FLIP ROBO

Project Name:

Micro Credit Default Project

Submitted By:

Swetha Joshi

ACKNOWLEDGMENT:

This is a special project from our telecommunication client, I thank fliprobo for providing an opportunity to prove myself and also for providing necessary data and assistance, I thank data trained institution for providing necessary assistance and along with google.

INTRODUCTION:

- Business Problem Framing

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

  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.

- Conceptual Background of the Domain Problem

In my view the conceptual background that requires to deal with this

type of problem is that one should know how to group the customer by grouping function, we have to group the costumers who have paid and not paid and should analyse the features of those people who didn’t pay back within the guaranteed time, and we need statistical knowledge, visualization knowledge, and algorithm knowledge.

- Review of Literature

  Read so many similar projects over kaggle but those were on banking sector and got some idea how to segregate the features and got a workflow on the project and came to know that each column should be analysed as per the requirement, and understood that every dataset is different according to condition we should think and apply suitable technique.

- Motivation for the Problem Undertaken

  For every work I do the only motivation is that i should take care of my mother so well and with respect to this project its quite challenging because I am first time dealing with such huge data and I get lot more things to learn,

Analytical Problem Framing

- Data Sources and their formats

  Almost all datatypes were in float 64 format and target variable was in int, through heatmap found certain important variables that has a way with target variable.

- Data Pre-processing Done

  I followed following steps to clean the data

  ```
  1.df.isnull().sum()
  2. k=df.applymap(np.isreal)
  k.head()
  3. for col in k:
     print(col)
     print(k[col].value_counts())
     print('\n')
  ```

  I cross checked whether I have true value in all the column ensuring that I don’t have any further null value I proceeded further with the project.

4.While cleaning the data we came across certain data which are far from reality like days be in negative number and recharging 23.5 times an hour so I assumed that i cant be possible so I took out those data.

- Data Inputs- Logic- Output Relationships

  Before inputting the data, we have certain steps like carrying out EDA and finding out which are the columns which has got nothing to do anything with any other column and removing the same, cleaning the data, removing the skewness, if there is any class imbalance is there means adopting suitable technique to neutralize that, after all these we will get a precise dataset which we will give as input,
  Logic we used is that preparing the precise dataset using various techniques, the output is purely depends on how was our input, its just a depiction of how we cleaned the data.

- State the set of assumptions (if any) related to the problem under consideration

  I assumed that days cant be negative and in some cases the number Of recharges per minute we are getting around 23.5 times, just removed those data by assuming that its not possible,

- Hardware and Software Requirements and Tools Used

  Software used is anaconda navigator and jupiter notebook is the environment, used i3 processor, it was taking several hours for computing,

  Libraries used are pandas, seaborn, matplotlib, sklearn, plotly, numpy etc.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  Method I adopted to solve this particular problem is grouping the target variables by category and analysing each category separately and adopting all classification algorithms and testing over each one to get the desirable answer.

- Testing of Identified Approaches (Algorithms)

  Algorithm used are:

  1.Logistic Regression

  2.GaussianNB

  3.Support Vector Classifier

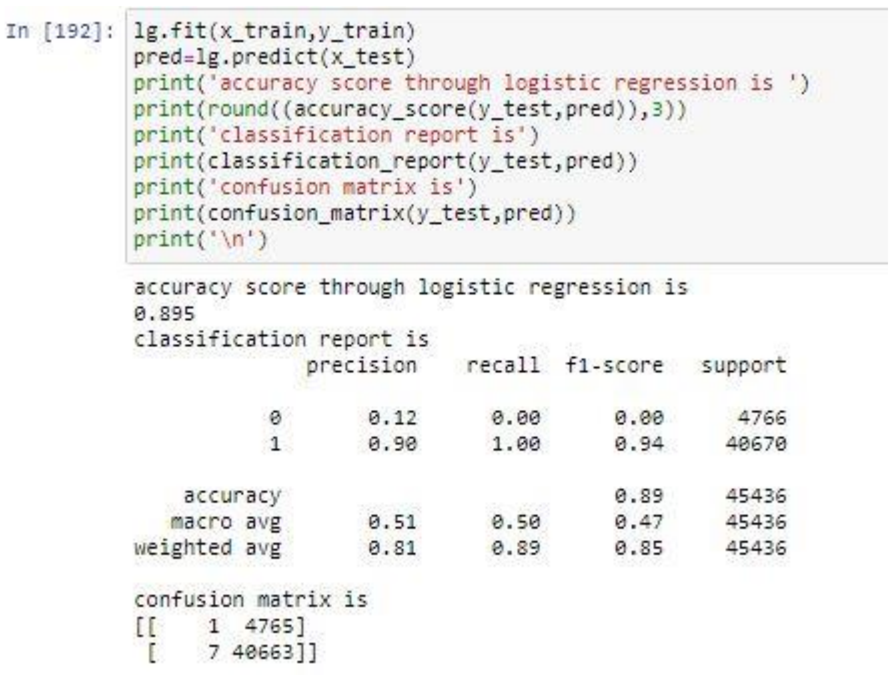
  4.Decision Tree Classifier

  5.KneighborsClassifier

  6.Adaboost

  7.Random Forest

- Run and Evaluate selected models



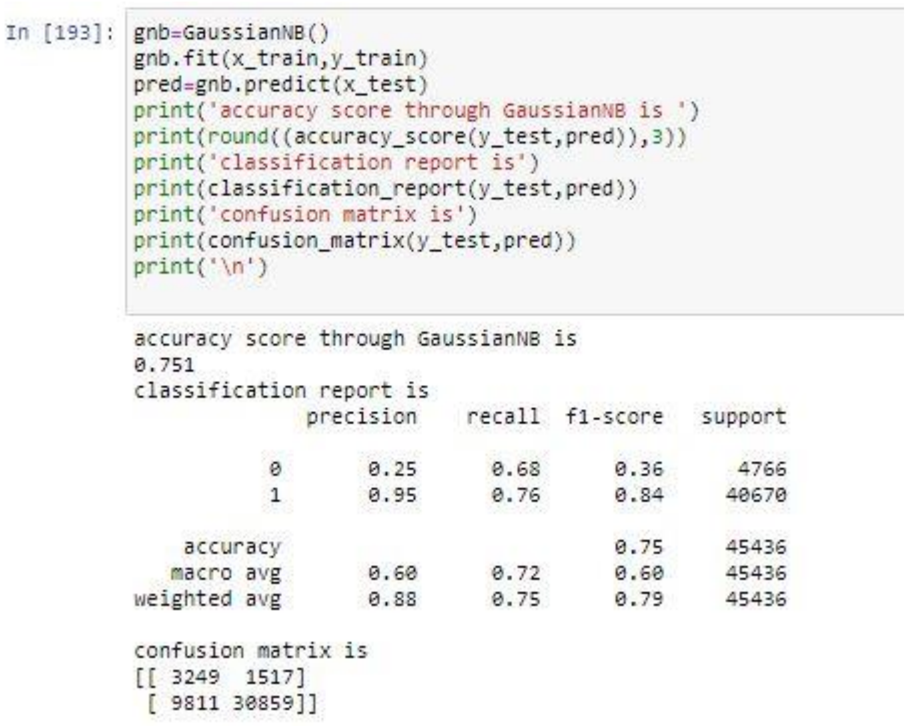
```
In [192]: lg.fit(x_train,y_train)
          pred=lg.predict(x_test)
          print('accuracy score through logistic regression is ')
          print(round((accuracy_score(y_test,pred)),3))
          print('classification report is')
          print(classification_report(y_test,pred))
          print('confusion matrix is')
          print(confusion_matrix(y_test,pred))
          print('\n')
```

```
accuracy score through logistic regression is
0.895
classification report is
              precision    recall  f1-score   support

           0       0.12      0.00      0.00      4766
           1       0.90      1.00      0.94     40670

    accuracy                           0.89     45436
   macro avg       0.51      0.50      0.47     45436
weighted avg       0.81      0.89      0.85     45436

confusion matrix is
[[    1  4765]
 [    7 40663]]
```



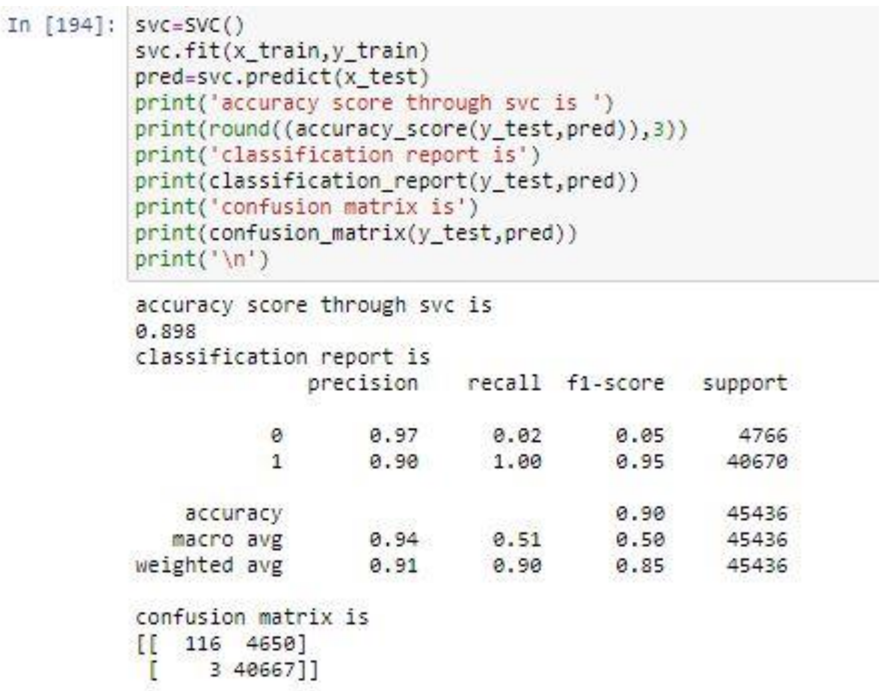
```
In [193]: gnb=GaussianNB()
          gnb.fit(x_train,y_train)
          pred=gnb.predict(x_test)
          print('accuracy score through GaussianNB is ')
          print(round((accuracy_score(y_test,pred)),3))
          print('classification report is')
          print(classification_report(y_test,pred))
          print('confusion matrix is')
          print(confusion_matrix(y_test,pred))
          print('\n')
```

```
accuracy score through GaussianNB is
0.751
classification report is
              precision    recall  f1-score   support

           0       0.25      0.68      0.36      4766
           1       0.95      0.76      0.84     40670

    accuracy                           0.75     45436
   macro avg       0.60      0.72      0.60     45436
weighted avg       0.88      0.75      0.79     45436

confusion matrix is
[[ 3249  1517]
 [ 9811 30859]]
```



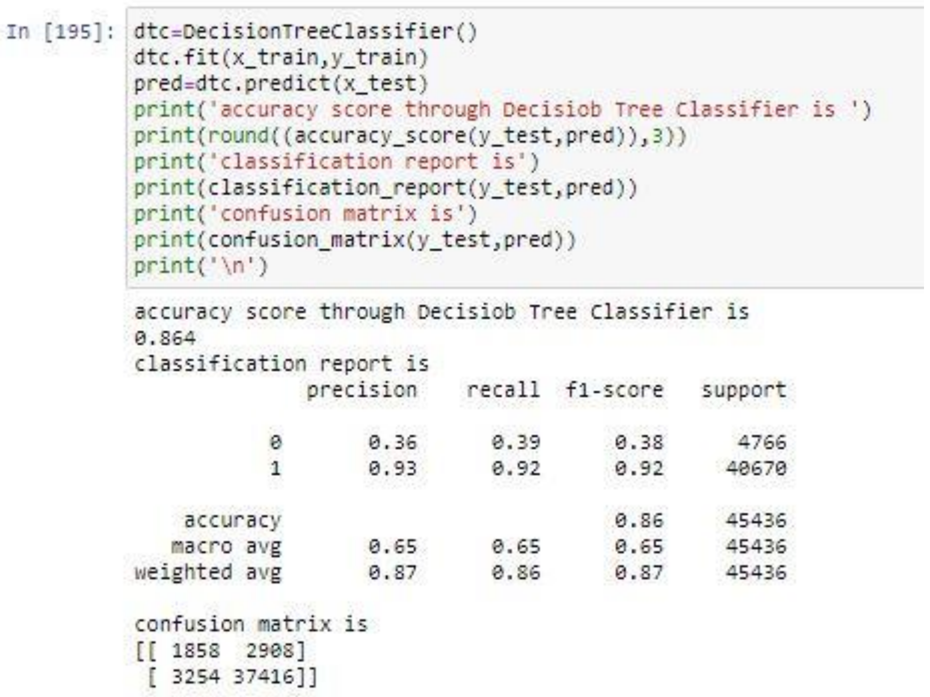
```
In [194]: svc=SVC()
          svc.fit(x_train,y_train)
          pred=svc.predict(x_test)
          print('accuracy score through svc is ')
          print(round((accuracy_score(y_test,pred)),3))
          print('classification report is')
          print(classification_report(y_test,pred))
          print('confusion matrix is')
          print(confusion_matrix(y_test,pred))
          print('\n')
```

```
accuracy score through svc is
0.898
classification report is
              precision    recall  f1-score   support

           0       0.97      0.02      0.05      4766
           1       0.90      1.00      0.95     40670

    accuracy                           0.90     45436
   macro avg       0.94      0.51      0.50     45436
weighted avg       0.91      0.90      0.85     45436

confusion matrix is
[[  116  4650]
 [    3 40667]]
```



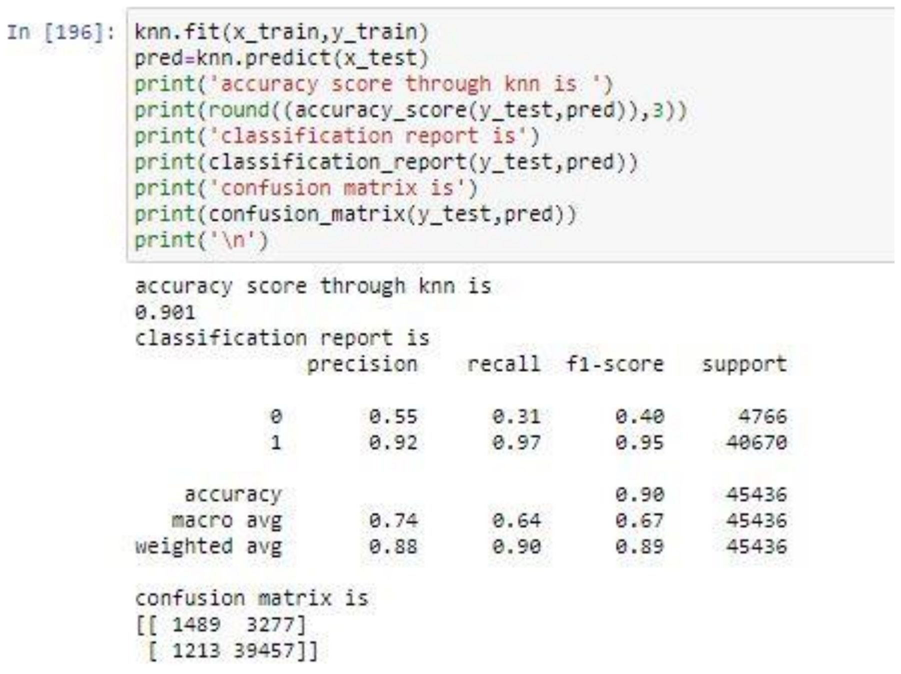
```
In [195]: dtc=DecisionTreeClassifier()
          dtc.fit(x_train,y_train)
          pred=dtc.predict(x_test)
          print('accuracy score through Decisiob Tree Classifier is ')
          print(round((accuracy_score(y_test,pred)),3))
          print('classification report is')
          print(classification_report(y_test,pred))
          print('confusion matrix is')
          print(confusion_matrix(y_test,pred))
          print('\n')
```

```
accuracy score through Decisiob Tree Classifier is
0.864
classification report is
              precision    recall  f1-score   support

           0       0.36      0.39      0.38      4766
           1       0.93      0.92      0.92     40670

    accuracy                           0.86     45436
   macro avg       0.65      0.65      0.65     45436
weighted avg       0.87      0.86      0.87     45436

confusion matrix is
[[ 1858  2908]
 [ 3254 37416]]
```



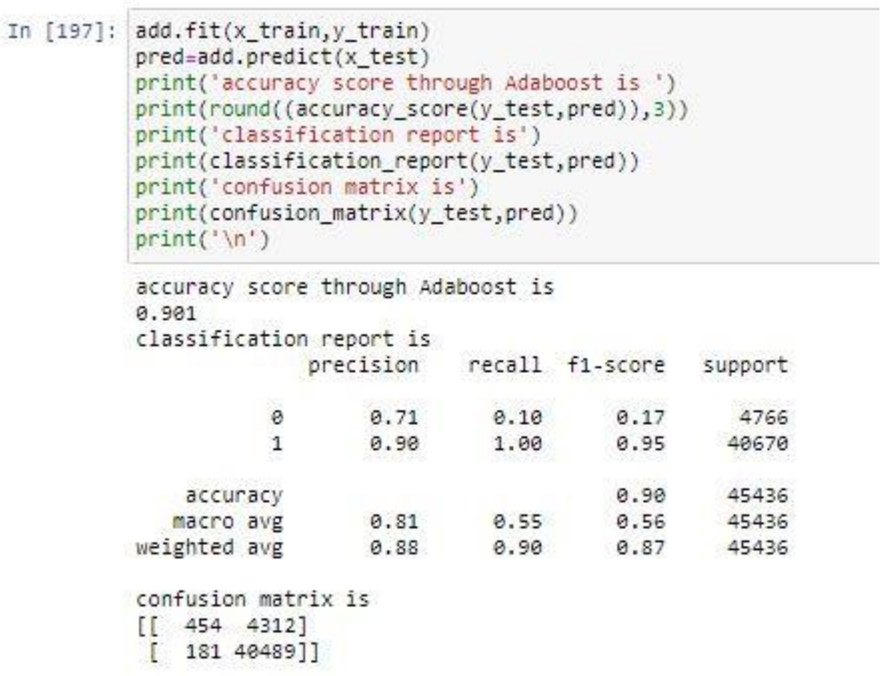
```
In [196]: knn.fit(x_train,y_train)
          pred=knn.predict(x_test)
          print('accuracy score through knn is ')
          print(round((accuracy_score(y_test,pred)),3))
          print('classification report is')
          print(classification_report(y_test,pred))
          print('confusion matrix is')
          print(confusion_matrix(y_test,pred))
          print('\n')
```

```
accuracy score through knn is
0.901
classification report is
              precision    recall  f1-score   support

           0       0.55      0.31      0.40      4766
           1       0.92      0.97      0.95     40670

    accuracy                           0.90     45436
   macro avg       0.74      0.64      0.67     45436
weighted avg       0.88      0.90      0.89     45436

confusion matrix is
[[ 1489  3277]
 [ 1213 39457]]
```



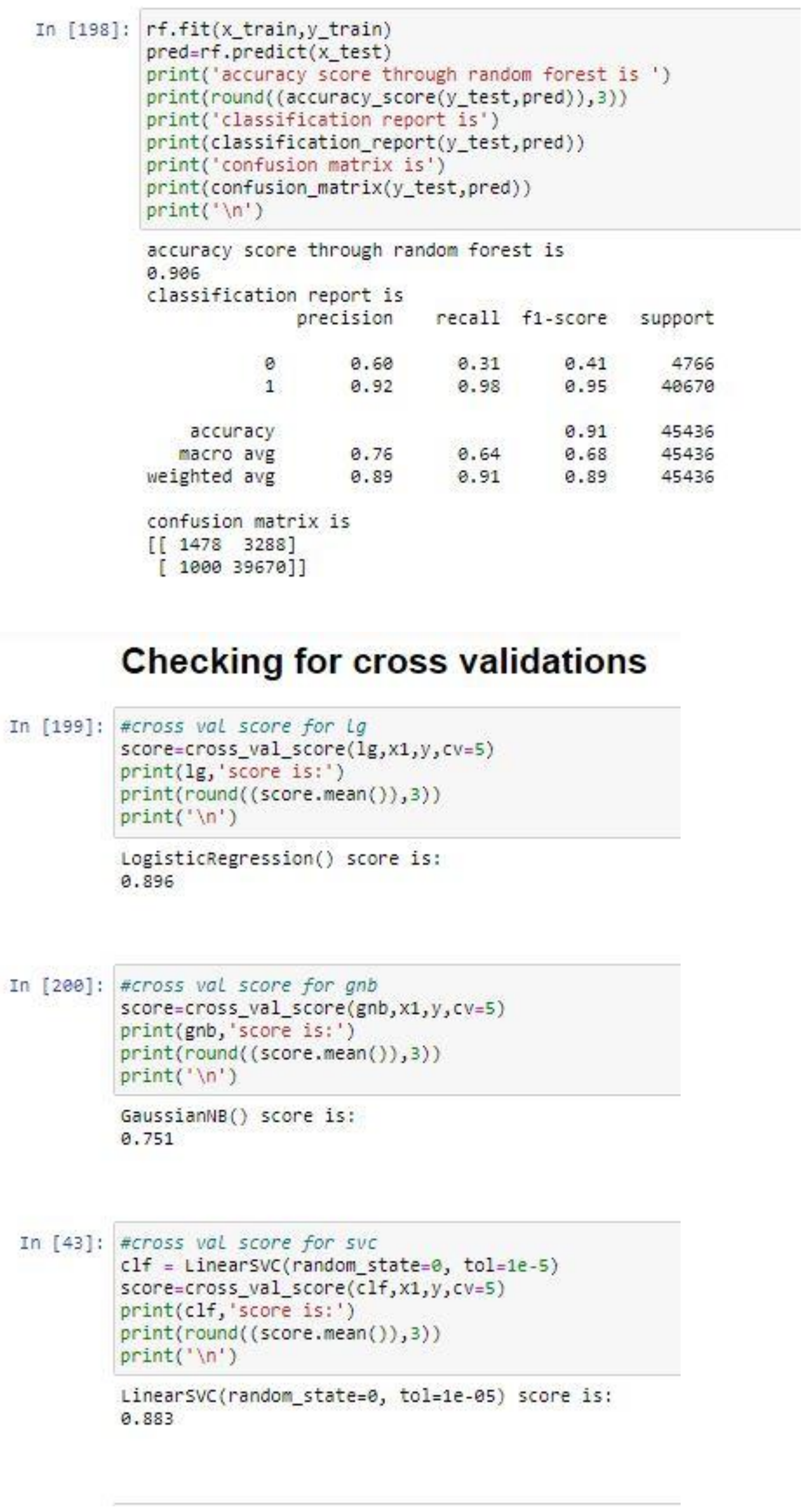
```
In [197]: add.fit(x_train,y_train)
          pred=add.predict(x_test)
          print('accuracy score through Adaboost is ')
          print(round((accuracy_score(y_test,pred)),3))
          print('classification report is')
          print(classification_report(y_test,pred))
          print('confusion matrix is')
          print(confusion_matrix(y_test,pred))
          print('\n')
```

```
accuracy score through Adaboost is
0.901
classification report is
              precision    recall  f1-score   support

           0       0.71      0.10      0.17      4766
           1       0.90      1.00      0.95     40670

    accuracy                           0.90     45436
   macro avg       0.81      0.55      0.56     45436
weighted avg       0.88      0.90      0.87     45436

confusion matrix is
[[  454  4312]
 [  181 40489]]
```



```
In [198]: rf.fit(x_train,y_train)
          pred=rf.predict(x_test)
          print('accuracy score through random forest is ')
          print(round((accuracy_score(y_test,pred)),3))
          print('classification report is')
          print(classification_report(y_test,pred))
          print('confusion matrix is')
          print(confusion_matrix(y_test,pred))
          print('\n')
```

```
accuracy score through random forest is 
0.906
classification report is
              precision    recall  f1-score   support

           0       0.60      0.31      0.41      4766
           1       0.92      0.98      0.95     40670

    accuracy                           0.91     45436
   macro avg       0.76      0.64      0.68     45436
weighted avg       0.89      0.91      0.89     45436

confusion matrix is
[[ 1478  3288]
 [ 1000 39670]]
```

Checking for cross validations

```
In [199]: #cross val score for lg
          score=cross_val_score(lg,x1,y,cv=5)
          print(lg,'score is:')
          print(round((score.mean()),3))
          print('\n')
```

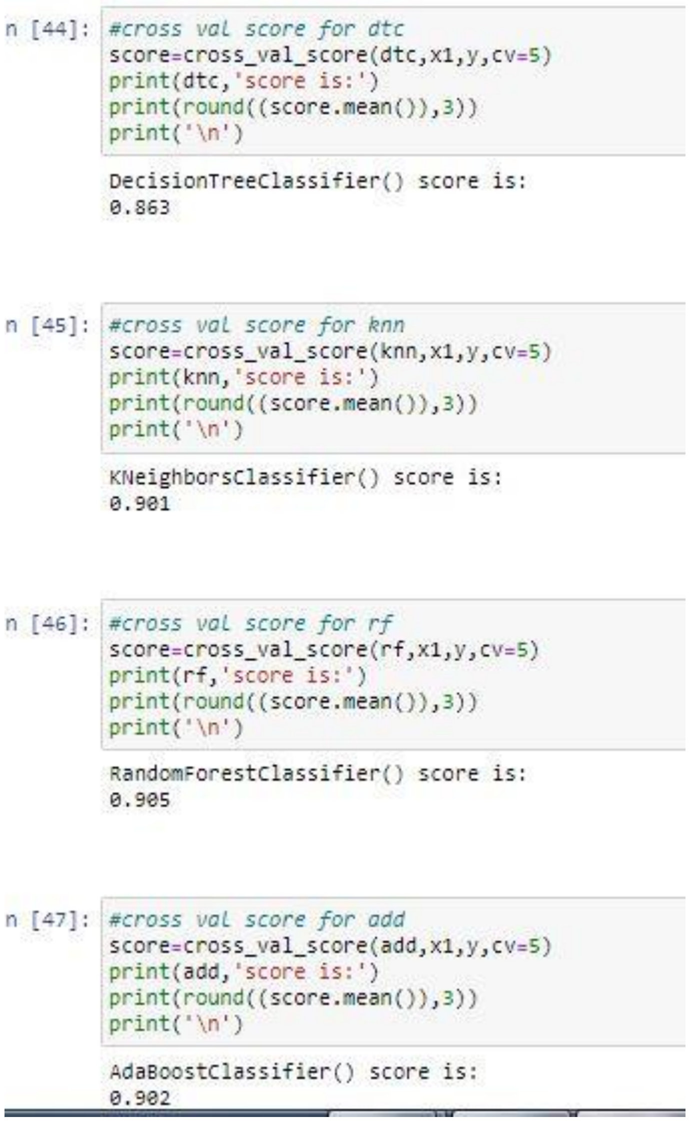
```
LogisticRegression() score is:
0.896
```

```
In [200]: #cross val score for gnb
          score=cross_val_score(gnb,x1,y,cv=5)
          print(gnb,'score is:')
          print(round((score.mean()),3))
          print('\n')
```

```
GaussianNB() score is:
0.751
```

```
In [43]: #cross val score for svc
         clf = LinearSVC(random_state=0, tol=1e-5)
         score=cross_val_score(clf,x1,y,cv=5)
         print(clf,'score is:')
         print(round((score.mean()),3))
         print('\n')
```

```
LinearSVC(random_state=0, tol=1e-05) score is:
0.883
```



```
n [44]: #cross val score for dtc
        score=cross_val_score(dtc,x1,y,cv=5)
        print(dtc,'score is:')
        print(round((score.mean()),3))
        print('\n')

        DecisionTreeClassifier() score is:
        0.863
```

```
n [45]: #cross val score for knn
        score=cross_val_score(knn,x1,y,cv=5)
        print(knn,'score is:')
        print(round((score.mean()),3))
        print('\n')

        KNeighborsClassifier() score is:
        0.901
```

```
n [46]: #cross val score for rf
        score=cross_val_score(rf,x1,y,cv=5)
        print(rf,'score is:')
        print(round((score.mean()),3))
        print('\n')

        RandomForestClassifier() score is:
        0.905
```

```
n [47]: #cross val score for add
        score=cross_val_score(add,x1,y,cv=5)
        print(add,'score is:')
        print(round((score.mean()),3))
        print('\n')

        AdaBoostClassifier() score is:
        0.902
```

- Key Metrics for success in solving problem under consideration

  The key metrics used is Logistic Regression, its performing better than other models, even though we got high accuracy in other models but their difference between cross Val score and accuracy score is large so Logistic Regression fit the best model position for the current project.

- Visualizations

  Major plots used are:

  1.Scatterplot

2.heatmap

3.corrplot

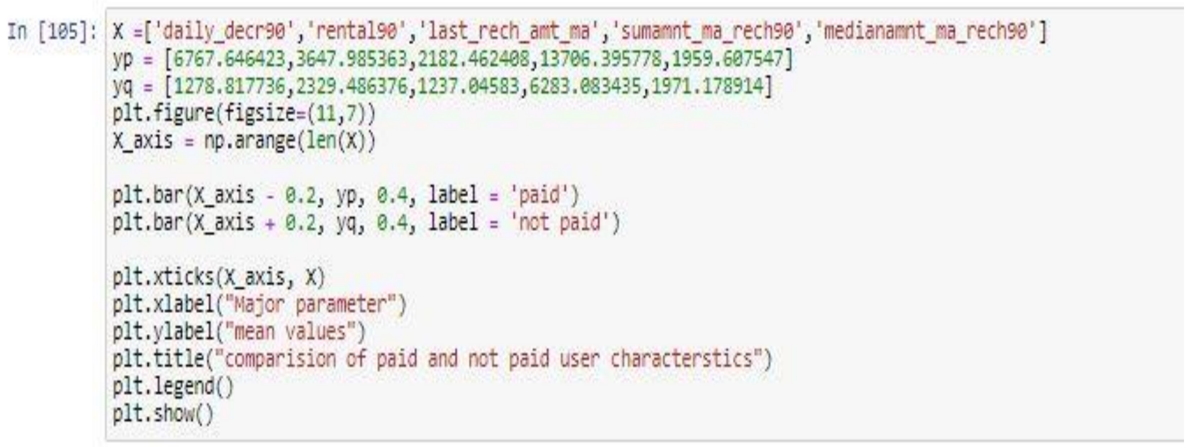
4.distplot

5.boxplot

6.relplot

Few examples of Visualization:

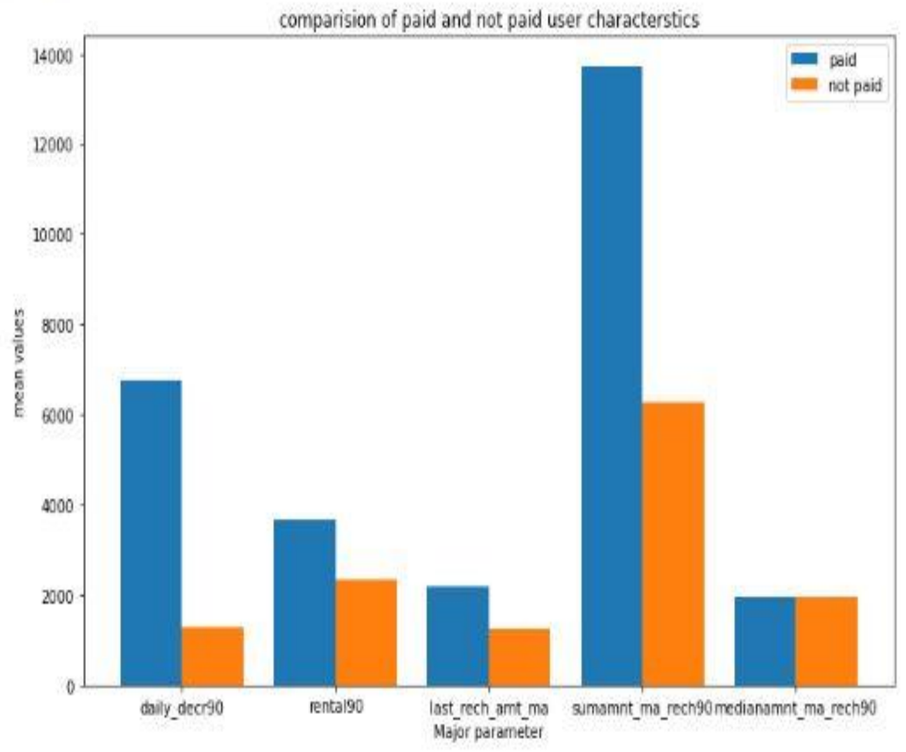
1.Comparision of paid and not paid user



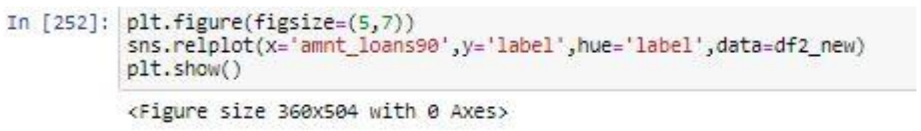
```
In [105]: X =['daily_decr90','rental90','last_rech_amt_ma','sumamnt_ma_rech90','medianamnt_ma_rech90']
          yp = [6767.646423,3647.985363,2182.462408,13706.395778,1959.607547]
          yq = [1278.817736,2329.486376,1237.04583,6283.083435,1971.178914]
          plt.figure(figsize=(11,7))
          X_axis = np.arange(len(X))

          plt.bar(X_axis - 0.2, yp, 0.4, label = 'paid')
          plt.bar(X_axis + 0.2, yq, 0.4, label = 'not paid')

          plt.xticks(X_axis, X)
          plt.xlabel("Major parameter")
          plt.ylabel("mean values")
          plt.title("comparision of paid and not paid user characterstics")
          plt.legend()
          plt.show()
```

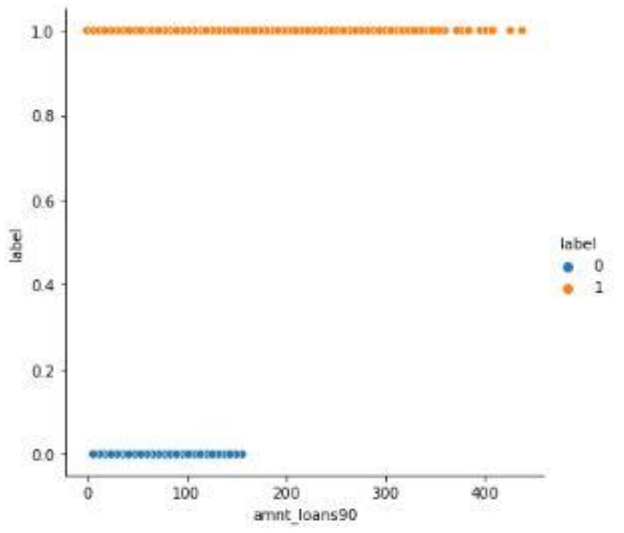


This graph shows the characterstics of paid and not paid user for some of the major characterstics which are highly correlated to target variable

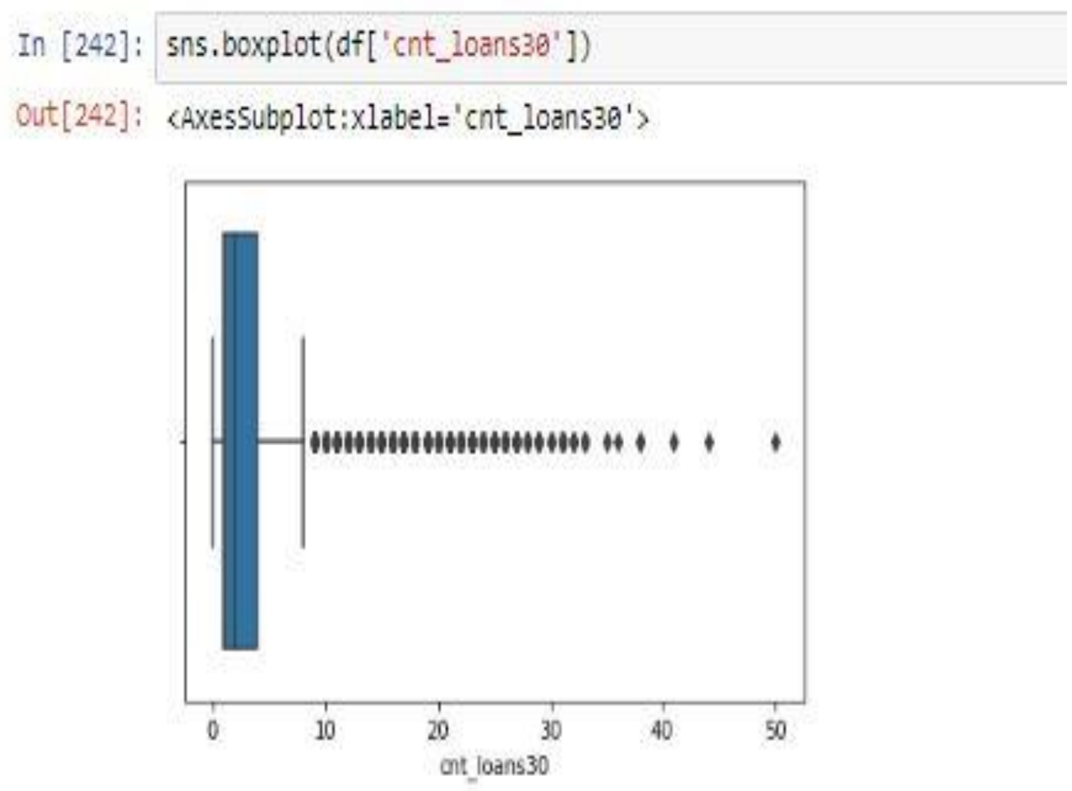


```
In [252]: plt.figure(figsize=(5,7))
          sns.relplot(x='amnt_loans90',y='label',hue='label',data=df2_new)
          plt.show()

<Figure size 360x504 with 0 Axes>
```



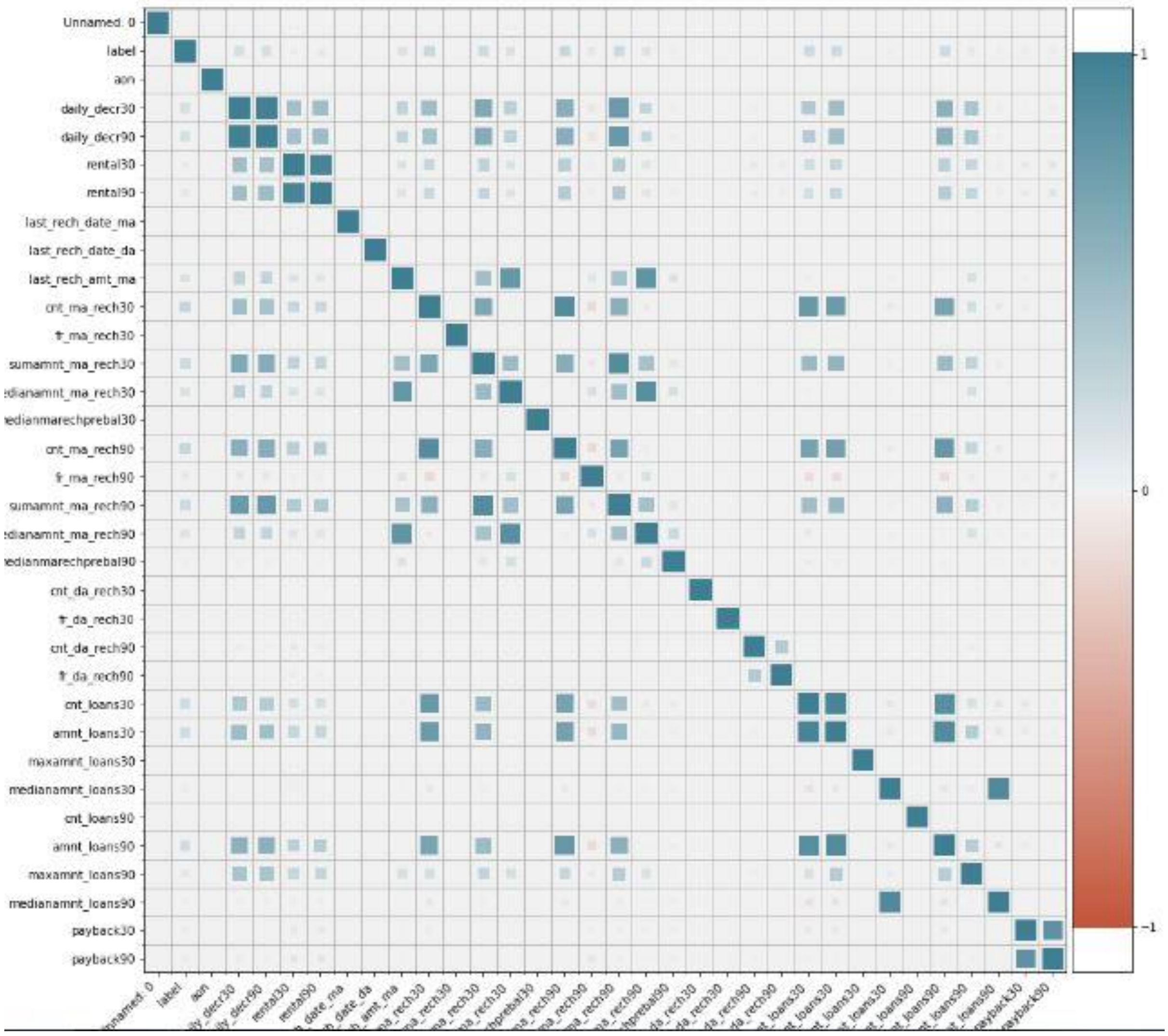
This is a relationship depicting plot between amount of loans taken in 90 days and label, this graph clearly shows that those people who taken amount of loans more than 160 wont belongs to non-pay group.



```
In [242]: sns.boxplot(df['cnt_loans30'])

Out[242]: <AxesSubplot:xlabel='cnt_loans30'>
```

This graph is boxplot and it is used for showing the outliers,



This is a corrplot which shows how strongly the variables are correlated.

CONCLUSION

- Interpretation of the Results

The major result what we have got from project

1. The user who pay back have a mean daily decr value around 6. 4k but the other category has around 1.7k.
2. Also, in case of sumamnt_ma_rech90 the paid user has a mean value double the non-paid user.

3. We can see that the medianamnt_ma_rech90 gives no informa tion regarding whether the user going to pay or not so we can drop that.
4. Those users who spend daily amount from main account, and have an average more than a lakh will always belongs to label 1.
5. Those people who have recharged their main account in last 3 months for more than 50 times always belongs to label 1
6. In case of total number of loans taken by user is more than 170 in 90 days then that user certainly belongs to label 1.

- Learning Outcomes of the Study in respect of Data Science

  The challenging part is data processing, since it’s a huge data it used to take a lot of time for processing, the major learning is that how to look at the data and study different variables and its effect over target variable, leant new library plotly.

- Limitations of this work and Scope for Future Work

  The solution provided has got an accuracy of 0.895, we need some tool for processing the data fastly so that we can hyper tune with more parameters, we can go for hypothesis at initial stage of project by taking only 1/4th of sample data and proving that it is going to equal the feature of total data, so that processing speed can be increased.

Thank you!