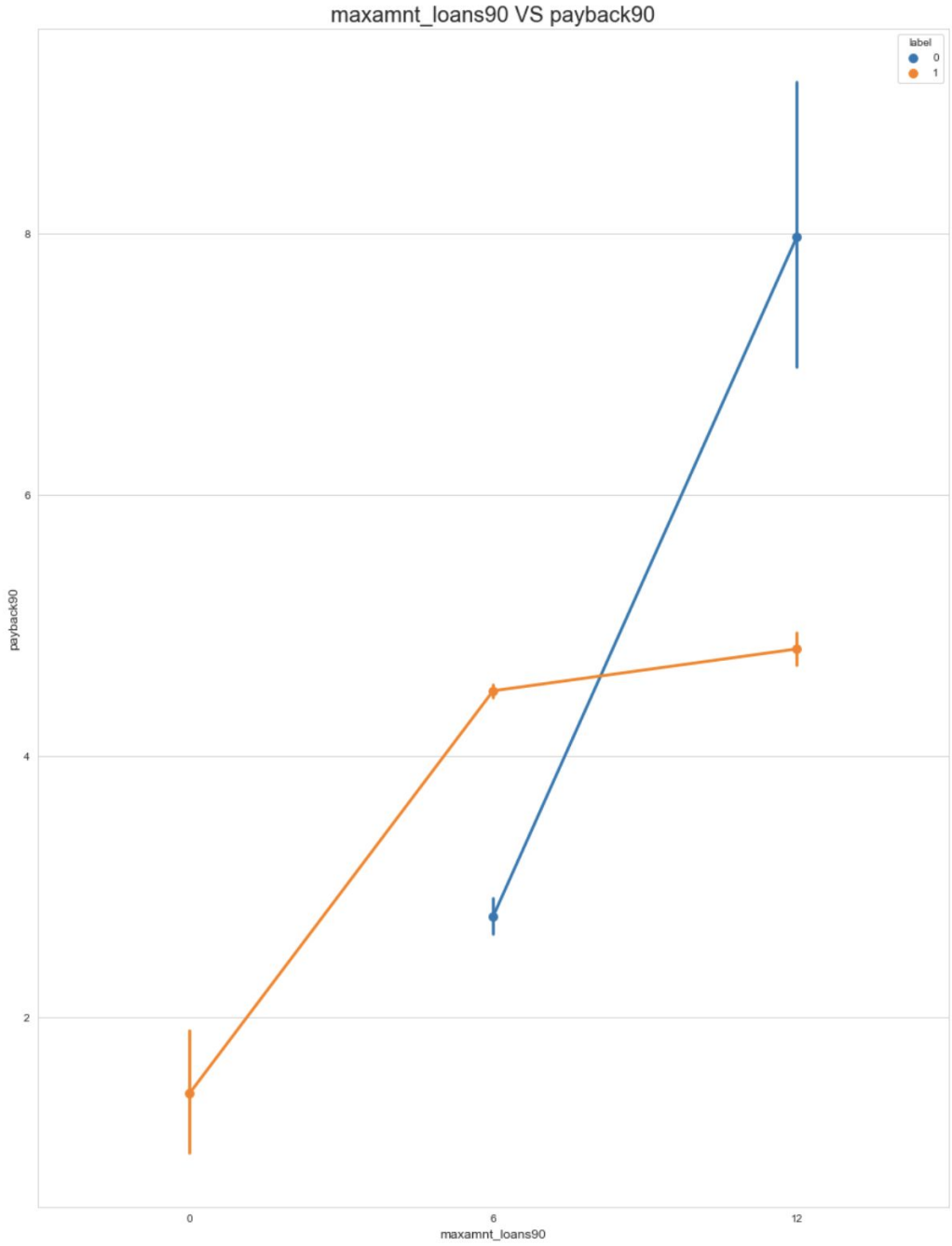




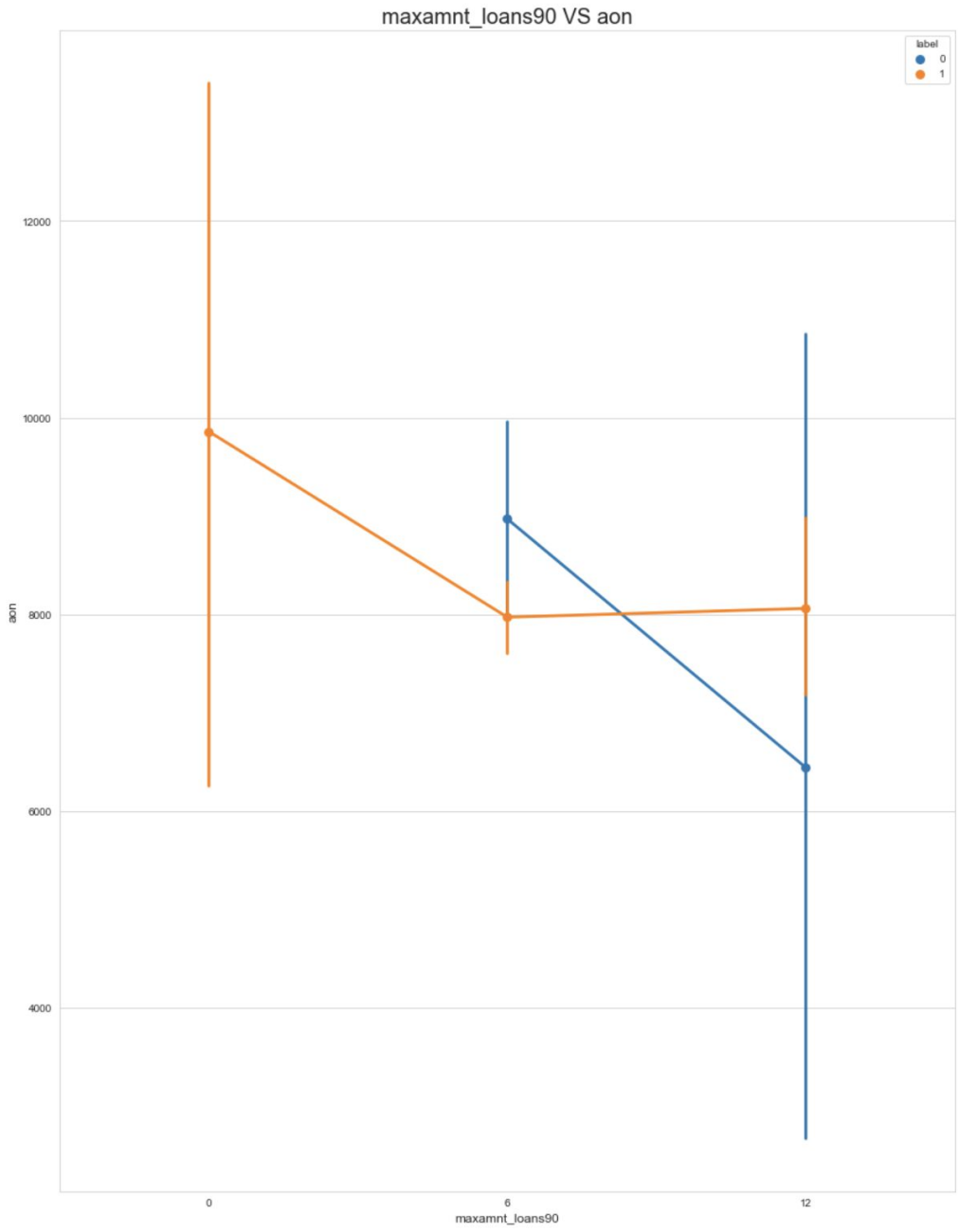


It is seen that with the help of the above bar graph there are no defaulters in the month of 8th and therefore in the month of 6th and 7th we could see that the defaulters of the loan is ranging from 10000 to 12000 in both months. Here after we could see that most of the customers pay their loan back on the 5th , 6th , 7th , and 8th day of the month.

```
plt.figure(figsize=(15,20))
sns.set_style('whitegrid')
sns.pointplot(x='maxamnt_loans90', y='payback90', data=df, hue='label',join=True)
plt.xlabel('maxamnt_loans90',{'fontsize' : 'large'})
plt.ylabel('payback90',{'fontsize':'large'})
plt.title("maxamnt_loans90 VS payback90",{'fontsize':20})
```



Under the non defaulters we could see that as there is an increase in the maximum loan amount of last 90 days is there is an increase in the payback time
Non-defaulters are taking less than 6 days to do the payment where the customers who have done the payment they are paying back in 6 days or above.



If we see the relation between the aon (age on cellular network in days) and maximum amount of loan in 90 days mostly above 6000 or 8000 days whatever the loan amount is it has been paid back that is they are non defaulters on the other hand mostly below 6000 of aon people are defaulting in the payment of the loans. There may be few defaulters ranging in between 8000 to 10000 but they will be negligible, so Statistically with respect to aon we could identify the customers who will be able to repay their loan amount.

CONCLUSION

- Key Findings and Conclusions of the Study

Key findings:

a) If Age on the cellular network is more than 6000 days or 8000 days then the customers are predicted not to be the defaulters and the customer who have joined recently or below 6000 days may be there is a chance of predicting them as a defaulters.
b) Maximum loan amount of 6 Rupiah in last 90 days defaulters ranging from 22000 to 24000 when compared to 12 Rupiah.
c) Amount loan of 6 Rupiah in the last 90 days has been taken by a maximum number of customers.
d) Maximum defaulters which have not paid the loan or the total amount of loans taken by the users in last 90 days is maximum for 6 Rupiah i.e. The amount of 6 Rupiah is the amount where the maximum number of users have paid and not paid the loan.
e) It has also come to our notice that the maximum amount of loan in the last 90 days is maximum for 6 Rupiah with respect to the 12 Rupiah.

Result: :

Model	Accuracy Score	Recall score "0"	Roc Auc Score	Cross Val Score
LogisticRegression	71.87%	84%	77.13%	71.63%
GaussianNB	65.67%	79%	71.45%	65.11%
DecisionTreeClassifier	84.77%	40%	65.45%	84.82%
RandomForestRegressor	89.84%	--	--	26.35%
SVC	**76.39%**	**84%**	**79.86%**	**75%**
KNeighborsClassifier	89%	30%	63.66%	96.96%
AdaBoostClassifier	89.26%	28%	62.62%	97.66%
RandomForestClassifier	**77.49%**	**82%**	**79.44%**	**76.98%**

Choosing a Model:

Most of the above result shows the accuracy score above 75% where the maximum accuracy score is for random forest regressor 89.84% but the cross val score of the model is very low that is 26.35 and test result for this model is : Root mean squared error : 0.27785398052823945 , where r -squared is = 0.2696943972877993. AdaBoost Classifier is performing good with 89.26% and cross val score for recall is also around 97% but it has low recall score for lable "0", if we compare it with Random forest Classifier then this model is not up to the mark. Random forest classifier result has a good output :

Random forest classifier=

The accuracy of RandomForestClassifier : 0.7852129974033265

The f1 score of RandomForestClassifier : 0.8649769483599143

[[5542 1306]

[10936 39212]]

	precision	recall	f1-score	support
0	0.34	0.81	0.48	6848
1	0.97	0.78	0.86	50148
accuracy			0.79	56996
macro avg	0.65	0.80	0.67	56996
weighted avg	0.89	0.79	0.82	56996

Accuracy: 0.785213

Precision: 0.967767

Recall: 0.781926

F1 score: 0.864977
ROC AUC: 0.795606

Other Than This , We have SVC who have performed best till now. The result is as below mentioned. We see that the accuracy score for the model is low as compared to the other models. As discussed that our motive for the project is to formulate in such a way that it shows reality with respect to the data. Hence In order to increase the recall I am sacrificing accuracy and precision.

SVC=

0.7639834374342059 [[5781 1067] [12385 37763]] precision recall f1-score support

0 0.32 0.84 0.46 6848

1 0.97 0.75 0.85 50148

accuracy 0.76 56996

macro avg 0.65 0.80 0.66 56996 weighted avg 0.89 0.76 0.80 56996

Accuracy: 0.763983 Precision: 0.972521 Recall: 0.753031 F1 score: 0.848817 ROC AUC: 0.798610

The F1 Score is the 2*((precision * recall)*/(precision+recall)). It is also called the F Score or the F Measure. Put another way, the F1 score conveys the balance between the precision and the recall which is showing more than 84% Accuracy is used when the True Positives and True negatives are more important while F1-score is used when the False Negatives and False Positives are crucial. ... In most real-life classification problems, imbalanced class distribution exists and thus F1-score is a better metric to evaluate our model on.

Hence I am choosing SVC to be my best model in order to predict the target variable.

- Learning Outcomes of the Study in respect of Data Science

    1) Working with such big data was fun and made me keep more patient with our data.
    2) With this project it taught me to be more careful with the data you have been provided as these data are very expensive and many people have worked day and night to build this data. If there is a loss of data more than 10 % then it would have a direct effect on the prediction and analysis of the data.
    3) It made me understand that accuracy score is not the only metric to identify the best model for our prediction because of that we have used various metrics to perform the prediction of the problem.

4) The Biggest challenge was to face this big data as it was taking too much time to run various algorithms in python but with the help of mentors it was all sorted .
5) Best algorithm for me in this project was :

```
svc=SVC(kernel='rbf',class_weight="balanced")

svc.fit(x_train,y_train)

svc.score(x_train,y_train)

predsvc=svc.predict(x_test)

print(accuracy_score(y_test,predsvc))

print(confusion_matrix(y_test,predsvc))

print(classification_report(y_test,predsvc))

# accuracy: (tp + tn) / (p + n)

accuracy = accuracy_score(y_test, predsvc)

print('Accuracy: %f' % accuracy)

# precision tp / (tp + fp)

precision = precision_score(y_test, predsvc)

print('Precision: %f' % precision)

# recall: tp / (tp + fn)

recall = recall_score(y_test, predsvc)

print('Recall: %f' % recall)

# f1: 2 tp / (2 tp + fp + fn)

f1 = f1_score(y_test, predsvc)
```

```
print('F1 score: %f' % f1)

# ROC AUC

auc = roc_auc_score(y_test, predsvc)

print('ROC AUC: %f' % auc)
```

- Limitations of this work and Scope for Future Work
    1) Too big a data set to work on, my device was not able to do the gridsearchcv technique as it was hanging all the time whenever I used it for any models above.
    2) There are few customers which have no loan history. They have been also considered in testing , if we could remove this data then we would have got a more efficient score. We couldn't remove it as we were not allowed to lose the data more than 8 or 10 %
    3) Distribution of the targeted variable for defaulters is very less that is 12.5% . for further study if we could remove the customers with no loan history and include mode defaulters in order to balance the Label data . This structure will help in training the data more efficiently and hence it will predict the defaulters more prominently.