FLIP ROBO

Micro Credit Defaulter Prediction

Submitted by:

ASHISH YADAV

ACKNOWLEDGMENT

We would like to express our deep and sincere gratitude to FlipnWork for giving us the opportunity to do this project. As a great bridge between academic and industry, this program educated us how to perform theoretical methodology in real life. We would like to express our sincere thankfulness to my mentor Sapna Verma for the continuous support , for their patience, enthusiasm, motivation and immense knowledge. As our academic mentor, Sapna Verma supported and helped in this project. Additionally, we would also like to thank all our friends who offered us some help, such as Divyanshu who helped me during this project

INTRODUCTION

Problem Statement :

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 donorsare 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 MFShas been uneven with both significant challenges and successes.

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

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

They understand the importance of communication and how it affects a person’s life, thus, focusing on providing their services and products to low income families and poor customers that can help them in the need of hour.

They arecollaborating 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 be6(in Indonesian Rupiah), while, for the loan amount of 10(in Indonesian Rupiah), the payback amount should be 12(in Indonesian Rupiah).

The sample data is provided to us from our client database. It is hereby given to you for this exercise. 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.

Exercise:

Build a model which can be used to predict in terms of a probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance ofloan. In this case, 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.

Business Goal :

Using micro credit as a poverty-reduction tool,by focusing on providing their services and products to low income families and poor customers that can help them in the need of hour.

Domain Understanding :

The telecom sector continues to be at the epicenter for growth, innovation, and disruption for virtually any industry. Mobile devices and related broadband connectivity continue to be more and more embedded in the fabric of society today and they are key in driving the momentum around some key trends such as video streaming, Internet of Things (IoT), and mobile payments. Our client is also a telecom player . They are a fixed wireless telecommunications network provider. They have launched various products and have developed its business and organization based on the budget operator model, offering better products at Lower Prices to all value conscious customers through a strategy of disruptive innovation that focuses on the subscriber.

They understand the importance of communication and how it affects a person's life, thus, focusing on providing their services and products to low income families and poor customers that can help them in the need of hour.

Literature :

- The main steps in our research were the following.

- **Exploratory Data Analysis (EDA):** By conducting explanatory data analysis, we obtain a better understanding of our data. This yields insights that can be helpful later when building a model, as well as insights that are independently interesting.

- **Balancing Dataset :** In order to balance the imbalance dataset ,we use technique like SMOTE.

- **Modeling:** We apply Decision Tree , Logical Regression models for prediction of the micro credit defaulter prediction

Analytical Problem Framing

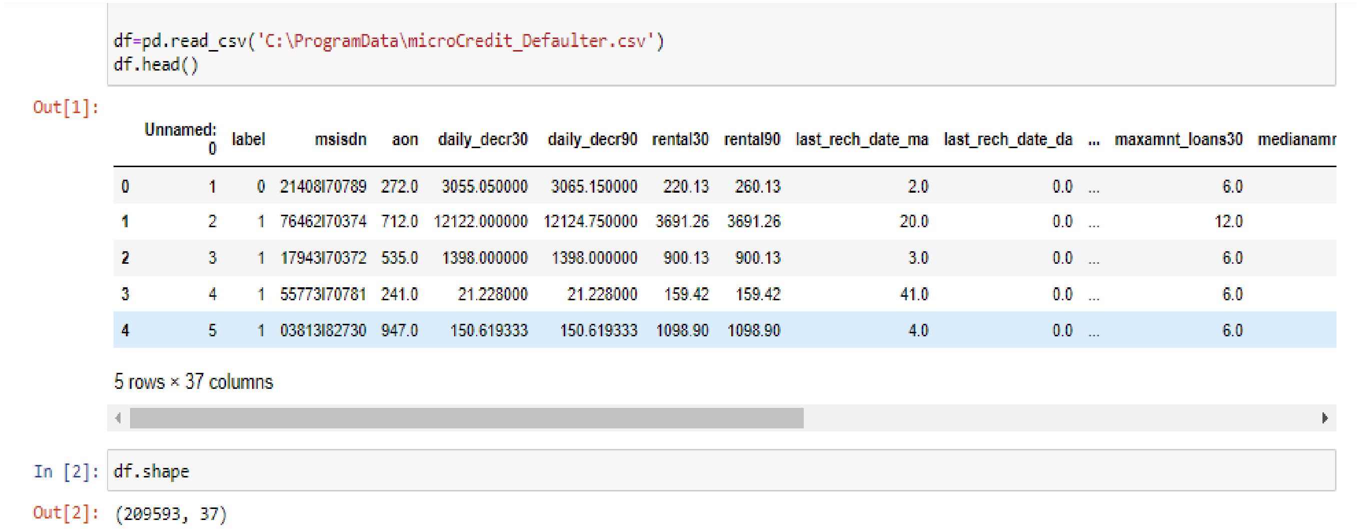
• Mathematical/ Analytical Modeling of the Problem

Telecom Industry client is collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days.

Client is a fixed wireless telecommunications network provider. They have launched various products and have developed its business and organization based on the budget operator model, offering better products at Lower Prices to all value conscious customers through a strategy of disruptive innovation that focuses on the subscriber.

They understand the importance of communication and how it affects a person's life, thus, focusing on providing their services and products to low income families and poor customers that can help them in the need of hour.

Dataset :



```
df=pd.read_csv('C:\ProgramData\microCredit_Defaulter.csv')
df.head()
```

Out[1]:

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

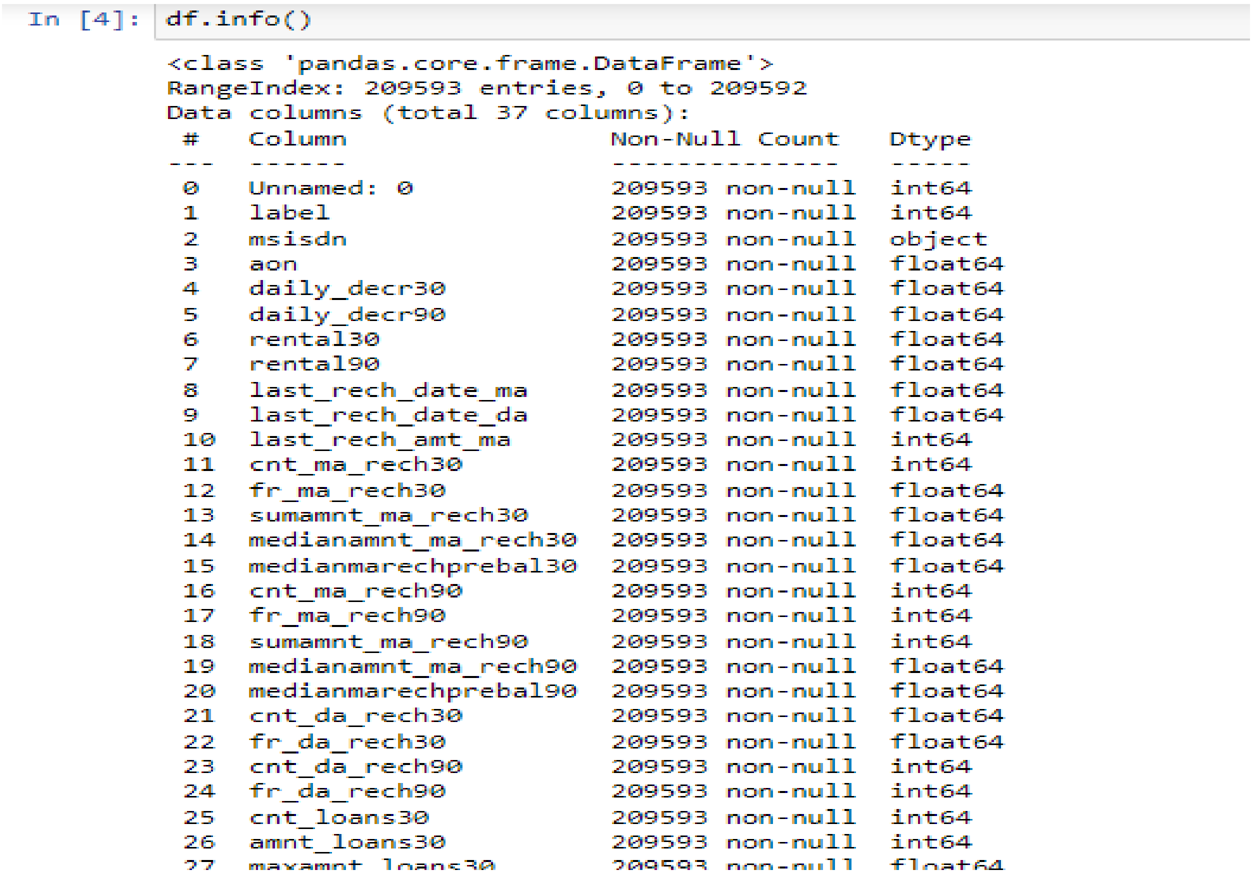
5 rows × 37 columns

In [2]:
```
df.shape
```

Out[2]:
```
(209593, 37)
```

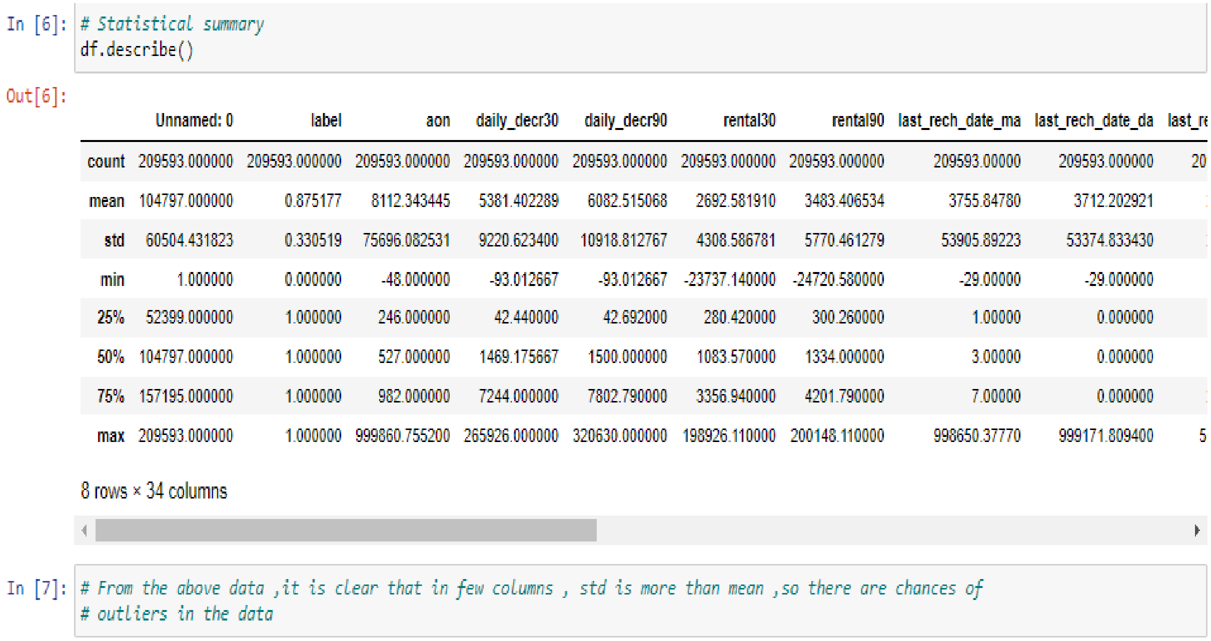
EXPLORATORY DATA ANALYSIS

- Data exploration is the first step in data analysis and typically involves summarizing the main characteristics of a data set, including its size, accuracy, initial patterns in the data and other attributes. It is commonly conducted by data analysts using visual analytics tools, but it can also be done in more advanced statistical software, Python. Before it can conduct analysis on data collected by multiple data sources and stored in data warehouses, an organization must know how many cases are in a data set, what variables are included, how many missing values there are and what general hypotheses the data is likely to support. An initial exploration of the data set can help answer these questions by familiarizing analysts with the data with which they are working.



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

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209593 entries, 0 to 209592
Data columns (total 37 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   Unnamed: 0            209593 non-null  int64  
 1   label                 209593 non-null  int64  
 2   msisdn                209593 non-null  object 
 3   aon                   209593 non-null  float64
 4   daily_decr30          209593 non-null  float64
 5   daily_decr90          209593 non-null  float64
 6   rental30              209593 non-null  float64
 7   rental90              209593 non-null  float64
 8   last_rech_date_ma     209593 non-null  float64
 9   last_rech_date_da     209593 non-null  float64
 10  last_rech_amt_ma      209593 non-null  int64  
 11  cnt_ma_rech30         209593 non-null  int64  
 12  fr_ma_rech30          209593 non-null  float64
 13  sumamnt_ma_rech30     209593 non-null  float64
 14  medianamnt_ma_rech30  209593 non-null  float64
 15  medianmarechprebal30  209593 non-null  float64
 16  cnt_ma_rech90         209593 non-null  int64  
 17  fr_ma_rech90          209593 non-null  int64  
 18  sumamnt_ma_rech90     209593 non-null  int64  
 19  medianamnt_ma_rech90  209593 non-null  float64
 20  medianmarechprebal90  209593 non-null  float64
 21  cnt_da_rech30         209593 non-null  float64
 22  fr_da_rech30          209593 non-null  float64
 23  cnt_da_rech90         209593 non-null  int64  
 24  fr_da_rech90          209593 non-null  int64  
 25  cnt_loans30           209593 non-null  int64  
 26  amnt_loans30          209593 non-null  int64  
 27  maxamnt_loans30       209593 non-null  float64
```

Statistical Summary :



In [6]:

```
# Statistical summary
df.describe()
```

Out[6]:

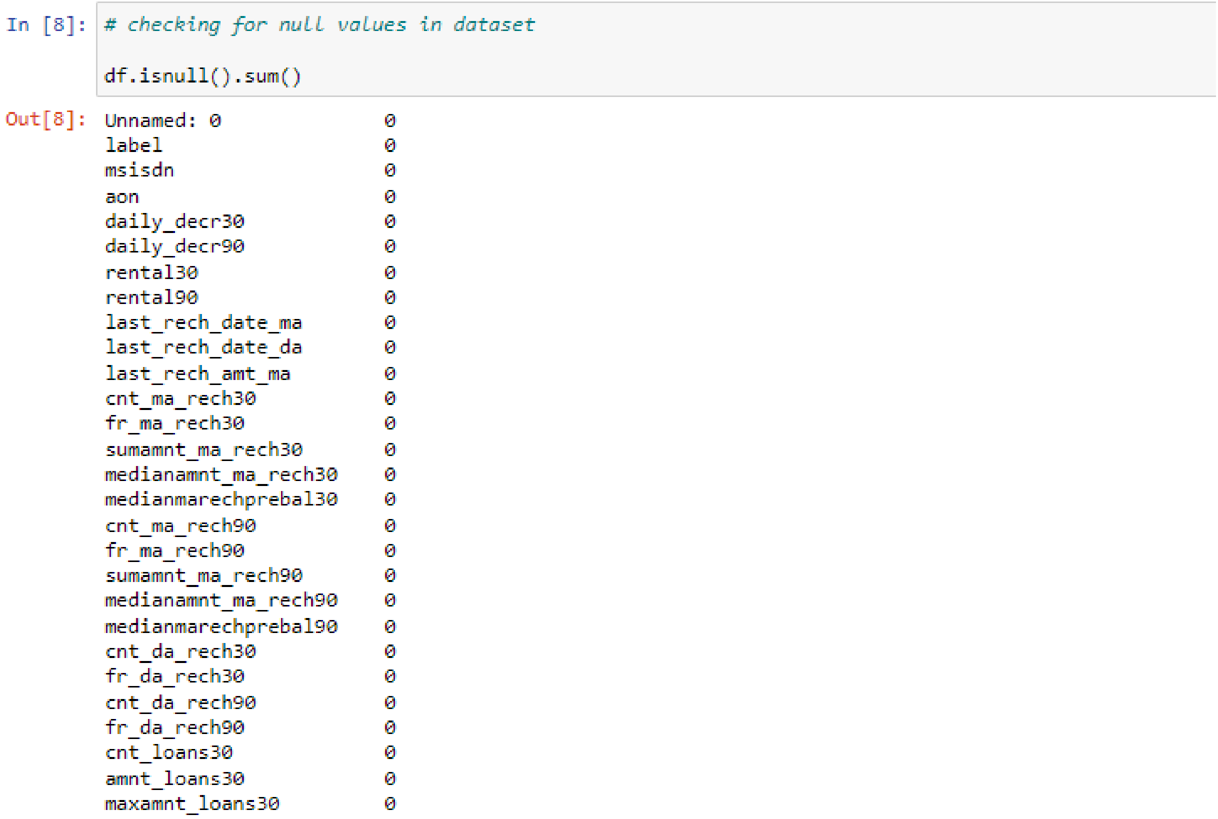
	Unnamed: 0	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_re
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.00000	209593.000000	20
mean	104797.000000	0.875177	8112.343445	5381.402289	6082.515068	2692.581910	3483.406534	3755.84780	3712.202921	
std	60504.431823	0.330519	75696.082531	9220.623400	10918.812767	4308.586781	5770.461279	53905.89223	53374.833430	
min	1.000000	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.00000	-29.000000	
25%	52399.000000	1.000000	246.000000	42.440000	42.692000	280.420000	300.260000	1.00000	0.000000	
50%	104797.000000	1.000000	527.000000	1469.175667	1500.000000	1083.570000	1334.000000	3.00000	0.000000	
75%	157195.000000	1.000000	982.000000	7244.000000	7802.790000	3356.940000	4201.790000	7.00000	0.000000	
max	209593.000000	1.000000	999860.755200	265926.000000	320630.000000	198926.110000	200148.110000	998650.37770	999171.809400	5

8 rows × 34 columns

In [7]:

```
# From the above data ,it is clear that in few columns , std is more than mean ,so there are chances of
# outliers in the data
```

Checking null values in dataset



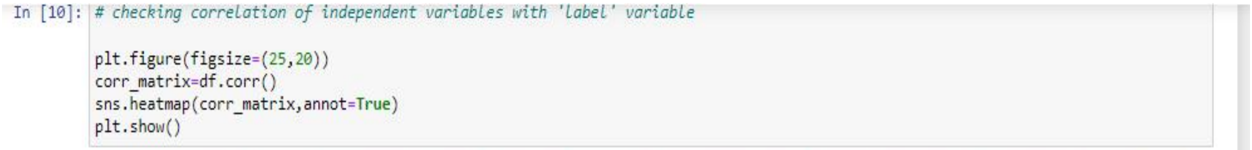
In [8]:

```
# checking for null values in dataset

df.isnull().sum()
```

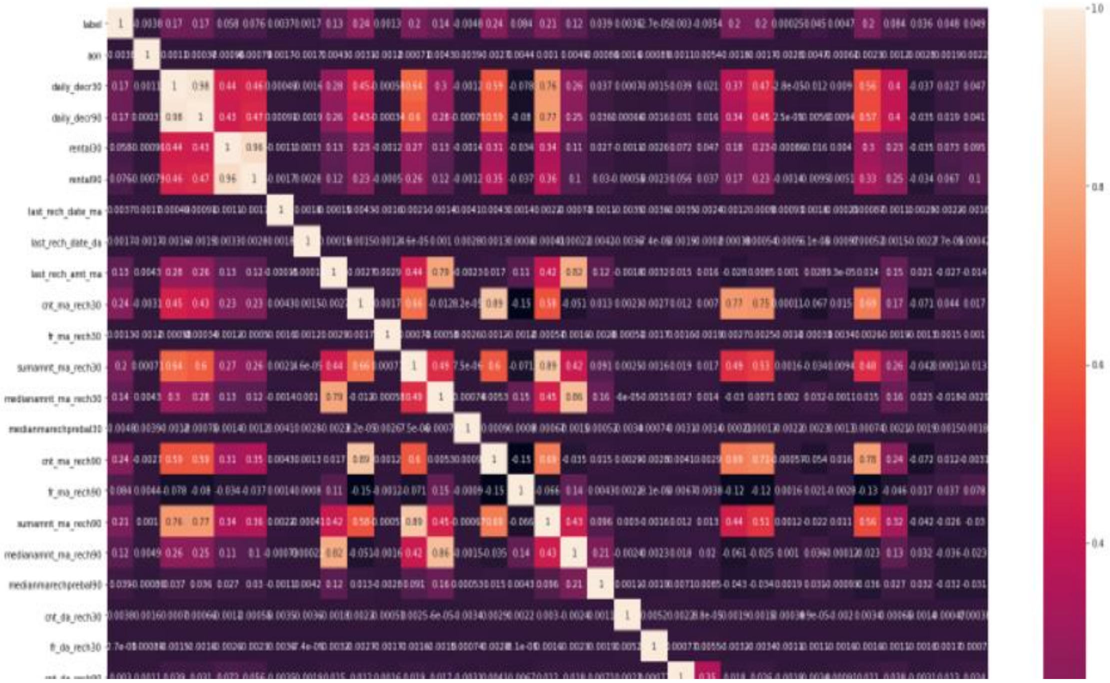
Out[8]:

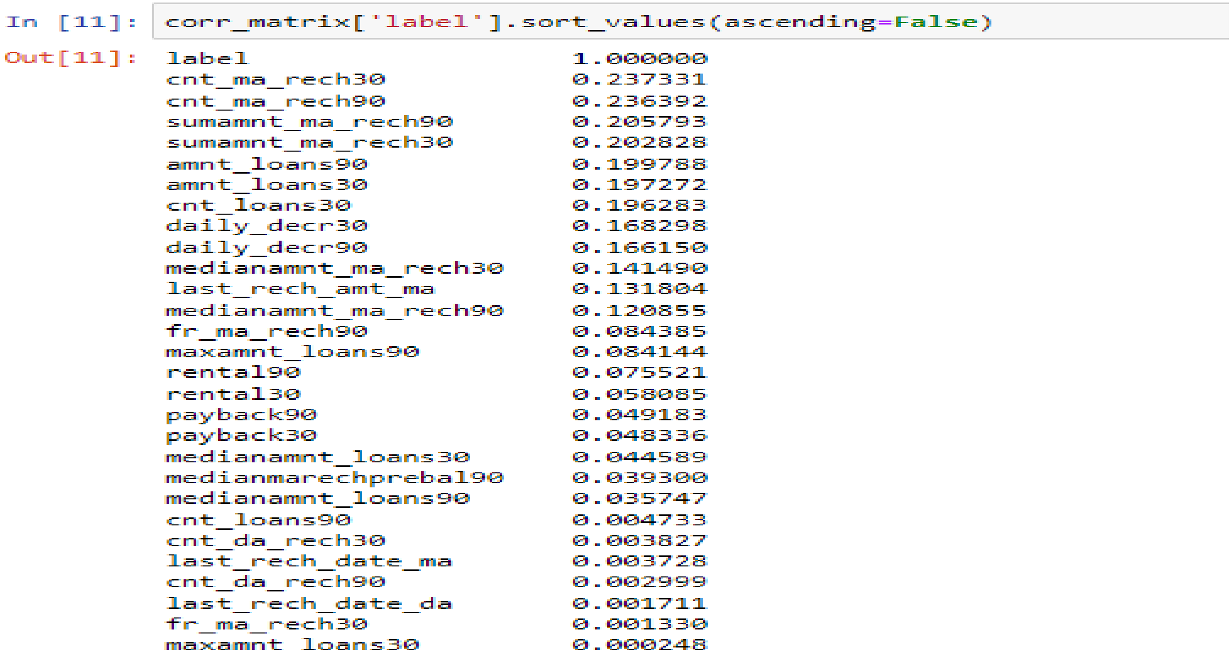
```
Unnamed: 0              0
label                   0
msisdn                  0
aon                     0
daily_decr30            0
daily_decr90            0
rental30                0
rental90                0
last_rech_date_ma       0
last_rech_date_da       0
last_rech_amt_ma        0
cnt_ma_rech30           0
fr_ma_rech30            0
sumamnt_ma_rech30       0
medianamnt_ma_rech30    0
medianmarechprebal30    0
cnt_ma_rech90           0
fr_ma_rech90            0
sumamnt_ma_rech90       0
medianamnt_ma_rech90    0
medianmarechprebal90    0
cnt_da_rech30           0
fr_da_rech30            0
cnt_da_rech90           0
fr_da_rech90            0
cnt_loans30             0
amnt_loans30            0
maxamnt_loans30         0
```

CORRELATION :



```
In [10]: # checking correlation of independent variables with 'label' variable

         plt.figure(figsize=(25,20))
         corr_matrix=df.corr()
         sns.heatmap(corr_matrix,annot=True)
         plt.show()
```



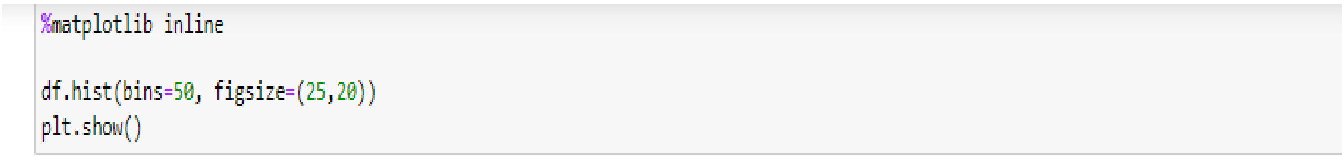


```
In [11]: corr_matrix['label'].sort_values(ascending=False)
Out[11]: label                      1.000000
         cnt_ma_rech30              0.237331
         cnt_ma_rech90              0.236392
         sumamnt_ma_rech90          0.205793
         sumamnt_ma_rech30          0.202828
         amnt_loans90               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
```

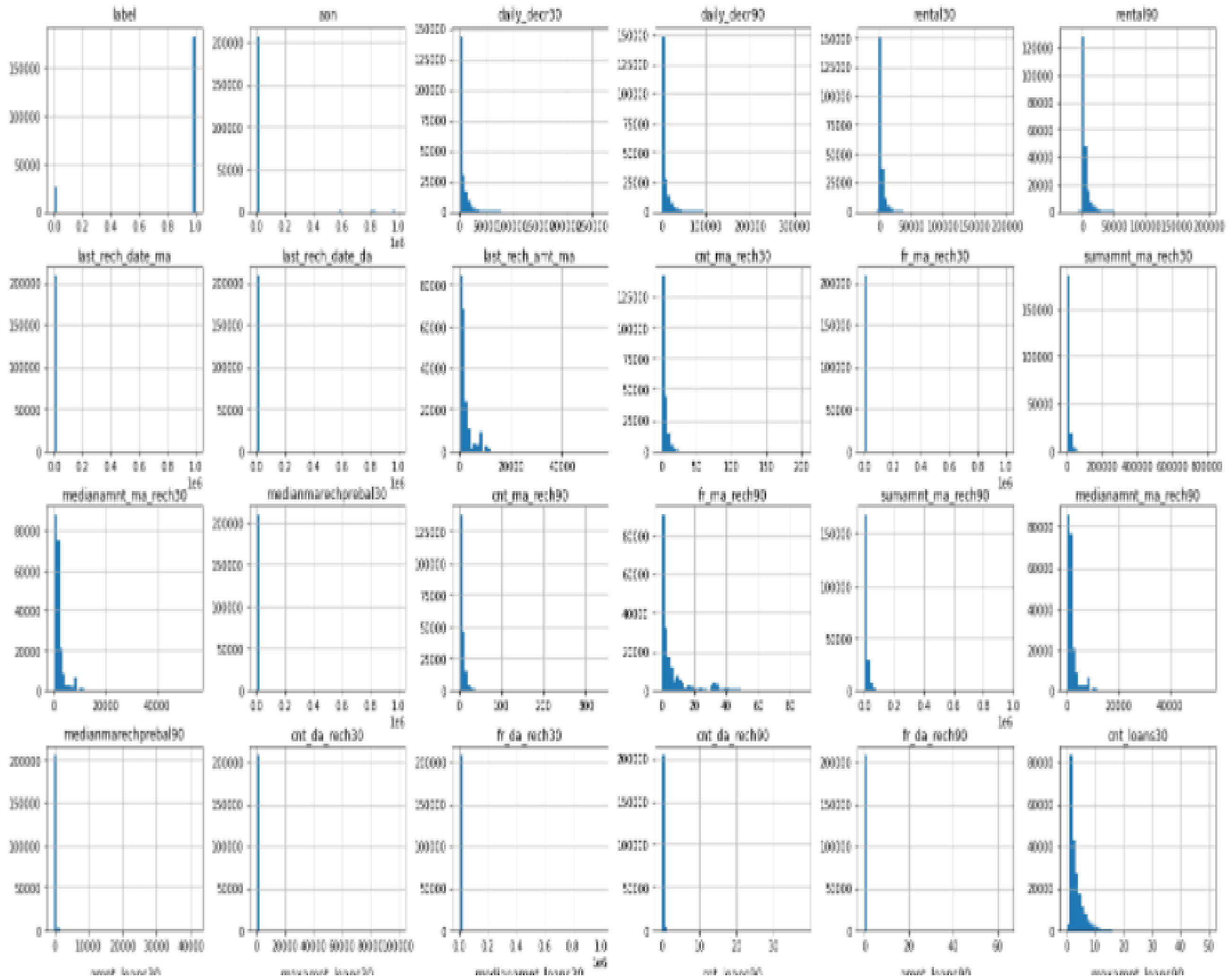
DATA VISUALIZATION :

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. In the world of Big Data, data visualization tools and technologies are essential to analyse massive amounts of information and make data-driven decisions.

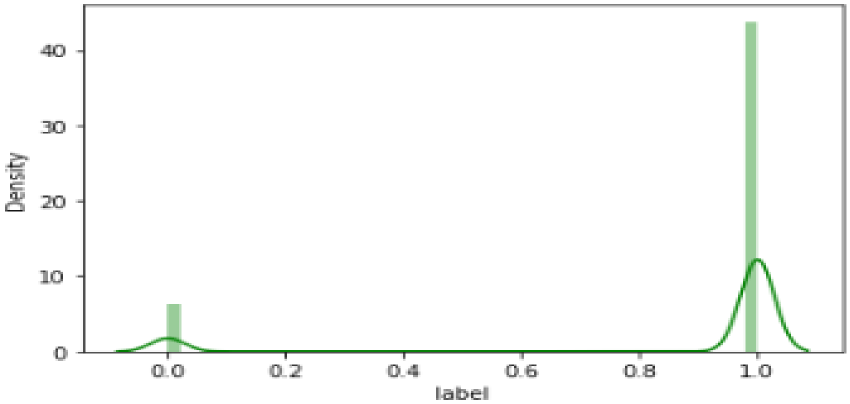
Distribution Plots :



```
%matplotlib inline

df.hist(bins=50, figsize=(25,20))
plt.show()
```

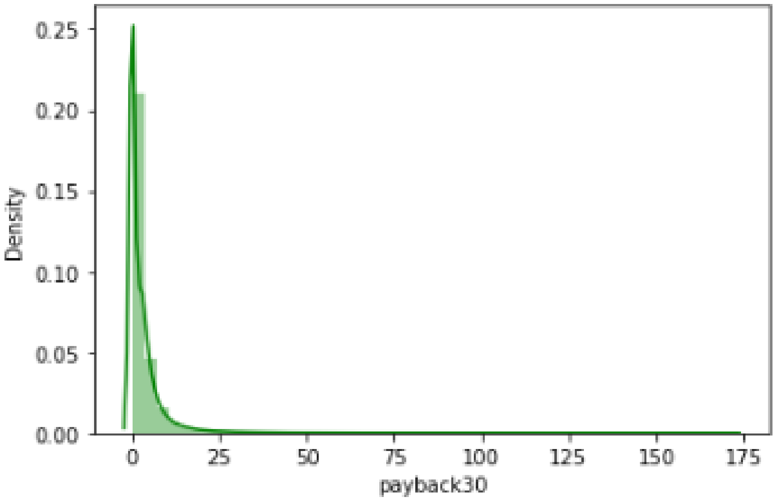


```
In [15]: # checking distibution of label feature

         sns.distplot(df['label'], color = 'green')

Out[15]: <AxesSubplot:xlabel='label', ylabel='Density'>
```

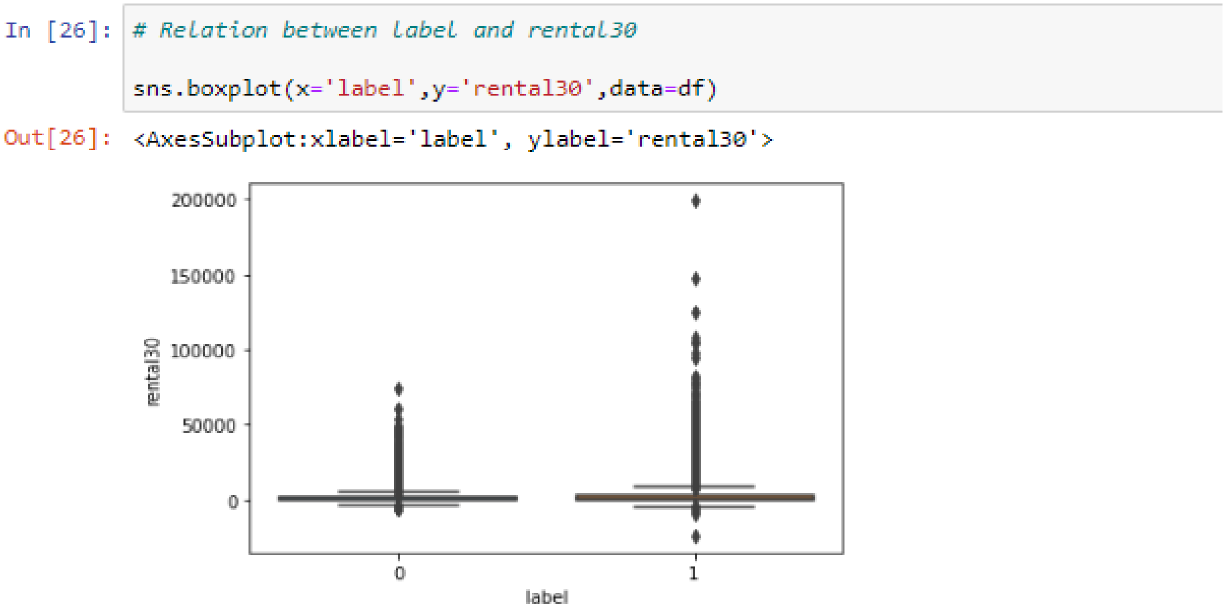


```
In [16]: df['label'].value_counts()

Out[16]: 1     183431
         0      26162
         Name: label, dtype: int64
```

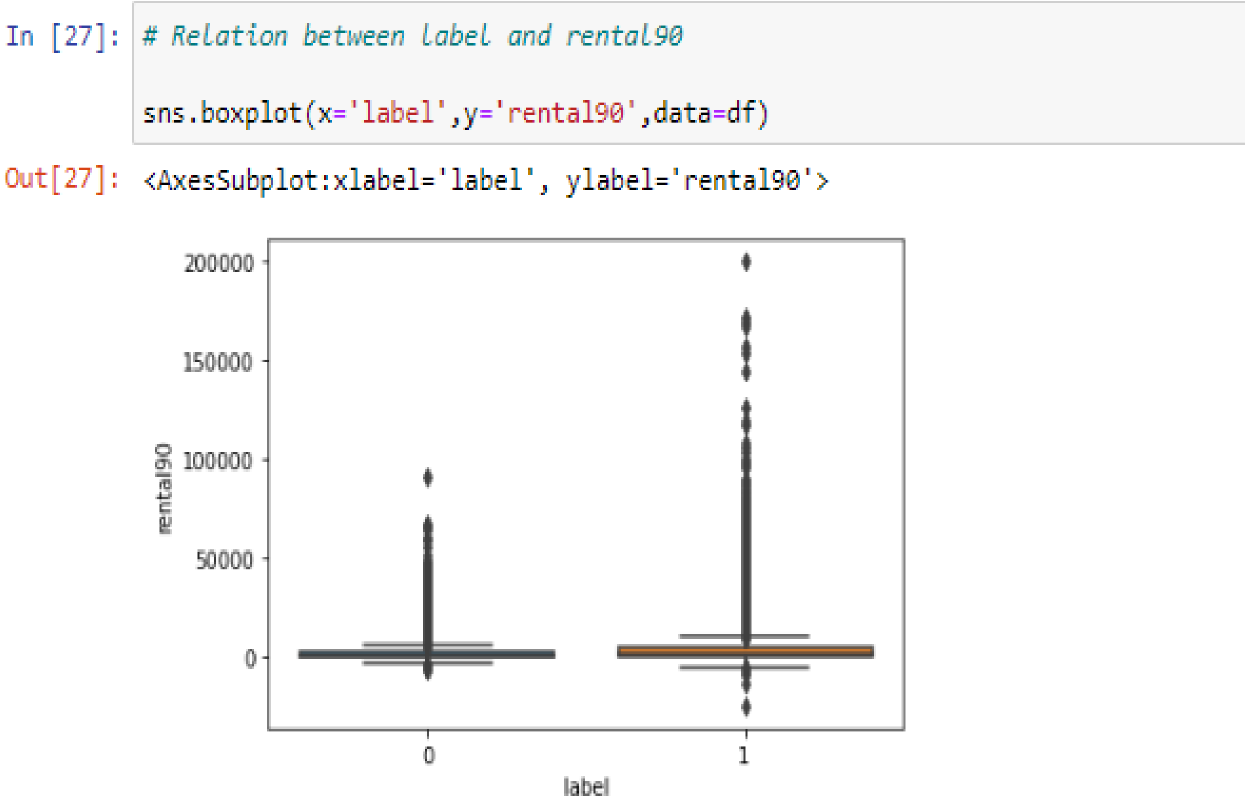
```
In [24]: # checking distibution of payback30 feature

         sns.distplot(df['payback30'], color = 'green')

Out[24]: <AxesSubplot:xlabel='payback30', ylabel='Density'>
```



BOX PLOTS :



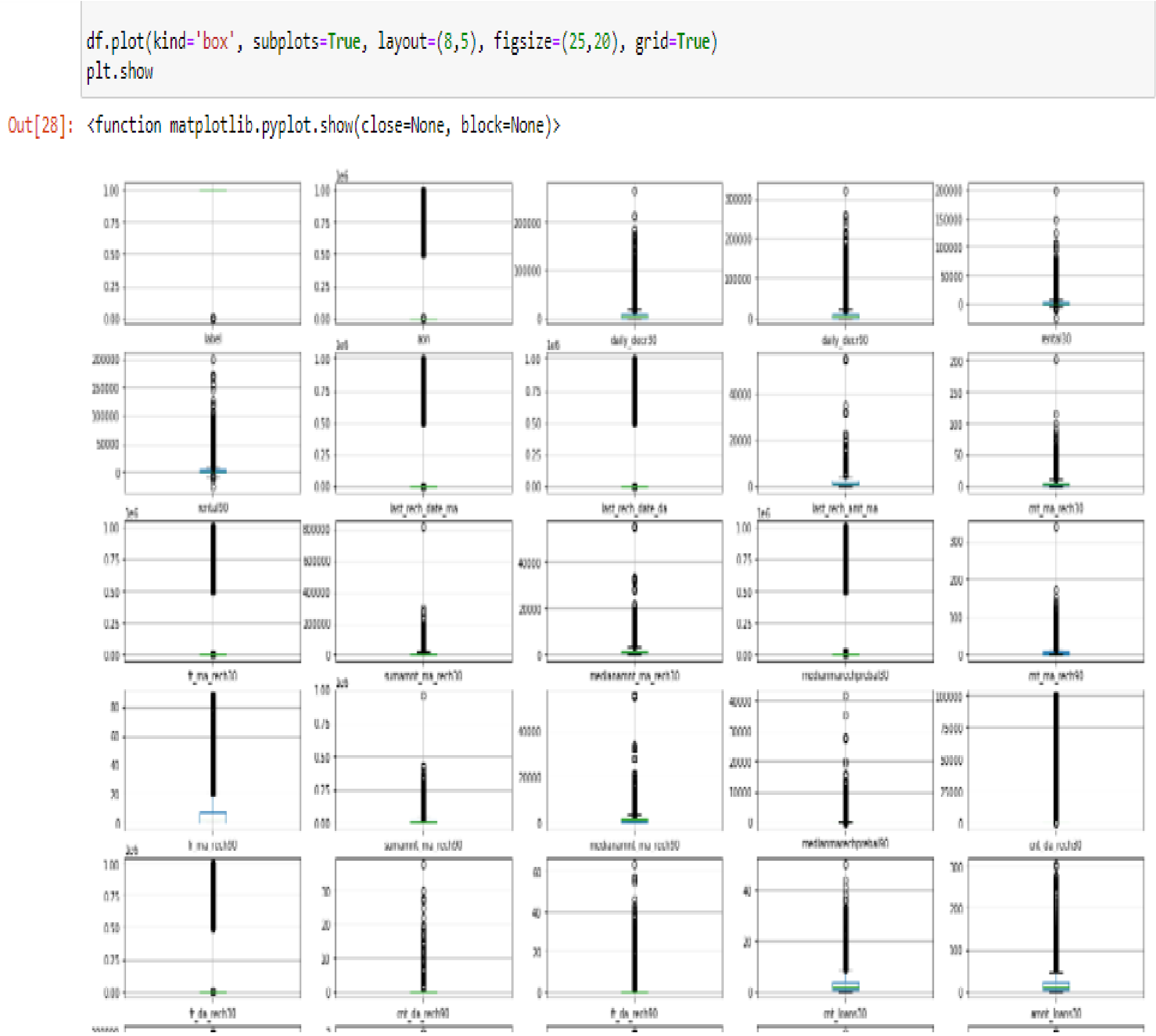
```
In [26]: # Relation between label and rental30

         sns.boxplot(x='label',y='rental30',data=df)

Out[26]: <AxesSubplot:xlabel='label', ylabel='rental30'>
```



```
In [27]: # Relation between label and rental90

         sns.boxplot(x='label',y='rental90',data=df)

Out[27]: <AxesSubplot:xlabel='label', ylabel='rental90'>
```

OUTLIERS :

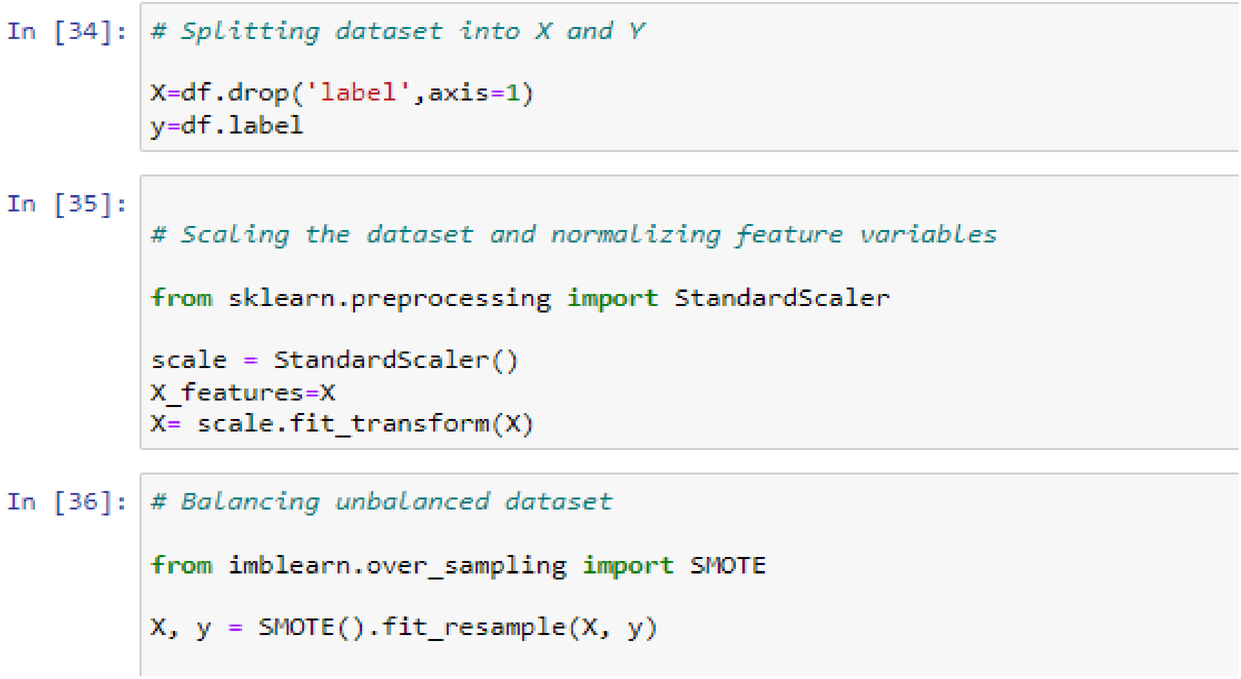
- An outlier is an object that deviates significantly from the rest of the objects. They can be caused by measurement or execution error. The analysis of outlier data is referred to as outlier analysis or outlier mining.
- It is a data point that is noticeably different from the rest. They represent errors in measurement, bad data collection, or simply show variables not considered when collecting the data.



```
df.plot(kind='box', subplots=True, layout=(8,5), figsize=(25,20), grid=True)
plt.show
```

```
Out[28]: <function matplotlib.pyplot.show(close=None, block=None)>
```

Splitting,scaling,balancing Dataset :

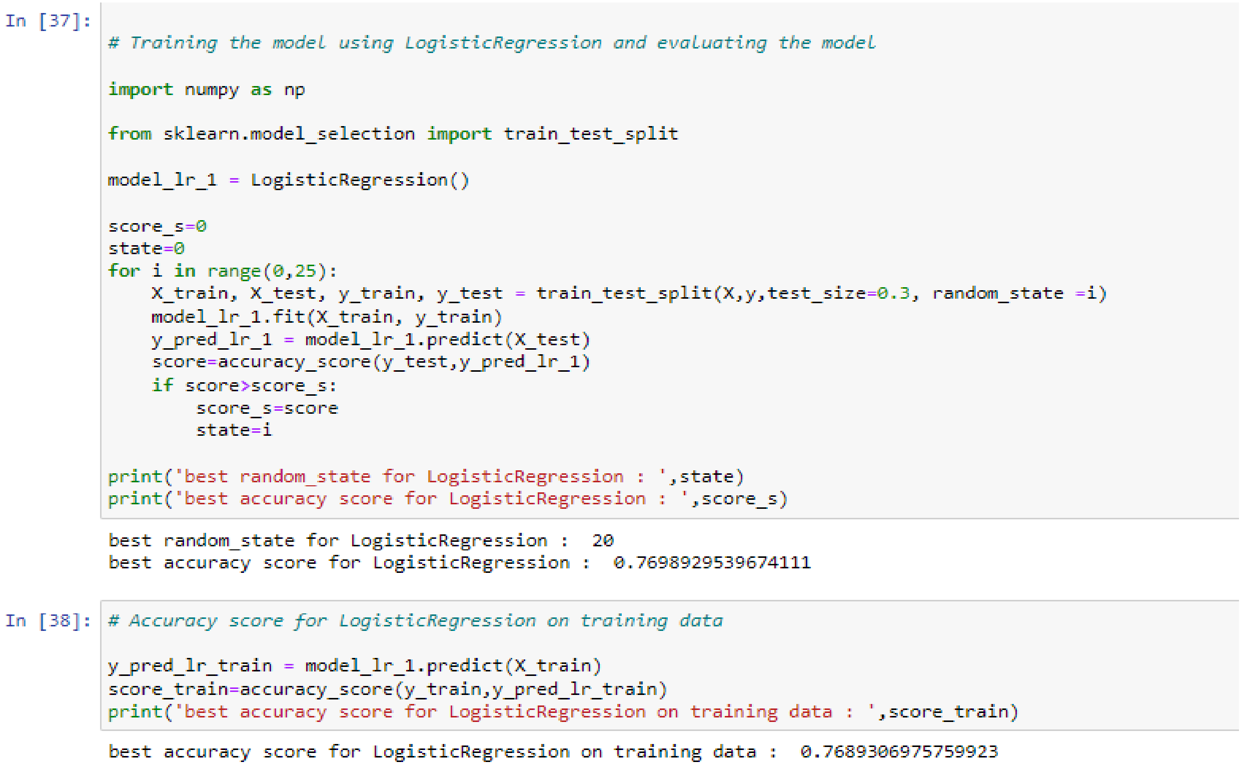


```
In [34]: # Splitting dataset into X and Y

         X=df.drop('label',axis=1)
         y=df.label
```

```
In [35]: 
         # Scaling the dataset and normalizing feature variables

         from sklearn.preprocessing import StandardScaler

         scale = StandardScaler()
         X_features=X
         X= scale.fit_transform(X)
```

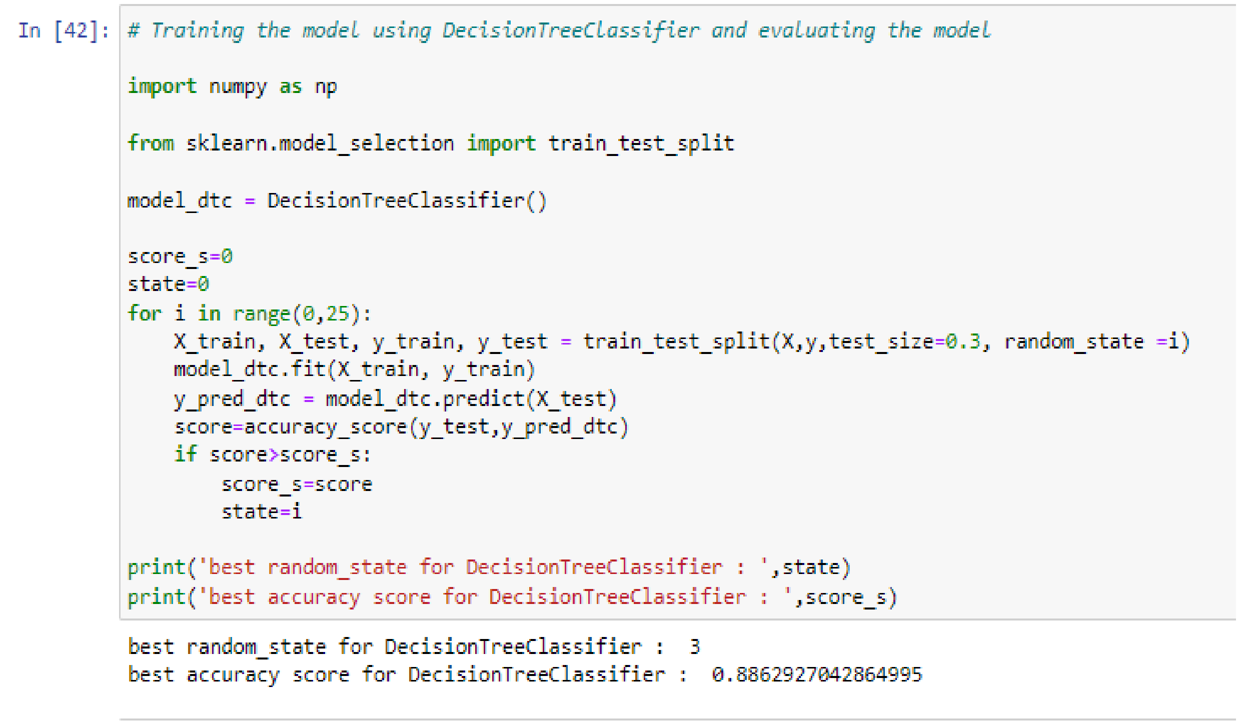
```
In [36]: # Balancing unbalanced dataset

         from imblearn.over_sampling import SMOTE

         X, y = SMOTE().fit_resample(X, y)
```

EVALUATION OF MODELS

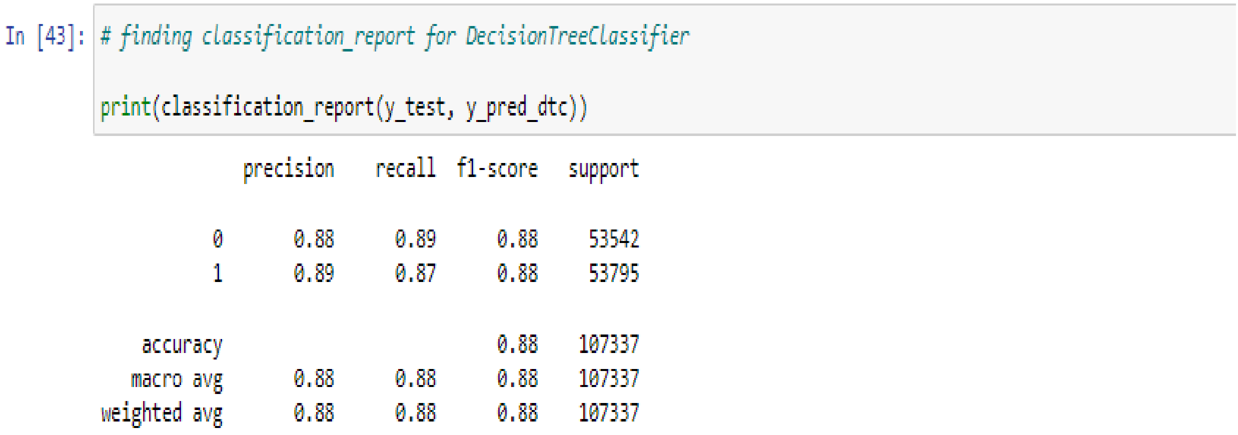


```
In [37]: 
         # Training the model using LogisticRegression and evaluating the model

         import numpy as np

         from sklearn.model_selection import train_test_split

         model_lr_1 = LogisticRegression()

         score_s=0
         state=0
         for i in range(0,25):
             X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state =i)
             model_lr_1.fit(X_train, y_train)
             y_pred_lr_1 = model_lr_1.predict(X_test)
             score=accuracy_score(y_test,y_pred_lr_1)
             if score>score_s:
                 score_s=score
                 state=i

         print('best random_state for LogisticRegression : ',state)
         print('best accuracy score for LogisticRegression : ',score_s)

         best random_state for LogisticRegression :  20
         best accuracy score for LogisticRegression :  0.7698929539674111
```

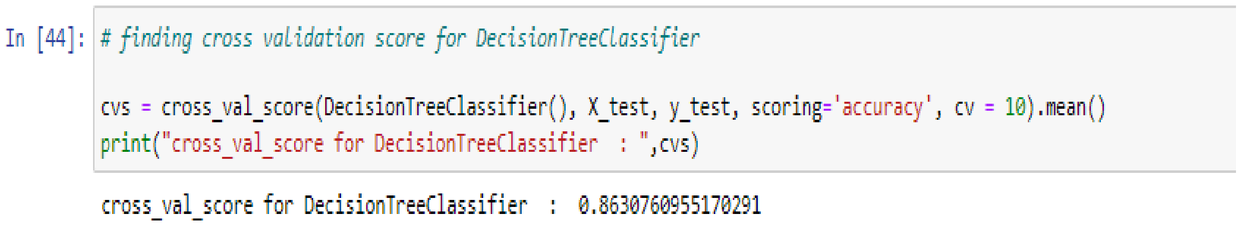
```
In [38]: # Accuracy score for LogisticRegression on training data

         y_pred_lr_train = model_lr_1.predict(X_train)
         score_train=accuracy_score(y_train,y_pred_lr_train)
         print('best accuracy score for LogisticRegression on training data : ',score_train)

         best accuracy score for LogisticRegression on training data :  0.7689306975759923
```



```
In [42]: # Training the model using DecisionTreeClassifier and evaluating the model

         import numpy as np

         from sklearn.model_selection import train_test_split

         model_dtc = DecisionTreeClassifier()

         score_s=0
         state=0
         for i in range(0,25):
             X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state =i)
             model_dtc.fit(X_train, y_train)
             y_pred_dtc = model_dtc.predict(X_test)
             score=accuracy_score(y_test,y_pred_dtc)
             if score>score_s:
                 score_s=score
                 state=i

         print('best random_state for DecisionTreeClassifier : ',state)
         print('best accuracy score for DecisionTreeClassifier : ',score_s)

         best random_state for DecisionTreeClassifier :  3
         best accuracy score for DecisionTreeClassifier :  0.8862927042864995
```

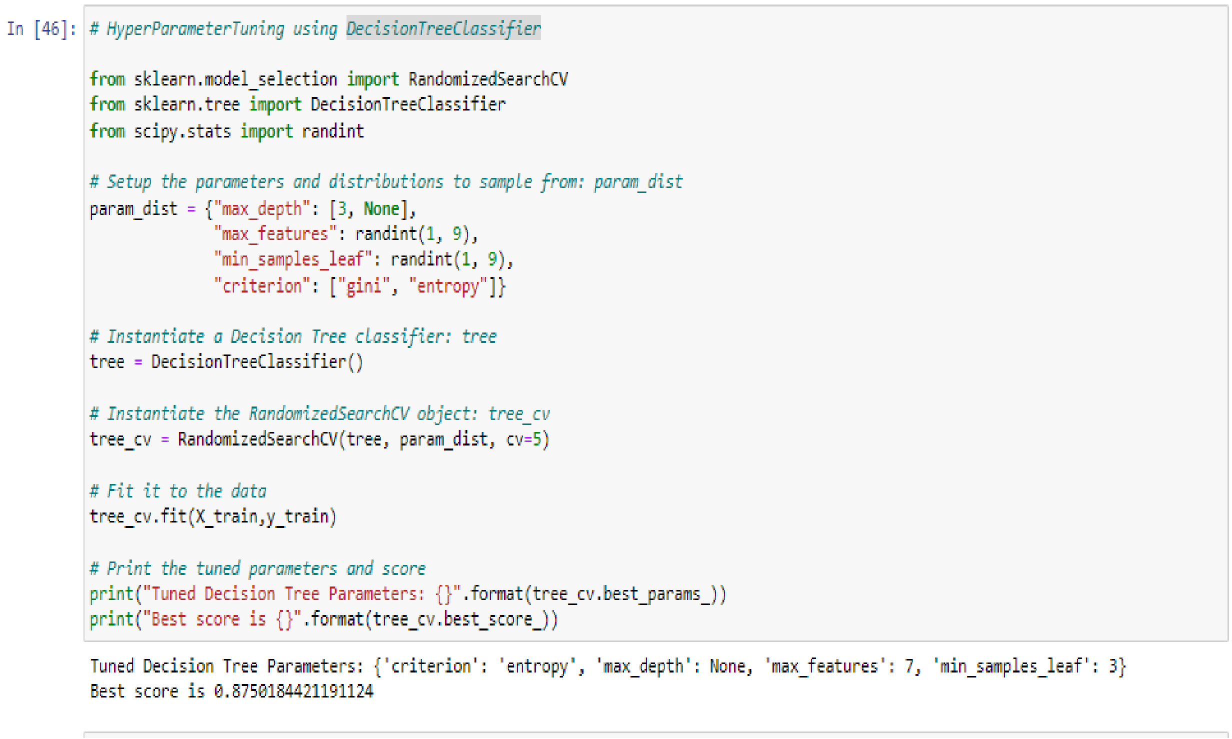


```
In [43]: # finding classification_report for DecisionTreeClassifier

         print(classification_report(y_test, y_pred_dtc))

                       precision    recall  f1-score   support

                    0       0.88      0.89      0.88     53542
                    1       0.89      0.87      0.88     53795

             accuracy                           0.88    107337
            macro avg       0.88      0.88      0.88    107337
         weighted avg       0.88      0.88      0.88    107337
```



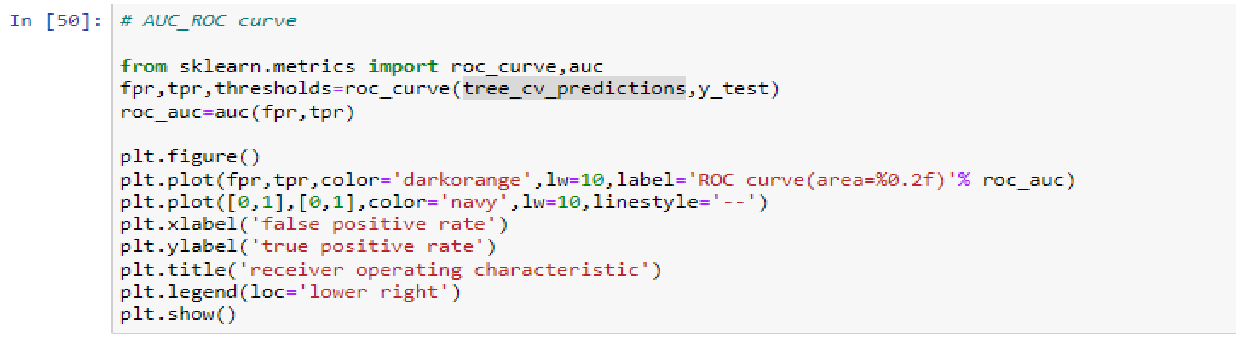
```
In [44]: # finding cross validation score for DecisionTreeClassifier

         cvs = cross_val_score(DecisionTreeClassifier(), X_test, y_test, scoring='accuracy', cv = 10).mean()
         print("cross_val_score for DecisionTreeClassifier  : ",cvs)

         cross_val_score for DecisionTreeClassifier  :  0.8630760955170291
```

HYPERPARAMETER TUNING OF DecisionTreeClassifier :

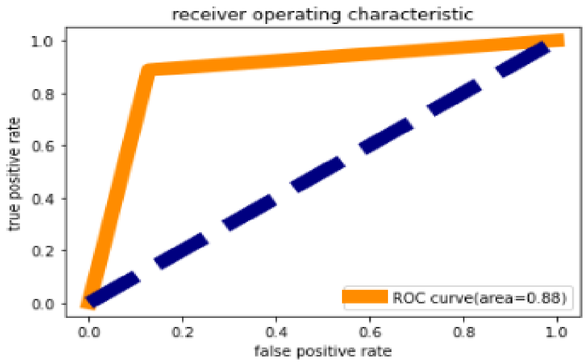


```
In [46]: # HyperParameterTuning using DecisionTreeClassifier

         from sklearn.model_selection import RandomizedSearchCV
         from sklearn.tree import DecisionTreeClassifier
         from scipy.stats import randint

         # Setup the parameters and distributions to sample from: param_dist
         param_dist = {"max_depth": [3, None],
                       "max_features": randint(1, 9),
                       "min_samples_leaf": randint(1, 9),
                       "criterion": ["gini", "entropy"]}

         # Instantiate a Decision Tree classifier: tree
         tree = DecisionTreeClassifier()

         # Instantiate the RandomizedSearchCV object: tree_cv
         tree_cv = RandomizedSearchCV(tree, param_dist, cv=5)

         # Fit it to the data
         tree_cv.fit(X_train,y_train)

         # Print the tuned parameters and score
         print("Tuned Decision Tree Parameters: {}".format(tree_cv.best_params_))
         print("Best score is {}".format(tree_cv.best_score_))

         Tuned Decision Tree Parameters: {'criterion': 'entropy', 'max_depth': None, 'max_features': 7, 'min_samples_leaf': 3}
         Best score is 0.8750184421191124
```

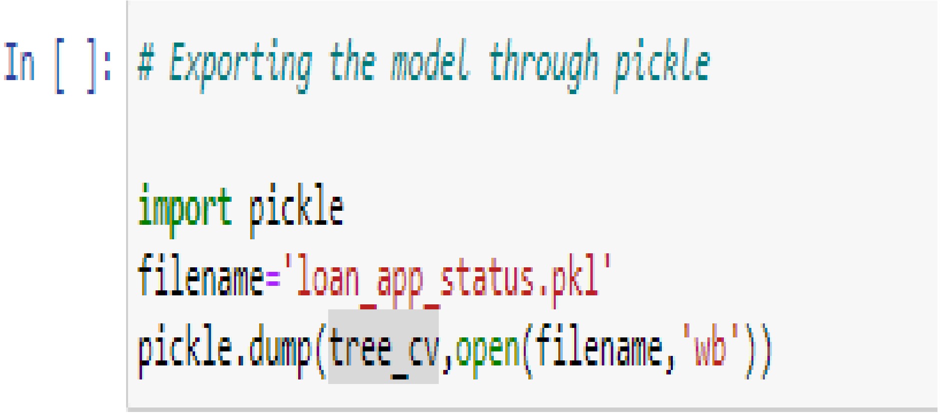
AUC_ROC Curve :



```
In [50]: # AUC_ROC curve

         from sklearn.metrics import roc_curve,auc
         fpr,tpr,thresholds=roc_curve(tree_cv_predictions,y_test)
         roc_auc=auc(fpr,tpr)

         plt.figure()
         plt.plot(fpr,tpr,color='darkorange',lw=10,label='ROC curve(area=%0.2f)'% roc_auc)
         plt.plot([0,1],[0,1],color='navy',lw=10,linestyle='--')
         plt.xlabel('false positive rate')
         plt.ylabel('true positive rate')
         plt.title('receiver operating characteristic')
         plt.legend(loc='lower right')
         plt.show()
```

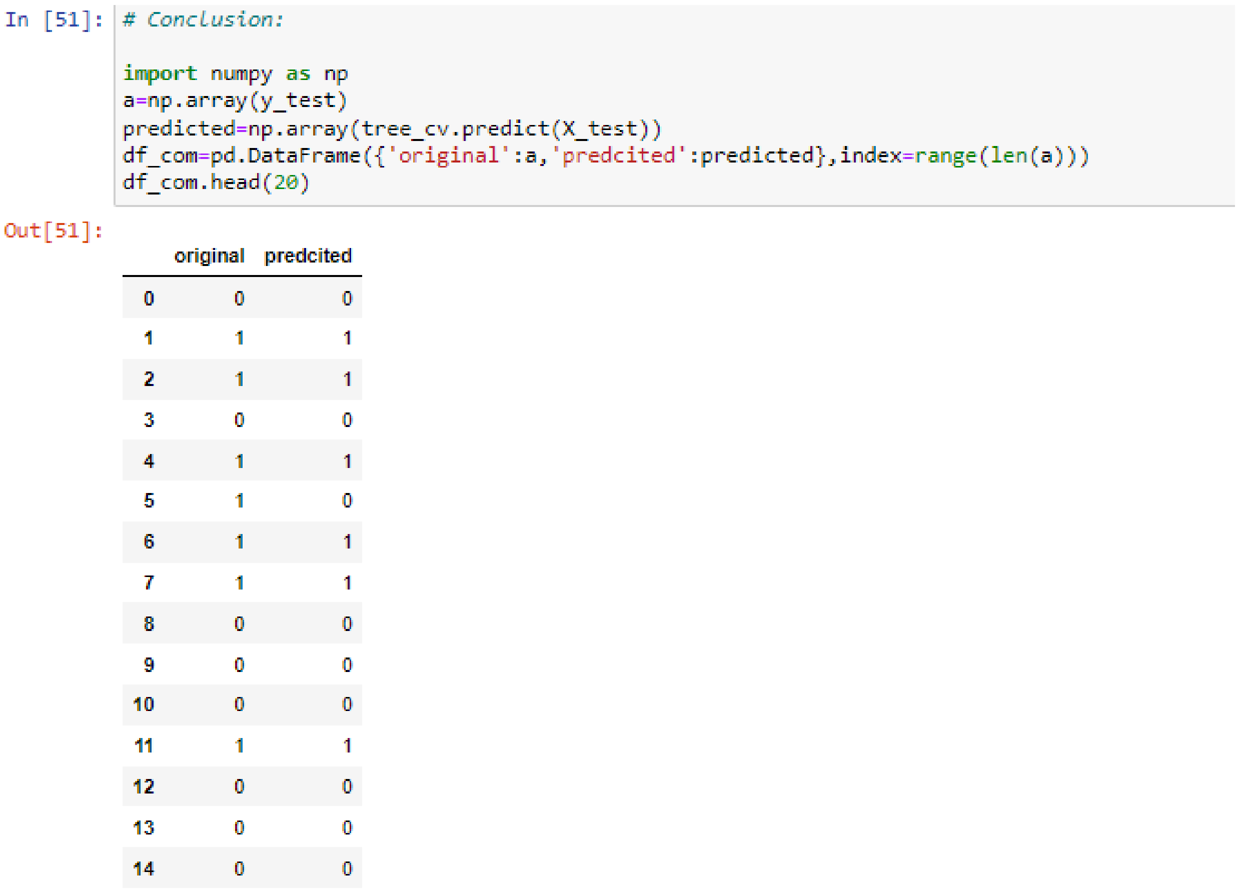


IMPORTING OF MODEL :



```
In [ ]: # Exporting the model through pickle

        import pickle
        filename='loan_app_status.pkl'
        pickle.dump(tree_cv,open(filename,'wb'))
```

Conclusion :



```
In [51]: # Conclusion:

         import numpy as np
         a=np.array(y_test)
         predicted=np.array(tree_cv.predict(X_test))
         df_com=pd.DataFrame({'original':a,'predcited':predicted},index=range(len(a)))
         df_com.head(20)
```

Out[51]:

	original	predcited
0	0	0
1	1	1
2	1	1
3	0	0
4	1	1
5	1	0
6	1	1
7	1	1
8	0	0
9	0	0
10	0	0
11	1	1
12	0	0
13	0	0
14	0	0