MICRO-CREDIT DEFAULTER PROJECT

Submitted by:

HARESH KHARSEL

ACKNOWLEDGMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to Flip Robo Technologies Bangalore for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I want to thank my SME Khushboo Garg for providing the Dataset and helping us to solve the problem and addressing out our Query in right time.

I would like to express my gratitude towards my parents & members of Flip Robo for their kind co-operation and encouragement which help me in completion of this project.

I would like to express my special gratitude and thanks to industry persons for giving me such attention and time.

INTRODUCTION

A Microfinance Institution (MFI) is an organization that offers financial services to low-income populations. MFS becomes very useful when targeting especially the unbanked poor families living in remote areas with not much sources of income. The Microfinance services (MFS) provided by MFI are Group Loans, Agricultural Loans, Individual Business Loans and so on.

Many microfinance institutions (MFI), experts and donors are supporting the idea of using mobile financial services (MFS) which they feel are more convenient and efficient, and cost saving, than the traditional high-touch model used since long for the purpose of delivering microfinance services. Though, the MFI industry is primarily focusing on low-income families and are very useful in such areas, the implementation of MFS has been uneven with both significant challenges and successes.

Today, microfinance is widely accepted as a poverty-reduction tool, representing $70 billion in outstanding loans and a global outreach of 200 million clients.

Business Problem Framing

We are working with one such client that is in Telecom Industry. They are a fixed **wireless telecommunications network provider**. They have launched various products and have developed its business and organization based on the budget operator model, offering better products at Lower Prices to all value conscious customers through a strategy of disruptive innovation that focuses on the subscriber.

They are collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed to be defaulter if he deviates from the path of paying back the loaned amount within the time duration of 5 days. For the loan amount of 5 (in Indonesian Rupiah), payback amount should be 6 (in Indonesian Rupiah), while, for the loan amount of 10 (in Indonesian Rupiah), the payback amount should be 12 (in Indonesian Rupiah).

Conceptual Background of the Domain Problem

The conceptual background of this domain problem that client wants to know whether their users are paying the loaned amount within the time frame or not, if not what are the criteria.

So, 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.

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.

Review of Literature

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.

I have observed that there are no null values present in the dataset given by clients and also there are some customers who doesn't have any loan history.

But the data set is quite imbalanced in terms of defaulter and non-defaulters, as the MFI to provide micro-credit on mobile balances to be paid back in 5 days. As per my research, I had seen that most of the users paid the amount within the time frame but if they missed to pay within the time frame of 5 days, they have paid almost like within 7 days, and for that, and I can observe that the client has charged rupiah 6 mostly to the customer.

Most number of loans are taken by Defaulter in the month of June 2016 and there are many valued customers who are good to our client and haven't paid the amount within the deadline.

Motivation for the Problem Undertaken

Our main objective of doing this project is to build a model to predict whether the users are paying the loan within the due date or not. We are going to predict by using Machine Learning algorithms.

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.

ANALYTICAL PROBLEM FRAMING

Mathematical/ Analytical Modelling of the Problem

There are various analytics which I have done before moving forward with exploratory analysis, on the basis of accounts which got recharged in the last 30 days. I set the parameter that if the person is not recharging their main account within 3 months, I simply dropped their data because they are not valuable and they might be old customers, but there is no revenue rotating. Then I had checked the date columns and found that the data belongs to the year 2016. I extracted the month and day from the date, saved the data in separate columns, and tried to visualize the data on the basis of months and days.

I had checked the maximum amount of loan taken by the people and found that the data had more outliers. As per the description given by the client, the loan amount can be paid by the customer is either rupiah 6 or 12 so that I have dropped all the loan amount that shows the loan is taken more than 12 rupiah.

Then I separated the defaulter's data and checked the valuable customers in the network and we found that their monthly revenue is more than 10000 rupiah. Although the data is quite imbalanced and many columns doesn't have that expected maximum value, we dropped that columns. We checked the skewed data and try to treat the skewed data before model processing.

When we try removing the unwanted data, i.e., the outliers, we found that almost 40000+ data has been chopped. Though the data given by the client had almost 35 columns, we have reduced the columns using principal component analysis and then scaled the data. After scaling my data, I have sent the data to various classification models and found that Gradient Boosting Classifier Algorithm is working well.

Data Sources and their formats

The data is been provided by one of our clients from telecom industry. They are a fixed wireless telecommunications network provider and 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.

The data is been given by Indonesian telecom company and they gave it to us in a CSV file, with data description file in excel format. They also had provided the problem statement by explaining what they need from us and also the required criteria to be satisfied.

Let's check the data now. Below I have attached the snapshot below to give an overview.

```
import pandas as pd
df=pd.read_csv('D:/Python file/Project Datasets/Data file.csv',parse_dates=['pdate'])  #Path location of the dataset
df.head()  #Checking out the top 5 rows of the dataset
```

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30	medianam
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	...	6.0	
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	...	12.0	
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	...	6.0	
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	...	6.0	
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	...	6.0	

5 rows × 37 columns

```
#We can see that Unnamed:0 is just the index number.lets drop that column
df.drop('Unnamed: 0',axis=1,inplace=True)
```

```
df.shape    #Checking the dimensions of the dataset
```

```
(209593, 36)
```

Data description

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

Data Pre-processing

Checking out the statistical summary of the dataset

```
df.describe()    #Statistical summary of the dataset
```

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	c
count	208546.000000	208546.000000	208546.000000	208546.000000	208546.000000	208546.000000	208546.000000	208546.000000	208546.000000	2
mean	0.875169	8129.800502	5382.176429	6083.328238	2692.977029	3484.126461	3753.245773	3716.538130	2064.445158	
std	0.330528	75789.889949	9221.838881	10920.483633	4309.893601	5772.799363	53885.607334	53402.052249	2371.554162	
min	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000	-29.000000	0.000000	
25%	1.000000	246.000000	42.470500	42.704167	280.600000	300.260000	1.000000	0.000000	770.000000	
50%	1.000000	527.000000	1471.000000	1500.000000	1084.220000	1334.850000	3.000000	0.000000	1539.000000	
75%	1.000000	982.000000	7243.000000	7800.585000	3356.900000	4202.640000	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 × 35 columns

If we can observe above, we will see that the data is quite imbalanced. In some columns the mean is greater than median, so that we can say the data which given by our client is right skewed from the origin, and for some columns the mean is lesser than the median so I can say the data is left skewed. I also observed that there is a huge difference in 75% quartile and maximum value so from that I conclude that the data contains huge outliers, and also in some columns I have seen that there is no data till 75% quartile and eventually the maximum data come in picture so that we can just drop that columns.

```
df['month']=df['pdate'].dt.month_name()
df['day']=df['pdate'].dt.day
df['year']=df['pdate'].dt.year
```

```
#Checking the value counts for year column
df['year'].value_counts()
```

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

We can see that for all the records, the year is 2016 so that we can drop the column while doing further analysis

```
#Checking the value count for month column
df['month'].value_counts()
```

```
July      85765
June      83154
August    40674
Name: month, dtype: int64
```

All the records are based only on June, July and August months respectively

Then I have checked the date columns and found that the data belongs to the year 2016, so what I have done is that I extracted the month and day from the date and saved the data in separate columns, and tried to visualize the data on the basis of months and days.

h_amt_ma	...	maxamnt_loans30	medianamnt_loans30	cnt_loans90	amnt_loans90	maxamnt_loans90	medianamnt_loans90	payback30	payback90	month	day
1539	...	6.0	0.0	2.0	12	6	0.0	29.000000	29.000000	July	20
5787	...	12.0	0.0	1.0	12	12	0.0	0.000000	0.000000	August	10
1539	...	6.0	0.0	1.0	6	6	0.0	0.000000	0.000000	August	19
947	...	6.0	0.0	2.0	12	6	0.0	0.000000	0.000000	June	6
2309	...	6.0	0.0	7.0	42	6	0.0	2.333333	2.333333	June	22

I had checked the maximum amount of loan taken by the user in last 30 days and found that the data have so much of outliers as per the description given by the client that the loan amount can be paid by the customer is either rupiah 6 or 12 so that I have dropped all the loan amount that shows the loan is taken more than 12 rupiah.

Here maxamnt_loans30 = maximum amount of loan taken by the user in last 30 days. There are only two options: 6 and 12, for which the people need to pay respectively. We can see that the maximum amount of loan amount taken by the people cannot be more than 13Rs so that we can drop them.

```
df[df['maxamnt_loans30'] > 13]    #Checking the records having loan amount > 13
```

	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	...	maxamnt_loan
118	1	23241l82735	1454.0	19.578667	19.578667	148.88	148.88	1.0	0.0	770	...	61907.697
125	1	70901l82730	811.0	166.796667	166.796667	-44.88	-44.88	4.0	0.0	2309	...	22099.413
146	1	73196l70786	198.0	18301.000000	28936.470000	8634.10	11994.34	1.0	0.0	4048	...	98745.934
369	1	50569l82738	1737.0	33.000000	33.000000	130.05	130.05	1.0	0.0	773	...	58925.364
374	1	85457l70781	603.0	9970.000000	10770.000000	9343.38	15443.96	9.0	0.0	770	...	78232.464
...	...	...	...	...	...	...	...	...	...	...	...	
209189	1	55429l85348	1211.0	16039.000000	18645.190000	8391.52	11936.73	2.0	0.0	1539	...	50824.996
209262	1	19105l89232	904.0	96.906667	96.906667	2516.20	2516.20	5.0	0.0	7526	...	17324.994
209331	1	10104l89239	1846.0	55.680000	55.680000	1140.16	1140.16	5.0	0.0	2309	...	92864.501
209392	1	38900l84450	1639.0	52.031333	52.031333	957.10	957.10	3.0	0.0	1539	...	54259.265
209424	1	83277l85348	1246.0	113.715000	113.715000	2229.66	2229.66	1.0	0.0	8000	...	96927.243

1047 rows × 36 columns

```
#As there are more than 1000 records which has the unwanted data, we can drop them
df.drop(df[df['maxamnt_loans30'] > 13].index, inplace = True)
```

Then I separated the defaulter's data and checked the valuable customer to the network and found that their monthly revenue is more than 10000 rupiah. Though the data is quite imbalanced and many columns doesn't have data except the maximum value so I dropped that columns.

Checking the data of defaulters alone

```
defaulters_data=df.loc[df['label'] == 0 ]
defaulters_data
```

	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	...	maxamnt_loans3
0	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	...	6.
11	0	82417I90848	82.0	65.166667	65.166667	326.20	326.20	17.0	0.0	7526	...	6.
15	0	24075I89239	1037.0	12.000000	12.000000	1216.80	1216.80	0.0	0.0	0	...	6.
16	0	82053I85350	1583.0	1000.000000	1000.000000	1000.80	1087.88	0.0	0.0	0	...	6.
21	0	75522I70784	378.0	514.693333	515.200000	56.26	58.20	2.0	0.0	773	...	6.
...	...	...	...	...	...	...	...	...	...	...	...	
209547	0	32172I88688	153.0	5670.733333	5672.200000	1817.08	2764.88	0.0	0.0	0	...	6.
209549	0	59552I90843	843.0	729.235000	758.470000	7470.90	9537.90	1.0	0.0	770	...	6.
209554	0	49076I89233	744.0	1454.491667	1461.750000	559.73	655.28	31.0	0.0	2309	...	6.
209571	0	59768I84453	827.0	1867.668667	1881.180000	1875.72	2312.65	14.0	0.0	1924	...	6.
209584	0	70387I89237	945.0	0.000000	0.000000	78.30	78.30	0.0	0.0	0	...	6.

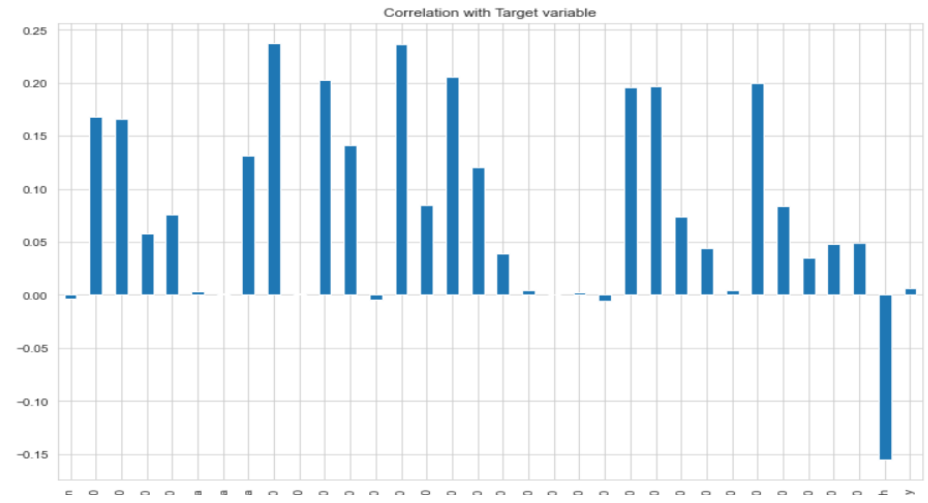
26033 rows × 36 columns

Data Inputs- Logic- Output Relationships

As the data given to us is in csv format and client has described it well that we need to predict whether the defaulters are paying the amount within 5 days or not, it's very clear to us that we have to predict the binary value which is 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. On the basis of correlational input data with my output obtained, we had plotted the figure and it is given below:

```
#lets check how the columns are correlated with the target variable
plt.figure(figsize=(12,8))
df.drop('label',axis=1).corrwith(df['label']).plot(kind='bar',grid=True)
plt.title('Correlation with Target variable')
```

```
Text(0.5, 1.0, 'Correlation with Target variable')
```



As I checked the input and found that almost all the data is quite positively correlated with my output data, except few columns data. Month column is highly negatively correlated with my output though I have set the parameters.

By observing the relation of input output, I am concluding that the user who paid the loan within time frame or not is very much dependent on the user's main account got recharged or not.

Assumptions related to the problem under consideration

I had made an assumption that any telecom company keeps the data of customer within 3 months so I have chopped off my data on basis of that.

I have dropped the 2016 year from pdate columns because the data is from the year 2016, only the date and months are different. We separated months and days to different columns.

Then I separately checked the defaulter's data and found that many valuable users are defaulters as they might have forgotten to pay or they are having a busy life. I separated them so that company can deal politely, because we cannot lose these customers.

Hardware and Software Requirements and Tools Used

For doing this project, the hardware used is a laptop with high end specification and a stable internet connection. While coming to software part, I had used anaconda navigator and in that I have used **Jupyter notebook** to do my python programming and analysis.

For using a csv file, Microsoft excel is needed. In Jupyter notebook, I had used lots of python libraries to carry out this project and I have mentioned below with proper justification:

1. Pandas- a library which is used to read the data, visualisation and analysis of data.
2. NumPy- used for working with array and various mathematical techniques.
3. Seaborn- visualization tool for plotting different types of plot.
4. Matplotlib- It provides an object-oriented API for embedding plots into applications.
5. zscore- technique to remove outliers.
6. skew ()- to treat skewed data using various transformation like sqrt, log, cube, boxcox, etc.
7. PCA- I used this to reduce the data dimensions to 10 columns.
8. standard scaler- I used this to scale my data before sending it to model.
9. train_test_split- to split the test and train data.
10. Then I used different classification algorithms to find out the best model for predictions.
11. joblib- library used to save the model in either pickle or obj file.

MODEL/S DEVELOPMENT AND EVALUATION

Identification of possible problem-solving approaches

As I loaded the dataset, I observed that the data was so imbalanced and there is not much categorical data, which means the visualization is going to be difficult. So, what I had done is that I separated the month and days.

As the data is so imbalanced most of the data is either left skewed or right skewed, I have treated the skewness using sqrt transformation method. Then I passed my data to remove outliers using zscore and found that nearly 40000 data have been chopped off from my data set.

In order to avoid huge data loss, I used the original data itself for further model building as the cleaned data showed that nearly 22% of data had been lost and due to that, the predictions would be affected and the metrics would have not given good scores.

Testing of Identified Approaches (Algorithms)

We are going to use Logistic Regression, GaussianNB, DecisionTreeClassifier and KNeighborsClassifier algorithms for finding out the best model among those. Also, we will use Ensemble Techniques like Random Forest, Adaboost and Gradient Boosting Classifier algorithms to find the best performing model.

Run and Evaluate selected models

The algorithms I have used to study this research are listed above and I have attached the code below for showing how I ran these algorithms:

```
#Importing various classification models for testing
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

```
#Initializing the instance of the model
LR=LogisticRegression()
gnb=GaussianNB()
dtc=DecisionTreeClassifier()
knc=KNeighborsClassifier()
rfc=RandomForestClassifier()
abc=AdaBoostClassifier()
gbc=GradientBoostingClassifier()
```

```
models= []
models.append(('Logistic Regression',LR))
models.append(('GaussianNB',gnb))
models.append(('DecisionTreeClassifier',dtc))
models.append(('KNeighborsClassifier',knc))
models.append(('RandomForestClassifier',rfc))
models.append(('AdaBoostClassifier',abc))
models.append(('GradientBoostingClassifier',gbc))
```

```
#Importing required modules and metrices
from sklearn.metrics import roc_curve,auc,confusion_matrix,classification_report
from sklearn.model_selection import cross_val_score
```

As you can see above, I had called the algorithms, then I called the empty list with the name models [ ], and calling all the model one by one and storing the result in that.

We can observe that I imported the metrics to find the accuracy score, roc_auc_curve, auc, confusion_matrix and classification_report in order to interpret the model's output. Then I also selected the model to find the cross_validation_score value.

Let’s check the code below:

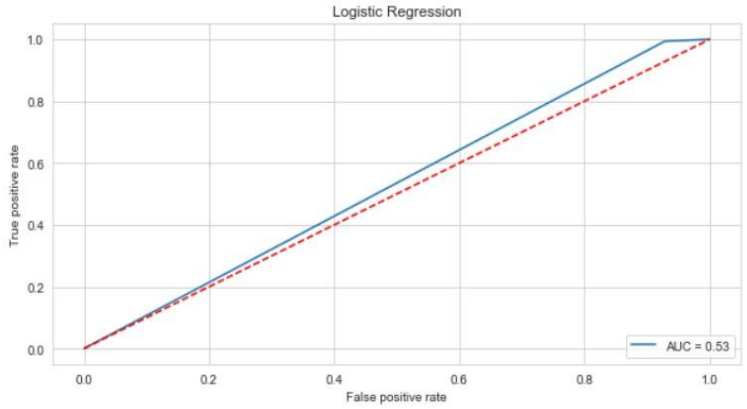
```
#Making a for loop and calling the algorithm one by one and save data to respective model using append function
Model=[]
score=[]
cvs=[]
rocscore=[]
for name,model in models:
    print('******************************',name,'**************************')
    print('\n')
    Model.append(name)
    model.fit(x_train,y_train)
    print(model)
    pre=model.predict(x_test)
    print('\n')
    AS=accuracy_score(y_test,pre)
    print('accuracy_score: ',AS)
    score.append(AS*100)
    print('\n')
    sc=cross_val_score(model,x,y,cv=5,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,pre)
    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:\n ')
    print(classification_report(y_test,pre))
    print('\n')
    print('Confusion matrix: \n')
    cm=confusion_matrix(y_test,pre)
    print(cm)
    print('\n')
    plt.figure(figsize=(10,50))
    plt.subplot(912)
    print('AUC_ROC curve:\n')
    plt.title(name)
    plt.plot(false_positive_rate,true_positive_rate, label='AUC = %0.2f'% roc_auc)
    plt.plot([0,1],[0,1],'r--')
    plt.legend(loc='lower right')
    plt.xlabel('False positive rate')
    plt.ylabel('True positive rate')
    plt.show()

    print('\n\n\n')
```

As you can observe above, I made a for loop and called all the algorithms one by one and appending their result to models. The same I had done to store roc_auc_curve, auc score, and cross validation score. Let me show the output so that we can glance the result in more appropriate way.

The following are the outputs of the different algorithms I had used, along with the metrics score obtained:

```
****************************** Logistic Regression *************************

LogisticRegression()

accuracy_score:  0.8785662910573004

cross_val_score:  0.8776912514473404

roc_auc_score:  0.5327353727765934

Classification report:

              precision    recall  f1-score   support

           0       0.62      0.07      0.13      5207
           1       0.88      0.99      0.93     36503

    accuracy                           0.88     41710
   macro avg       0.75      0.53      0.53     41710
weighted avg       0.85      0.88      0.83     41710


Confusion matrix:

[[  374  4833]
 [  232 36271]]
```

AUC_ROC curve:




```
****************************** GaussianNB *************************

GaussianNB()

accuracy_score:  0.8594581635099496

cross_val_score:  0.8588896441033699
```

roc_auc_score: 0.6974221191655938

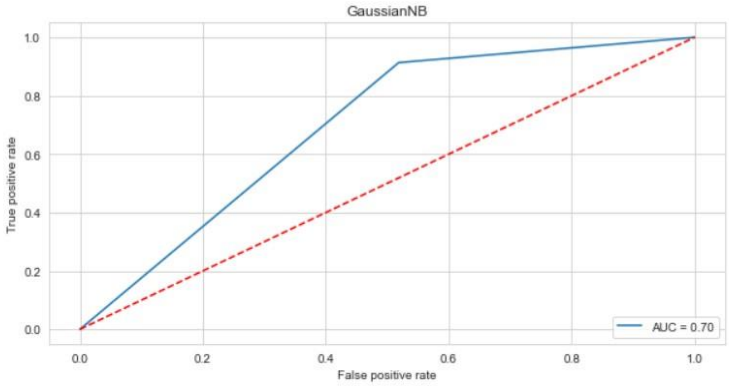
Classification report:

	precision	recall	f1-score	support
0	0.44	0.48	0.46	5207
1	0.93	0.91	0.92	36503
accuracy			0.86	41710
macro avg	0.68	0.70	0.69	41710
weighted avg	0.86	0.86	0.86	41710

Confusion matrix:

```
[[ 2507  2700]
 [ 3162 33341]]
```

AUC_ROC curve:



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

DecisionTreeClassifier()

accuracy_score: 0.8512347158954687

cross_val_score: 0.8542671643644102

roc_auc_score: 0.680621794722829

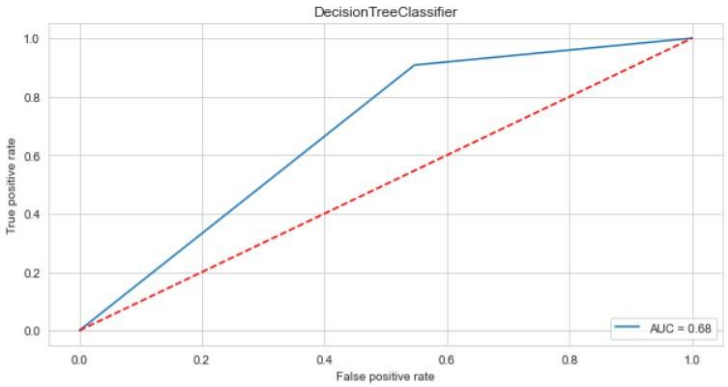
Classification report:

	precision	recall	f1-score	support
0	0.41	0.45	0.43	5207
1	0.92	0.91	0.91	36503
accuracy			0.85	41710
macro avg	0.67	0.68	0.67	41710
weighted avg	0.86	0.85	0.85	41710

Confusion matrix:

```
[[ 2360  2847]
 [ 3358 33145]]
```

AUC_ROC Curve:



********** KNeighborsClassifier **********

KNeighborsClassifier()

accuracy_score: 0.8771997122992088

cross_val_score: 0.8784057035836295

roc_auc_score: 0.6697701435664181

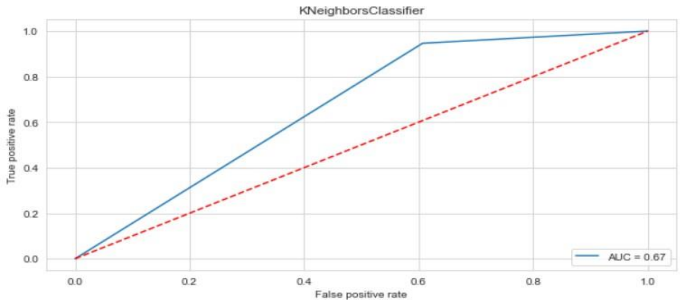
Classification report:

	precision	recall	f1-score	support
0	0.51	0.39	0.44	5207
1	0.92	0.95	0.93	36503
accuracy			0.88	41710
macro avg	0.71	0.67	0.69	41710
weighted avg	0.87	0.88	0.87	41710

Confusion matrix:

[[2048 3159]
[1963 34540]]

AUC_ROC curve:



********** RandomForestClassifier **********

RandomForestClassifier()

accuracy_score: 0.8861424118916327

cross_val_score: 0.8868163424934133

roc_auc_score: 0.6734797523501742

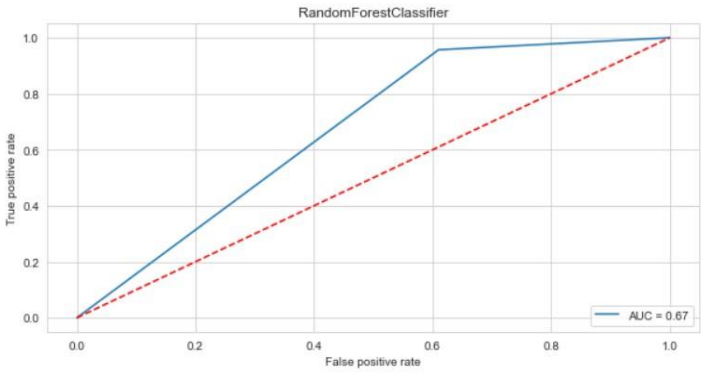
Classification report:

	precision	recall	f1-score	support
0	0.56	0.39	0.46	5207
1	0.92	0.96	0.94	36503
accuracy			0.89	41710
macro avg	0.74	0.67	0.70	41710
weighted avg	0.87	0.89	0.88	41710

Confusion matrix:

```
[[ 2031  3176]
 [ 1573 34930]]
```

AUC_ROC curve:



********** AdaBoostClassifier **********

AdaBoostClassifier()

accuracy_score: 0.8832414289139295

cross_val_score: 0.8817718960758117

roc_auc_score: 0.6172395016284458

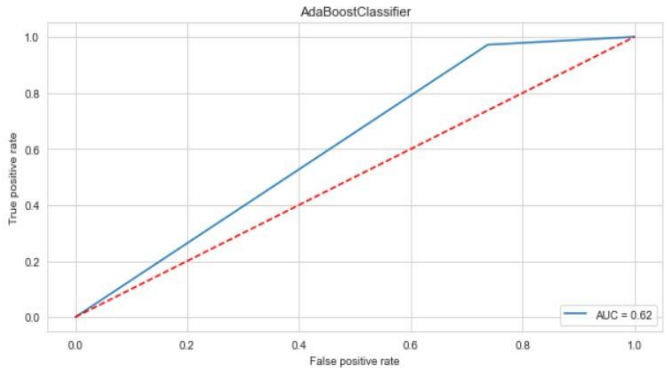
Classification report:

	precision	recall	f1-score	support
0	0.57	0.26	0.36	5207
1	0.90	0.97	0.94	36503
accuracy			0.88	41710
macro avg	0.74	0.62	0.65	41710
weighted avg	0.86	0.88	0.86	41710

Confusion matrix:

```
[[ 1368  3839]
 [ 1031 35472]]
```

AUC_ROC curve:



********** GradientBoostingClassifier **********

GradientBoostingClassifier()

accuracy_score: 0.8907935746823303

cross val score: 0.8902544291432415

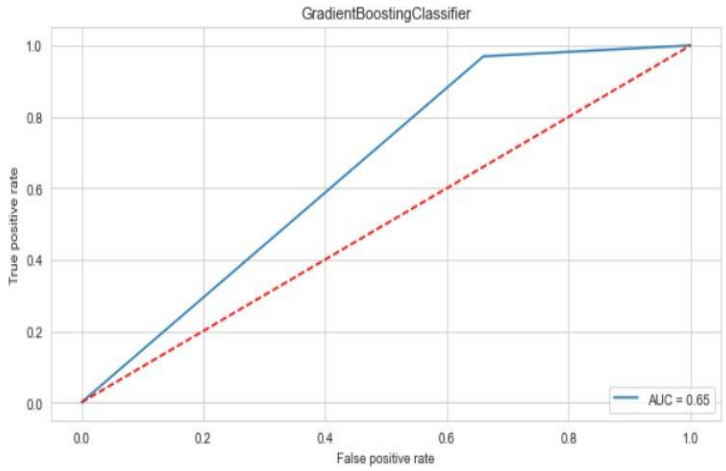
Classification report:

	precision	recall	f1-score	support
0	0.61	0.34	0.44	5207
1	0.91	0.97	0.94	36503
accuracy			0.89	41710
macro avg	0.76	0.65	0.69	41710
weighted avg	0.87	0.89	0.88	41710

Confusion matrix:

[[1771 3436]
[1119 35384]]

AUC_ROC curve:



We will save the outputs obtained in a new data frame and the code is given below:

```
#Finalizing the result
result=pd.DataFrame({'Model':Model, 'Accuracy_score': score,'Cross_val_score':cvs,'roc_auc_score':rocscore})
result
```

	Model	Accuracy_score	Cross_val_score	roc_auc_score
0	Logistic Regression	87.856629	87.769125	53.273537
1	GaussianNB	85.945816	85.888964	69.742212
2	DecisionTreeClassifier	85.123472	85.426716	68.062179
3	KNeighborsClassifier	87.719971	87.840570	66.977014
4	RandomForestClassifier	88.614241	88.681634	67.347975
5	AdaBoostClassifier	88.324143	88.177190	61.723950
6	GradientBoostingClassifier	89.079357	89.025443	65.473203

We can see that Gradient Boosting Classifier algorithm is performing well compared to other algorithms, as it is giving an accuracy score of 89.07 and cross validation score of 89.02.

Key Metrics for success in solving problem under consideration

The key metrics used here were accuracy_score, cross_val_score, classification report, auc_score and confusion matrix. We tried to find out the best parameters and also to increase our scores by using Hyperparameter Tuning and we will be using GridSearchCV method.

1. Cross Validation:

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 holdout 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 holdout 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.

2. Confusion Matrix:

A **confusion matrix**, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a **matching matrix**). Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class (or vice versa). The name stems from the fact that it makes it easy to see whether the system is confusing two classes (i.e., commonly mislabelling one as another).

It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table).

3. Classification Report:

The classification report visualizer displays the precision, recall, F1, and support scores for the model. There are four ways to check if the predictions are right or wrong:

1. **TN / True Negative**: the case was negative and predicted negative
2. **TP / True Positive**: the case was positive and predicted positive
3. **FN / False Negative**: the case was positive but predicted negative
4. **FP / False Positive**: the case was negative but predicted positive

Precision: Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class, it is defined as the ratio of true positives to the sum of a true positive and false positive. It is the accuracy of positive predictions. The formula of precision is given below:

$$\text{Precision} = TP/ (TP + FP)$$

Recall: Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives. It is also the fraction of positives that were correctly identified. The formula of recall is given below:

$$\text{Recall} = TP/(TP+FN)$$

F1 score: The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy. The formula is:

$$F1\ Score = 2*(Recall * Precision) / (Recall + Precision)$$

Support: Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support doesn't change between models but instead diagnoses the evaluation process.

4. AUC-ROC Curve and score:

AUC (**Area Under the Curve**) - ROC (**Receiver Operating Characteristics**) curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represent the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0s as 0s and 1s as 1s. By analogy, the Higher the AUC, the better the model is at distinguishing between patients with the disease and no disease.

The ROC curve is plotted with TPR against the FPR where TPR is on the y-axis and FPR is on the x-axis.

Score is the area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

5. Hyperparameter Tuning:

There is a list of different machine learning models. They all are different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as **Hyperparameters.** These hyperparameters will define the architecture of the model, and the best part about these is that you get a choice to select these for your model. You must select from a specific list of hyperparameters for a given model as it varies from model to model.

We are not aware of optimal values for hyperparameters which would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically. This selection procedure for hyperparameter is known as **Hyperparameter Tuning. We can do tuning by using GridSearchCV.**

GridSearchCV is a function that comes in Scikit-learn (or SK-learn) model selection package. An important point here to note is that we need to have Scikit-learn library installed on the computer. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

Hyperparameter Tuning

Gradient Boosting Classifier

```
#Creating parameter list to pass in GridSearchCV
parameters={'n_estimators':[50,100,500],'learning_rate':[0.0001,0.001,0.01,0.1]}
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
gbc=GradientBoostingClassifier(random_state=116)   #Using the best random state we obtained
gbc=GridSearchCV(gbc,parameters,cv=5,scoring='accuracy')
gbc.fit(x_train,y_train)
print(gbc.best_params_)     #Printing the best parameters obtained
print(gbc.best_score_)      #Mean cross-validated score of best_estimator
```

```
{'learning_rate': 0.1, 'n_estimators': 500}
0.8909947611387684
```

```
#Using the best parameters obtained
gbc=GradientBoostingClassifier(random_state=116,n_estimators=500,learning_rate=0.1)
gbc.fit(x_train,y_train)
pred=gbc.predict(x_test)
print("Accuracy score: ",accuracy_score(y_test,pred)*100)
print('Cross validation score: ',cross_val_score(gbc,x,y,cv=5,scoring='accuracy').mean()*100)
print('Classification report: \n')
print(classification_report(y_test,pred))
print('Confusion matrix: \n')
print(confusion_matrix(y_test,pred))
```

We got the best parameters and after using those values to get the final metrics scores after tuning, the output will be as follows:

```
Accuracy score:  89.16806521217934
Cross validation score:  89.13237376622523
Classification report:

              precision    recall  f1-score   support

           0       0.61      0.37      0.46      5207
           1       0.91      0.97      0.94     36503

    accuracy                           0.89     41710
   macro avg       0.76      0.67      0.70     41710
weighted avg       0.88      0.89      0.88     41710

Confusion matrix:

[[ 1902  3305]
 [ 1213 35290]]
```

After applying the Tuning, we can see that our scores had been increased with the help of best parameters obtained, i.e., accuracy score from 89.07 to 89.16 and cross validation score from 89.02 to 89.13, and they are good scores too. Now, we will finalize Gradient Boosting Classifier algorithm model as the final model.

Final the model

```
gbc_prediction=gbc.predict(x)
print('Predictions of GradientBoosting Classifier: ',gbc_prediction)   #Printing the predicted values
```

```
Predictions of GradientBoosting Classifier:  [1 1 1 ... 1 1 1]
```

```
#Saving the final model
import joblib
joblib.dump(gbc,'Micro_CreditProject_Classification.obj')
```

```
['Micro_CreditProject_Classification.obj']
```

Saving the predicted values in a csv file

```
pred_results=pd.DataFrame(gbc_prediction)
pred_results.to_csv('Micro_CreditProject_Predictions.csv')
```

We will final the best model obtained by saving the predictions in a separate csv file and also, we will save the model in either a pickle or obj file using joblib library.

Visualizations

Now, we will see the different plots done with this dataset in order to know the insight of the data present. Below are the codes given for the plots and the output obtained:

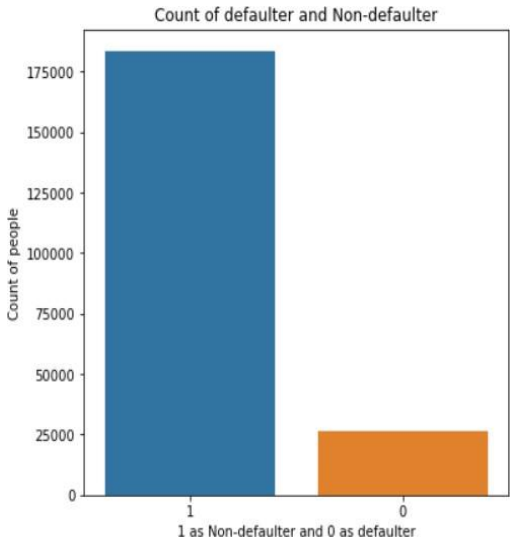
Exploratory Data Analysis

```
#Importing Matplotlib and Seaborn
import seaborn as sns
import matplotlib.pyplot as plt
```

Univariate Analysis

```
#Counting the number of defaulter and non-defaulter
print('\n','Label '1' indicates that the loan has been payed i.e. Non- defaulter','\n','while', '\n', 'Label '0' indicates that t
print(df['label'].value_counts())
plt.subplots(figsize=(6,6))
sns.countplot(x='label',data=df,order= df['label'].value_counts().index)
plt.title('Count of defaulter and Non-defaulter')
plt.xlabel('1 as Non-defaulter and 0 as defaulter')
plt.ylabel('Count of people')
plt.show()
```

```
 Label '1' indicates that the loan has been payed i.e. Non- defaulter
 while
 Label '0' indicates that the loan has not been payed i.e. defaulter

1    183431
0     26162
Name: label, dtype: int64
```



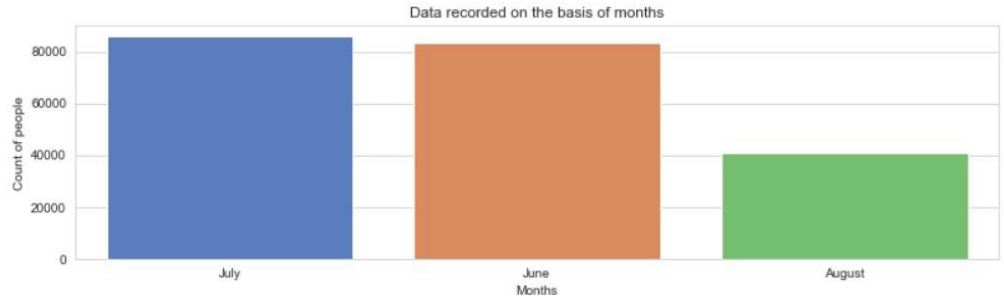
➔ We can see that 183431 people had paid their loan amount whereas 26162 people did not pay the amount.

➔ Here the dataset is imbalanced. Label '1' has approximately 87.5% records, while label '0' has approximately 12.5% records.

Data recorded on the basis of months

```
#Data recorded on the basis of months
print(df['month'].value_counts())
plt.figure(figsize = (13,11))
sns.set_style('whitegrid')
plt.subplot(311)
sns.countplot(x='month',data=df,palette='muted',order= df['month'].value_counts().index)
plt.title('Data recorded on the basis of months')
plt.xlabel('Months')
plt.ylabel('Count of people')
plt.show()
```

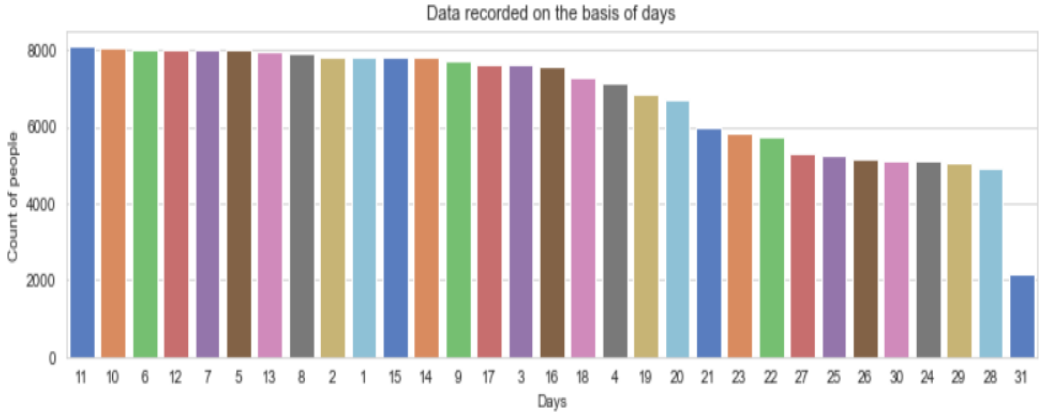
```
July      85765
June      83154
August    40674
Name: month, dtype: int64
```



➔ Maximum number of people records have been recorded in the month of July with a value of 85765 whereas least number of records have been recorded in the month of August with a value of 40674.

➔ All the records are recorded in the months of June, July and August respectively.

Data recorded on the basis of days

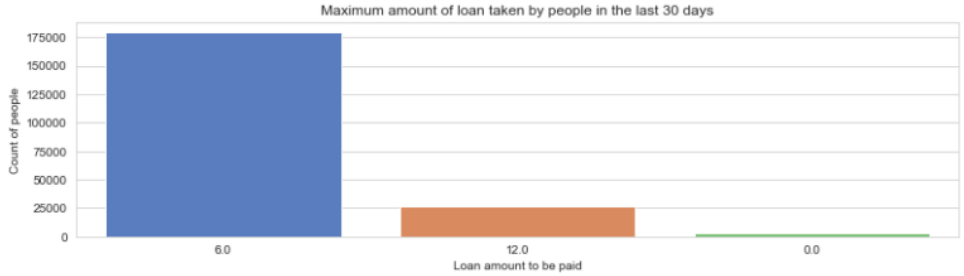


➔ On 11th day, maximum number of people either took the loans or repaid the loan amount. The number of people is 8092.

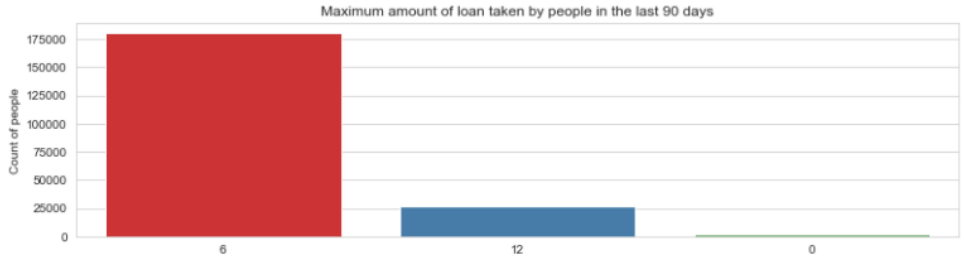
➔ On 31st day, minimum number of people either took the loans or repaid the loan amount. The number of people is 2178.

Maximum amount of loan taken by people in last 30 and 90 days

```
6.0     179193
12.0     26109
0.0       3244
Name: maxamnt_loans30, dtype: int64
```




```
6     180038
12     26477
0       2031
Name: maxamnt_loans90, dtype: int64
```

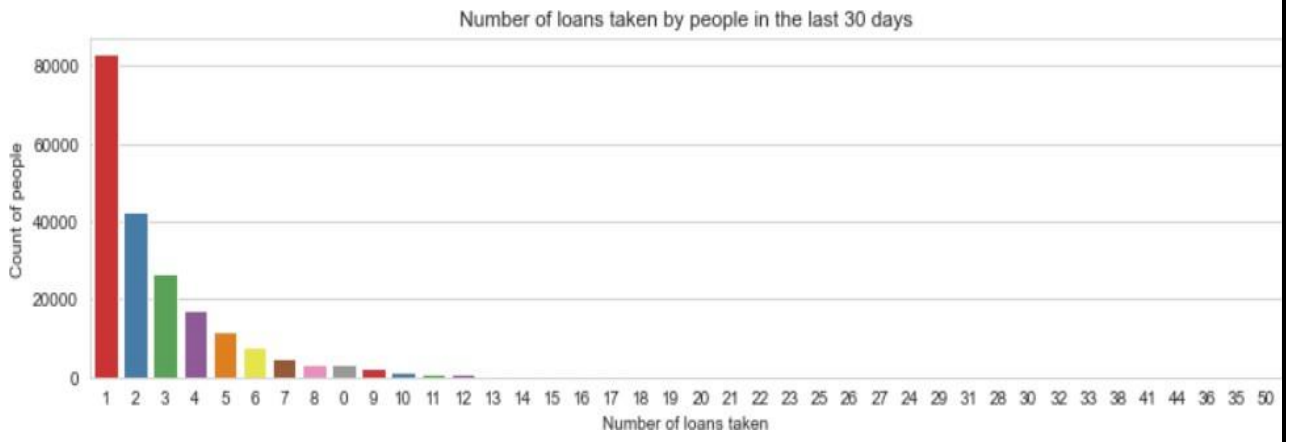


➔ In 30 days, maximum number of people had taken 6Rs as the loan amount and the number of people is 179193 whereas the minimum number of people had not taken loan and their number is 3244.

➔ In 90 days, maximum number of people had taken 6Rs as the loan amount and the number of people is 180038 whereas the minimum number of people had not taken loan and their number is 2031.

➔ Maximum number of people had taken 12Rs as the loan amount within 90 days and their number is 26477 whereas for 30 days the number of people who had taken 12Rs is 26109 respectively.

Number of loans taken by people in last 30 days



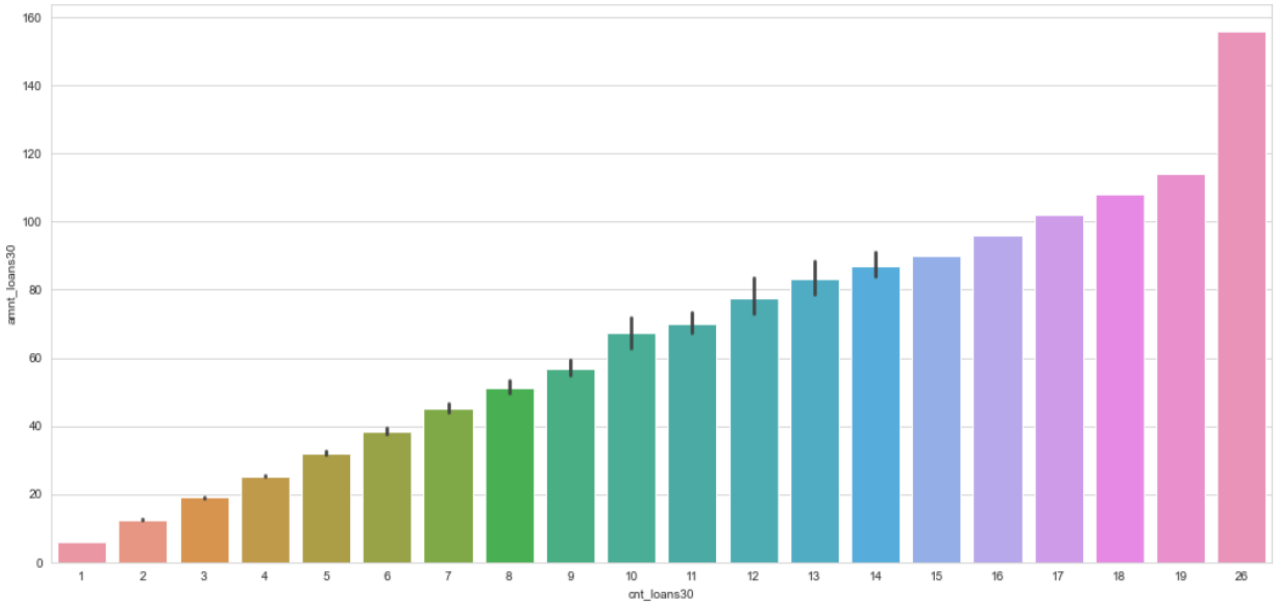
➔ 83028 is the maximum number of people who had taken loans and they had taken only once.

➔ Single person had taken loans for 41, 50, 36 times etc.

Bivariate Analysis:

Checking the data of defaulters alone

- **Number of loans taken by people in last 90 days vs Amount of loan taken by the people in last 90 days**



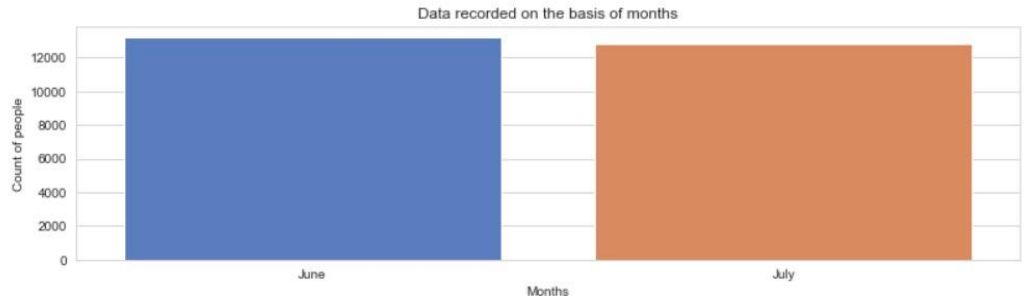
➔ Maximum number of times the loan taken by the people is 26 and the amount is equivalent to 15.

➔ Minimum number of times the loan taken by the people is 1 and the amount is equivalent to 6.

- **Data recorded on the basis of months**

```
#Data recorded on the basis of months
print(defaulters_data['month'].value_counts())
plt.figure(figsize = (13,11))
plt.subplot(311)
sns.countplot(x='month',data=defaulters_data,palette='muted',order=defaulters_data['month'].value_counts().index)
plt.title('Data recorded on the basis of months')
plt.xlabel('Months')
plt.ylabel('Count of people')
plt.show()
```

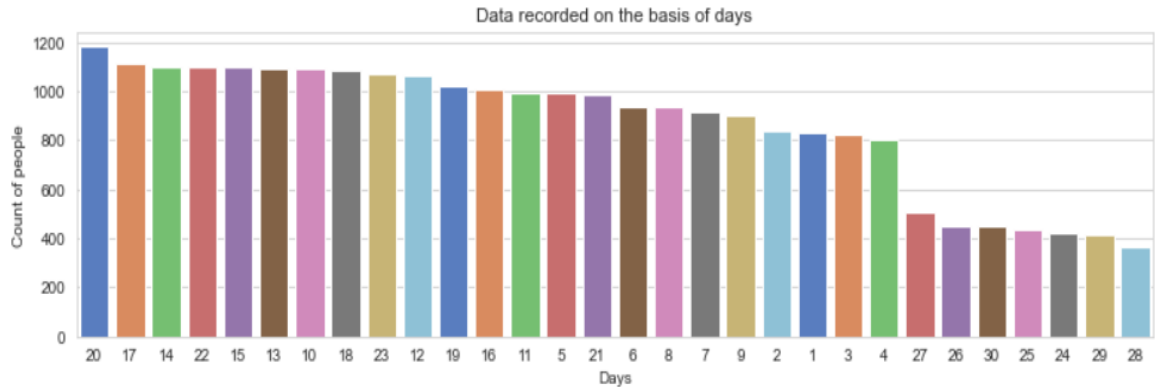
```
June    13187
July    12846
Name: month, dtype: int64
```



➔ The records had been available in the months of June and July, whereas there are no records in August.
➔ Maximum number of records are available in June with a value of 13187.

- **Data recorded on the basis of days**

```
#Data recorded on the basis of days
print(defaulters_data['day'].value_counts())
plt.figure(figsize = (13,11))
plt.subplot(312)
sns.countplot(x='day',data=defaulters_data,palette='muted',order=defaulters_data['day'].value_counts().index)
plt.title('Data recorded on the basis of days')
plt.xlabel('Days')
plt.ylabel('Count of people')
plt.show()
```

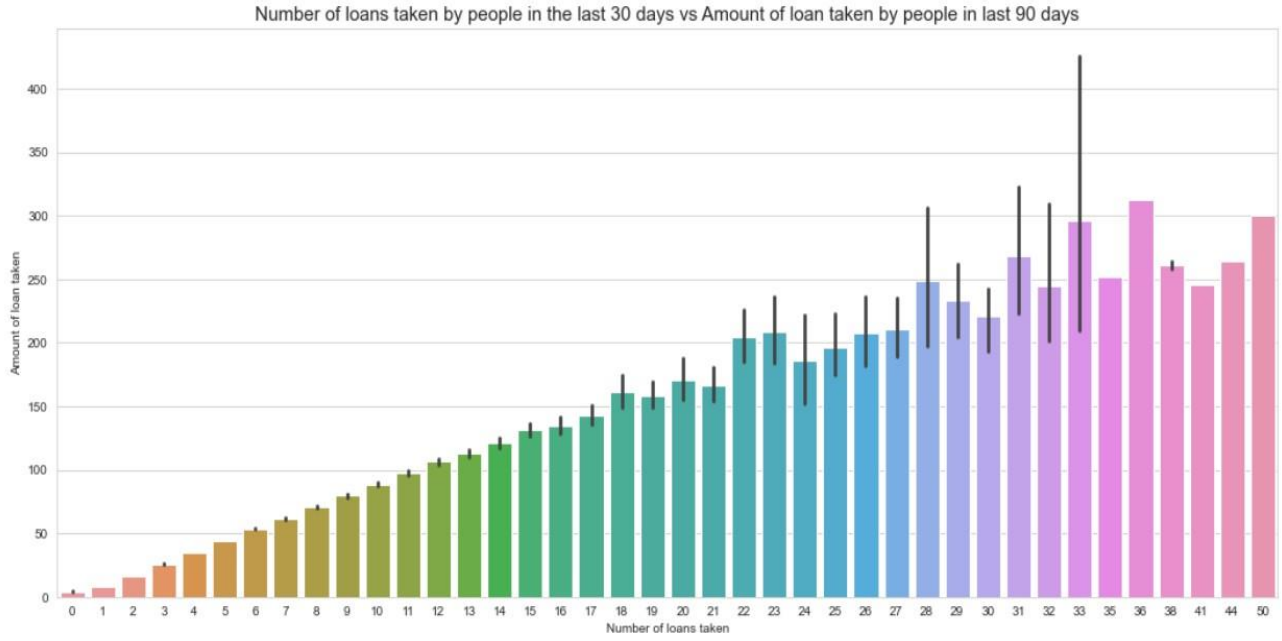


➔ On 20th day the maximum number of records are there and the value is 1182.

➔ On 28th day, the minimum number of records are there and the value is 363.

Number of loans taken by people in last 30 days vs Amount of loan taken by the people in last 90 days

```
#Checking the number of loans taken by people in last 30 days vs Amount of loan taken by the people in last 90 days.
plt.figure(figsize=(18,8))
sns.barplot(x="cnt_loans30",y='amnt_loans90', data=df)
plt.title('Number of loans taken by people in the last 30 days vs Amount of loan taken by people in last 90 days', fontsize=15)
plt.xlabel('Number of loans taken')
plt.ylabel('Amount of loan taken')
plt.show()
```



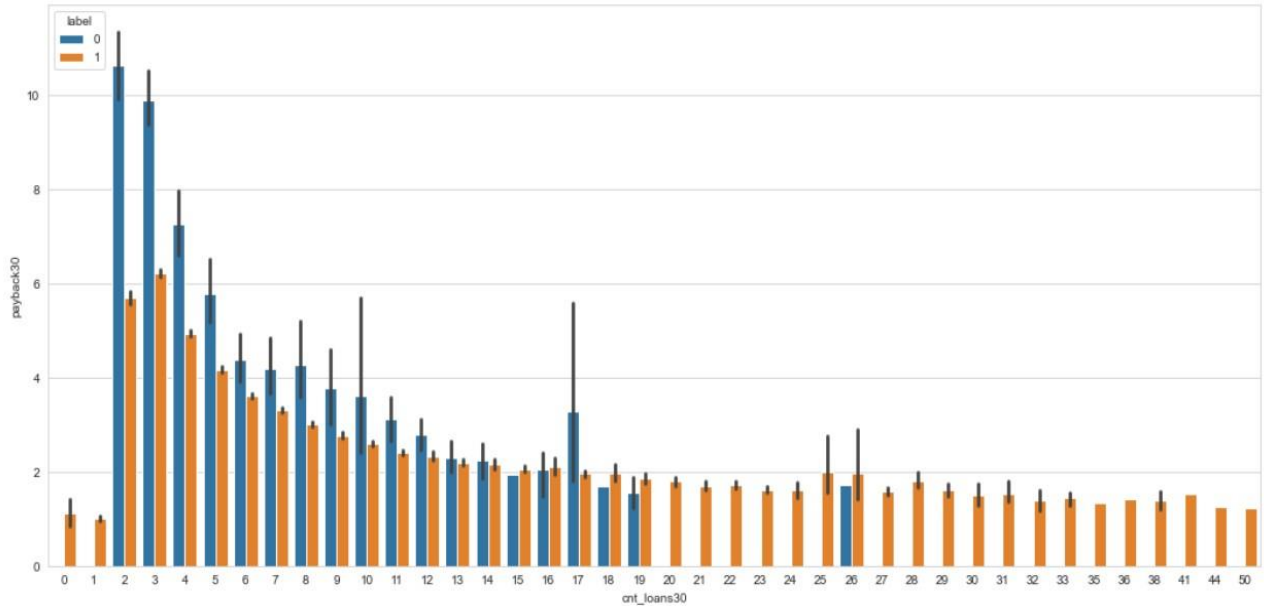
➔ Maximum number of loans taken by the people is 36 and the amount is equivalent to 320.

➔ Minimum number of loans taken by the people is 0.

Maximum number of loans taken vs Amount paid within due dates by people or not on the basis of label

```
#Checking the maximum number of loans taken vs Amount payed within due dates by people or not on the basis of label
plt.figure(figsize=(18,8))
sns.barplot(x="cnt_loans30",y='payback30',hue='label', data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2f95ee4c0>
```



We can observe that the Average payback time over last 30 days is higher for people who had taken 2 times the loan and say that the users with a smaller number of loans taking are more than the defaulters.

Interpretation of the Results

- ## Visualization

➔ The data set is quite imbalanced and there is lot of work to do before going to explore in depth with this dataset.

➔ As per the client, they want us to predict whether the loan amount has been paying by users in 5 days or not.

➔ As this dataset belongs from the year 2016, the datas are recorded in the month of June, July and August.

➔ From the visualization, I am able to say that the most loan amount taken is rupiah 6 and most of the users are paying the loan within the time frame of 5 days, but many early users failed to do so. They usually take almost 7 to 8 days to pay the loan amount and even the valuable customers some time fails to pay the amount within the time frame.

➔ I would suggest the client to increase the payment duration to 7 days, although they want from us to predict the amount within 5 days.

➔ One more thing I noticed that, the smaller number of loans taken by the people are more defaulters and the frequently loan taking customers are less defaulters.

➔ Most importantly, the people are paying the amount early or lately and sometimes they might fail to pay within the time frame, but I observed that almost 80% of users are paying the amount within 7-8 days.

➔ There are no null values in the dataset.

➔ Label 1 has approximately 87% records, and label 0 have 13 % records which is purely imbalanced.

➔ maxamnt_loans90 columns gives information about customers with no loan history.

➔ msisdn features some values are duplicated which might not be realistic. So, we have dropped the row which does not contain realistic value.

➔ The collected data is only for one Telecom circle area as per Dataset Documentation so that we had dropped that column.

- **Correlation Analysis**

➔ Here, we check how the different independent features are corelated with each other and their strength of Relationship, weather they are positively correlated or negatively corelated.

➔ The value of Correlation ranges from -1 to +1. -1 indicate the negative correlation with increase in one independent variable the dependent variable decreases +1 indicates the positive correlation means with increase in independent variable dependent variable also increase.

➔ In our case, some of the independent features are highly corelated with each other with a person corelation strength more than 0.9 hence we have to dropped all the features which are highly multicollinear to avoid multicollinearity.

➔ month column is highly negative correlated with the target variable with a highest value of -54%.

➔ 98% is the highest positive correlated value.

• Checking for skewness

➔ Skewed data are not normally distributed; either they are positive skewed or negative skewed. If the data is skewed, it impacts on the accuracy of the model. So, it's very important to remove the skewness for right and left skewed data by using transform methods like square and cube root, boxcox and logarithm transformation.

➔ For visualization, we use distplot to check the distribution of data points and the shape of the curve. Any value greater than 0.55 or less than -0.55 is considered to be skewed data.

➔ In our case, most of the data are skewed and hence we have to remove the skewness during Scaling because if we remove the skewness by log or boxcox method it will induce nan values. Sometimes while using root transforms, it can cause to form nan values. It is better to remove those values before scaling because it will show ValueError while running the code.

- **Checking for outliers**

➔ Outliers are the data points that differ significantly from other observations. Any data points greater than +3 and -3 standard deviations are called as outliers.

➔ Z-score is the automated method used for handling outliers and it's important to remove outliers as it impacts on the Accuracy of the Model.

➔ In our case, each and every information is important to us and we can't afford to lose 7-8% of data. But during the outlier's analysis, we found that we are losing nearly 22% of data and hence we have decided not to use the outliers removed data, but using the original data itself.

➔ Nearly 40000+ rows of data had been lost after the removal of outliers. We can visualize the presence of outliers by using boxplots.

- **Splitting of data**

➔ The data is divided into two parts using component splitting.

➔ In this experiment, data is split based on a ratio of 80:20 for the training set and the testing set.

➔ The training set data is used in the logistic regression component for model training, while the testing set data is used in the prediction component.

➔ We reduce the dimensions of the dataset to 10 features by using a technique known as Principal Component Analysis, as there are nearly 35 features present.

- **Standardization**

➔ Standardization of data is done by using Standard Scaler.

➔ Standard scaler is used to bring the data points to standard Normal Distribution having mean = 0 and SD +- 1 to enhance accuracy of the model. In our case, its most important due to presence of high skewed data and Outliers.

CONCLUSION

Key Findings and Conclusions of the Study

➔ After getting an insight of this dataset, we were able to understand how the people took loans and how they repaid on the basis of various factors.

➔ First, we loaded the dataset and did the EDA process and other pre-processing techniques like skewness check and removal, handling the outliers present, filling the missing data, visualizing the distribution of data, etc.

➔ There were some customers with no loan history and it is because the data is imbalanced such that, label '1' has approximately 87.5% records, while, label '0' has approximately 12.5% records.

➔ There were many outliers present in the dataset and even though they were present, we used the original data itself as the data loss should not be more than 7-8% of data. When we checked the percentage loss after handling outliers, it was nearly 22% so that we used the original data itself with outliers in it.

➔ Then we did the model training, building the model and finding out the best model on the basis of different metrices scores we got like ROC-AUC curve, Classification Report, Confusion matrix, etc.

➔ We used algorithms like Logistic Regression, GaussianNB, DecisionTreeClassifier and KNeighborsClassifier algorithms for finding out the best model among those. We also used Ensemble Techniques like

RandomForest, Adaboost and Gradient Boosting algorithms to find the best performing model.

➔ After performing the analysis, we got Gradient Boosting Classifier algorithm as the best algorithms among all as it gave an accuracy score of 89.07 and cross_val_score of 89.02, which was highest among all. Then for finding out the best parameter and improving the scores, we performed Hyperparameter Tuning.

➔ The problem while doing Hyperparameter Tuning is that it took nearly 2 hours to fetch the best parameters as there were nearly 2lakh records to process through and for ensemble techniques normally, tuning takes more time. For avoiding this problem, we can use other notebooks like Google Colaboratory for processing faster.

➔ After Tuning, we saw that our scores had been increased with the help of best parameters obtained, i.e., accuracy score from 89.07 to 89.16 and cross validation score from 89.02 to 89.13, and they are good scores too.

➔ We finalized the best model we obtained by saving the model in an obj file. Also, we found the predictions obtained and saved it in a new data frame.

➔ The problems in this dataset were: High skewness, more outliers, time consumption due to a greater number of records.

➔ Overall, we can say that it is a good dataset to predict the micro credit predictions and we can also use the finalized model for deployment process too.

➔ We can improve the data by adding more features that are positively correlated with the target variable, having less outliers, normally distributed values, etc.

Learning Outcomes of the Study in respect of Data Science

While exploring this dataset, I came across various challenges to tackle first. I saw that the dataset is so imbalanced that irrespective of any null data, the dataset was so complicated. While visualizing this dataset, I can say how company can boom with the help of Data science and as I further studied this dataset, I can say that company should increase their days of payment because most of the users paid their loan within 7-8 days. The company should be focused more on less amount of loan. As I started the model building, I found that Gradient Boosting Classifier algorithm was working quite well with an accuracy score of almost 89% and also with a cross validation score of 89%. Even after Tuning the model, the scores were increased too. The only problem was that it took me nearly 2 hours to find the scores as the dataset was large. After building the model, I predicted the same input and got quite good response from the model. Though the dataset was quite imbalanced and challenging, but I learnt so many things while exploring it.

Limitations of this work and Scope for Future Work

- ➔ The problems in this dataset were: High skewness, more outliers, time consumption due to a greater number of records.
- ➔ Some of the duplicate records were present in the dataset and they are successfully handled during Pre-processing.
- ➔ Overall, we can say that it is a good dataset to predict the micro credit predictions and we can also use the finalized model for deployment process too.
- ➔ We can improve the data by adding more features that are positively correlated with the target variable, having less outliers, normally distributed values, etc.
- ➔ There is no word of enough in Data Science as there are more people with different mind sets, who can explore this dataset in different angles. The limitation is that we have to use the data carefully because this dataset is having lots of outliers and data is much more skewed, and as per the client, we can't lose the data more than 7-8%.