FLIP ROBO

Micro Credit Defaulter Project

Submitted by:

Chaganti Sai Mahathi

ACKNOWLEDGMENT

REFERENCES:

[1] H. Kim, H. Cho, and D. Ryu, ‘‘An empirical study on credit card loan delinquency,’’ Econ. Syst., vol. 42, no. 3, pp. 437–449, Sep. 2018.

[2] F. Butaru, Q. Chen, B. Clark, S. Das, A. W. Lo, and A. Siddique, ‘‘Risk and risk management in the credit card industry,’’ J. Banking Finance, vol. 72, pp. 218–239, Nov. 2016.

[3] H. A. Bekhet and S. F. K. Eletter, ‘‘Credit risk assessment model for jordanian commercial banks: Neural scoring approach,’’ Rev. Develop. Finance, vol. 4, no. 1, pp. 20–28, Jan. 2014.

[4] M. Leo, S. Sharma, and K. Maddulety, ‘‘Machine learning in banking risk management: A literature review,’’ Risks, vol. 7, no. 1, p. 29, Mar. 2019.

[5] E. W. T. Ngai, Y. Hu, Y. H. Wong, Y. Chen, and X. Sun, ‘‘The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature,’’ Decis. Support Syst., vol. 50, no. 3, pp. 559–569, Feb. 2011.

[6] S. Tian and Y. Yu, ‘‘Financial ratios and bankruptcy predictions: An international evidence,’’ Int. Rev. Econ. Finance, vol. 51, pp. 510–526, Sep. 2017.

[7] G. Kou, X. Chao, Y. Peng, F. E. Alsaadi, and E. Herrera-Viedma, ‘‘Machine learning methods for systemic risk analysis in financial sectors,’’ Technol. Econ. Develop. Economy, vol. 25, no. 5, pp. 716–742, May 2019.

[8] F. Barboza, H. Kimura, and E. Altman, ‘‘Machine learning models and bankruptcy prediction,’’ Expert Syst. Appl., vol. 83, pp. 405–417, Oct. 2017.
[9] F. Ciampi, ‘‘Corporate governance characteristics and default prediction modeling for small enterprises. An empirical analysis of italian firms,’’ J. Bus. Res., vol. 68, no. 5, pp. 1012–1025, May 2015.

[10] M. TkáčÂ and R. Verner, ‘‘Artificial neural networks in business: Two decades of research,’’ Appl. Soft Comput., vol. 38, pp. 788–804, Jan. 2016.

[11] S. Hamori, M. Kawai, T. Kume, Y. Murakami, and C. Watanabe, ‘‘Ensemble learning or deep learning? Application to default risk analysis,’’ J. Risk Financial Manage., vol. 11, no. 1, p. 12, March.

[12] https://www.researchgate.net/publication/344914401_An_Investigation_of_Credit_Card_Default_Prediction_in_the_Imbalanced_Datasets

[13] An Investigation of Credit Card Default Prediction
https://nova.newcastle.edu.au

[14] https://towardsdatascience.com//

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.

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 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).

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.

- Objective: Now a days the prediction of defaulting the borrower in future is a challenging task for credit card companies . Therefore the main objective is to develop prediction models for defaulting the borrower in the future by taking advantage of the available technological advancement
- So here we will build a model using classification technique 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 issuance of loan. 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.

1)Data Collection and Processing:

Here in this once we load the data we will notice that it has 209395 rows and 37 columns.

The 37 features that can be able to found out the defaulter are:

label-Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}

msisdn - mobile number of user

aon-age on cellular network in days

daily_decr30-Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)

daily_decr90-Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)

rental30-Average main account balance over last 30 days Unsure of given definition

rental90-Average main account balance over last 90 days Unsure of given definition

last_rech_date_ma-Number of days till last recharge of main account

last_rech_date_da-Number of days till last recharge of data account

last_rech_amt_ma-Amount of last recharge of main account (in Indonesian Rupiah)

cnt_ma_rech30-Number of times main account got recharged in last 30 days

fr_ma_rech30-Frequency of main account recharged in last 30 days Unsure of given definition

sumamnt_ma_rech30-Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)

medianamnt_ma_rech30-Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)
medianmarechprebal30-Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)
cnt_ma_rech90-Number of times main account got recharged in last 90 days

fr_ma_rech90-Frequency of main account recharged in last 90 days Unsure of given definition

sumamnt_ma_rech90-Total amount of recharge in main account over last 90 days (in Indonasian Rupiah)

medianamnt_ma_rech90-Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah)
medianmarechprebal90-Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah)
cnt_da_rech30-Number of times data account got recharged in last 30 days

fr_da_rech30-Frequency of data account recharged in last 30 days

cnt_da_rech90-Number of times data account got recharged in last 90 days

fr_da_rech90-Frequency of data account recharged in last 90 days

cnt_loans30-Number of loans taken by user in last 30 days

amnt_loans30-Total amount of loans taken by user in last 30 days

maxamnt_loans30-maximum amount of loan taken by the user in last 30 days There are only two options: 5 & 10 Rs., for which the user needs to pay back 6 & 12 Rs. respectively

medianamnt_loans30-Median of amounts of loan taken by the user in last 30 days

cnt_loans90-Number of loans taken by user in last 90 days

amnt_loans90-Total amount of loans taken by user in last 90 days

maxamnt_loans90-maximum amount of loan taken by the user in last 90 days

medianamnt_loans90-Median of amounts of loan taken by the user in last 90 days

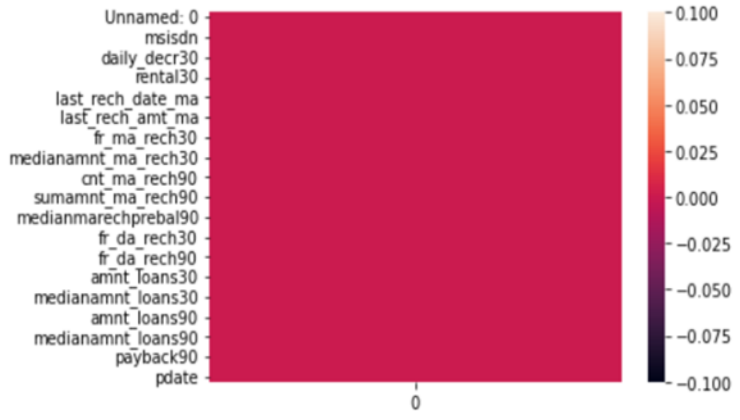
payback30-Average payback time in days over last 30 days

payback90-Average payback time in days over last 90 days p-circle-telecom circle, p-date-date.

Now that we got to know about the features present in the dataset let us check the null count present in the features.

checking the null count using heat map:

```
]: sns.heatmap(df.isnull().sum().to_frame())

]: <AxesSubplot:>
```

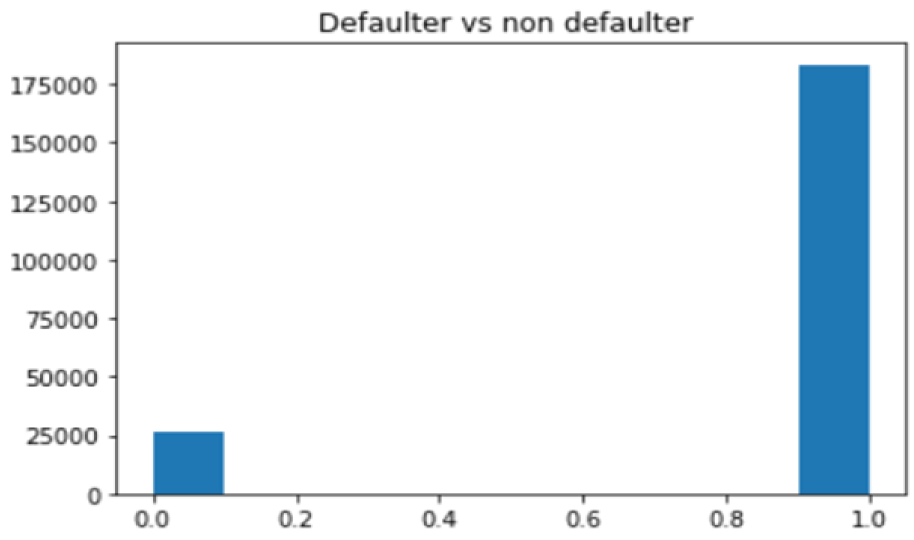


The above graph is a heat map which declares the null count of all the features.

By visualising the graph, we can say that here in this dataset there are no null values occupied.

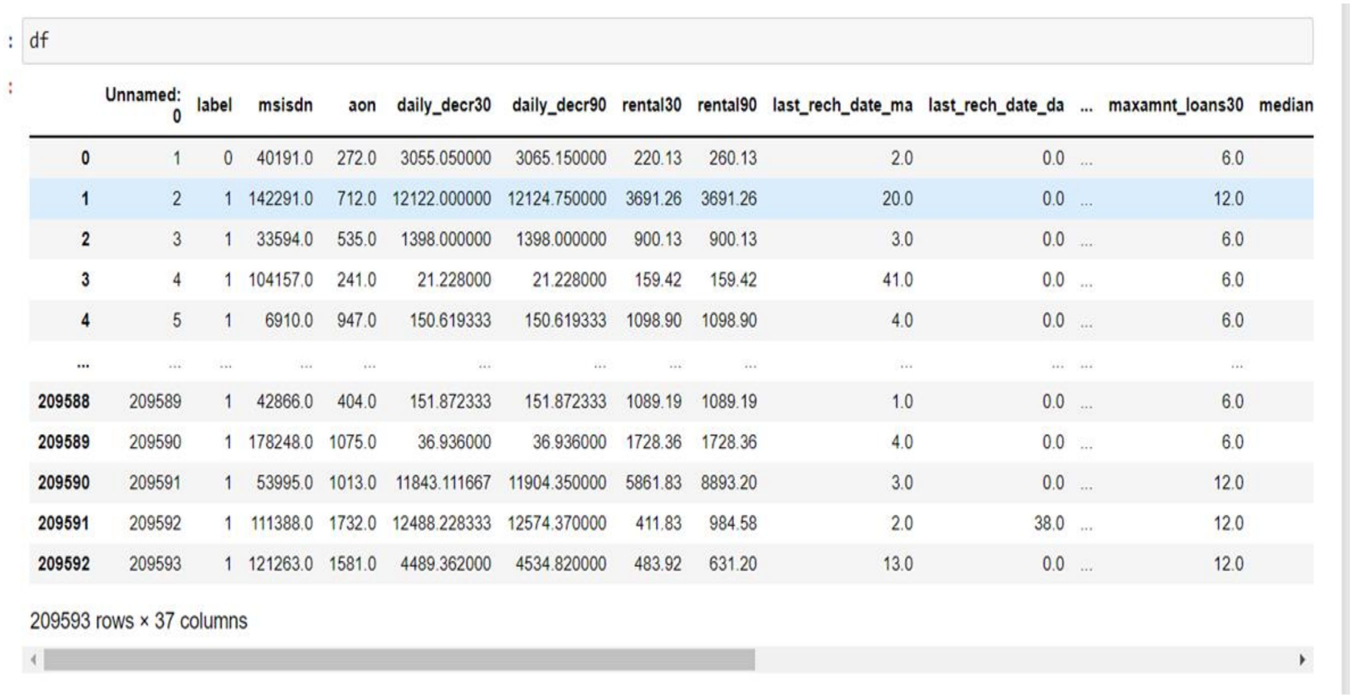
Now, to describe the target variable 'Label' a Histogram is been plotted to analyse the data.

```
df['label'].hist(grid=False)
plt.title('Defaulter vs non defaulter ')
plt.show()
```



The dataset is imbalanced. Label ‘1’ has approximately 87.5% records, while, label ‘0’ has approximately 12.5% records. This imbalance of data can be modified further at model building.

Then, there is a need of encoding the data to analyse the model is most efficient way. Here we use ordinal encoder which encodes all the object data to numeric features.



```
: df
```

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30	median
0	1	0	40191.0	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	...	6.0	
1	2	1	142291.0	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	...	12.0	
2	3	1	33594.0	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	...	6.0	
3	4	1	104157.0	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	...	6.0	
4	5	1	6910.0	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	...	6.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
209588	209589	1	42866.0	404.0	151.872333	151.872333	1089.19	1089.19	1.0	0.0	...	6.0	
209589	209590	1	178248.0	1075.0	36.936000	36.936000	1728.36	1728.36	4.0	0.0	...	6.0	
209590	209591	1	53995.0	1013.0	11843.111667	11904.350000	5861.83	8893.20	3.0	0.0	...	12.0	
209591	209592	1	111388.0	1732.0	12488.228333	12574.370000	411.83	984.58	2.0	38.0	...	12.0	
209592	209593	1	121263.0	1581.0	4489.362000	4534.820000	483.92	631.20	13.0	0.0	...	12.0	

209593 rows × 37 columns

The above image is the new encoded dataset.

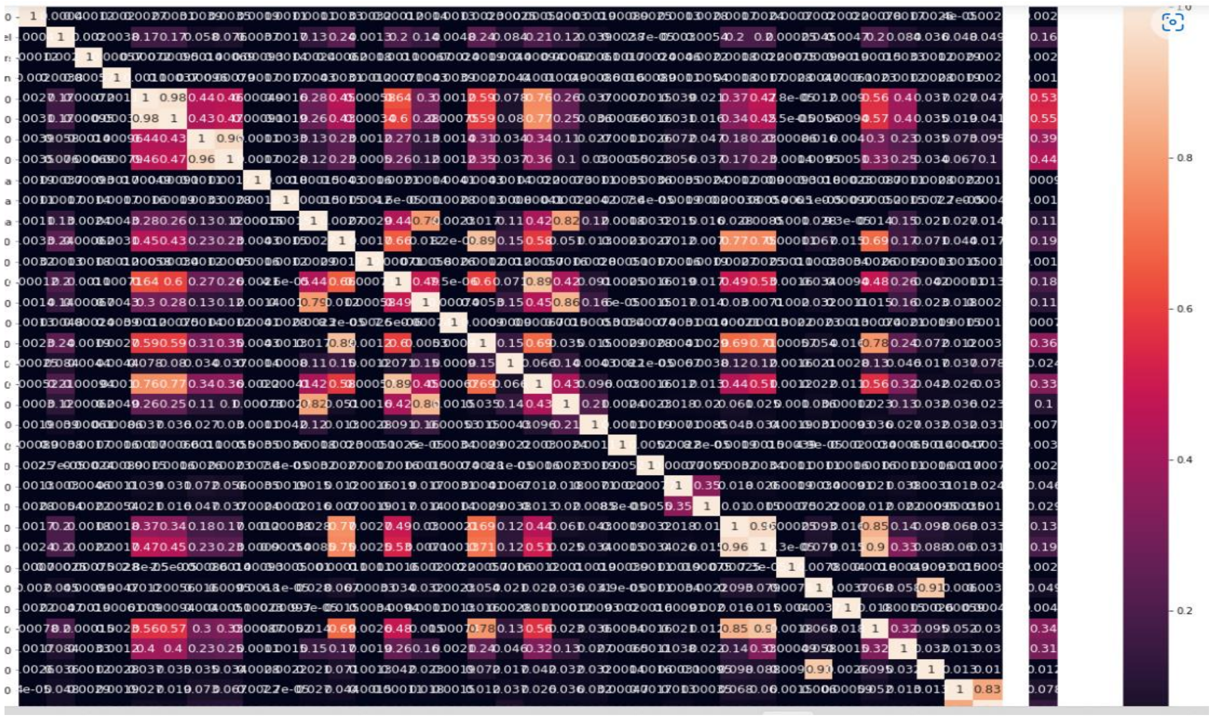
2)EDA:

a**) Correlation**: Correlation explains how one or more variables are related to each other. These variables can be input data features which have been used to forecast our target variable.

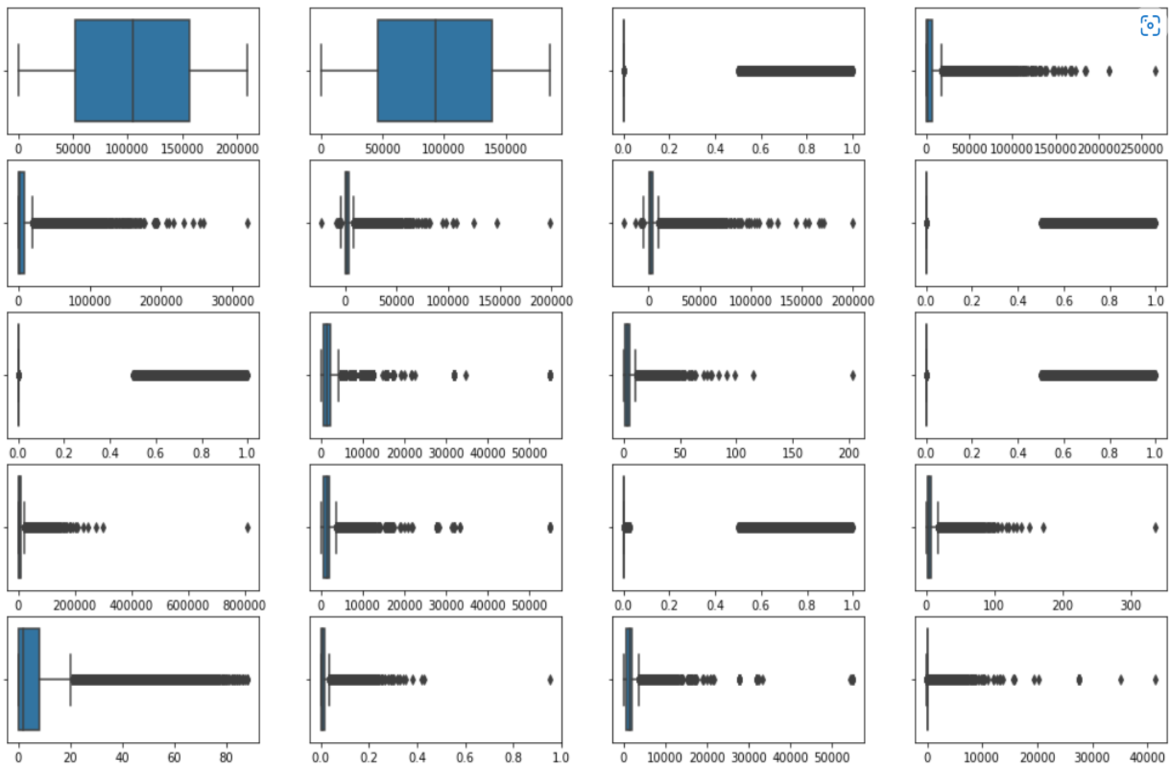
Correlation, statistical technique which determines how one variables moves/changes in relation with the other variable. It gives us the idea about the degree of the relationship of the two variables. It's a bi-variate analysis measure which describes the association between different variables. In most of the business it's useful to express one subject in terms of its relationship with others**.**

Here the correlation can be determined using heat map and other plot graphs to check whether their exists any multi collinearity problem or not.

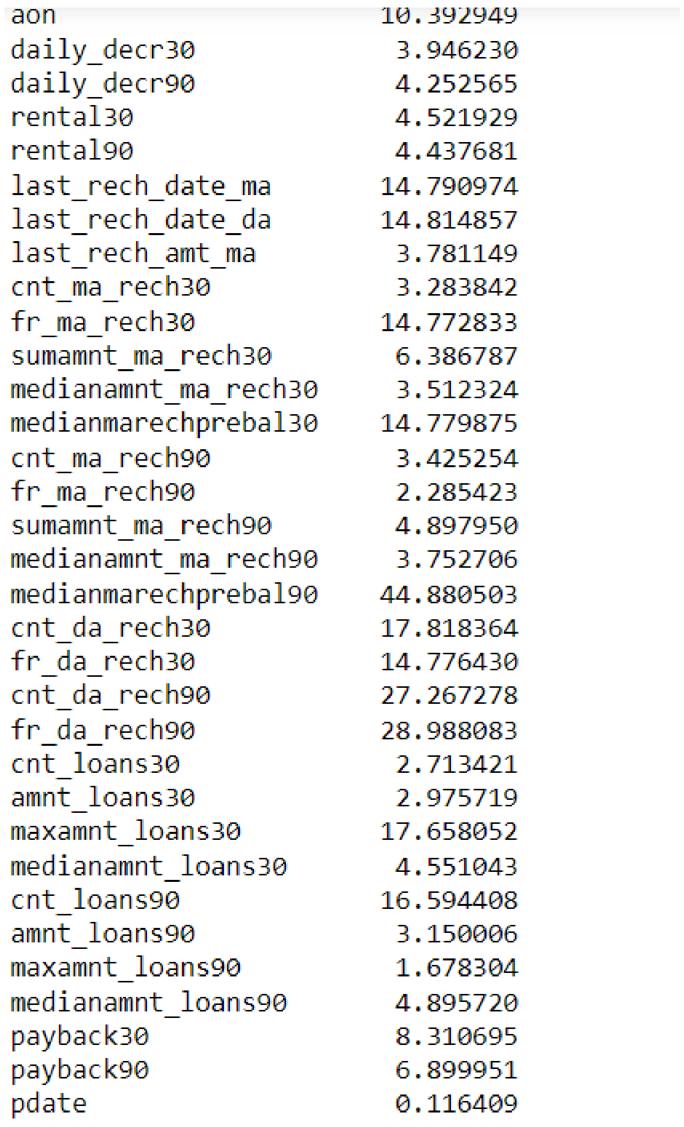
Since there are many features find the correlation of each feature with respect to other is very difficult we move to checking out outliers using boxplot and skewness



Outliers:



Skewness:

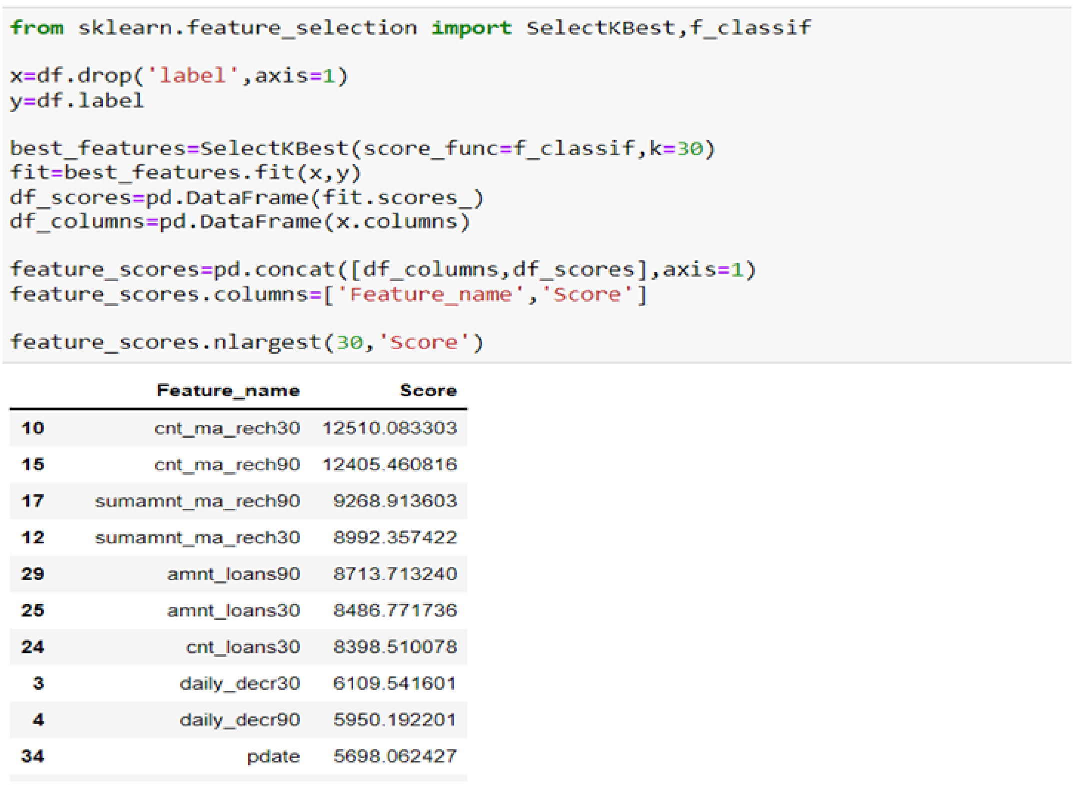


```
aon                     10.392949
daily_decr30             3.946230
daily_decr90             4.252565
rental30                 4.521929
rental90                 4.437681
last_rech_date_ma       14.790974
last_rech_date_da       14.814857
last_rech_amt_ma         3.781149
cnt_ma_rech30            3.283842
fr_ma_rech30            14.772833
sumamnt_ma_rech30        6.386787
medianamnt_ma_rech30     3.512324
medianmarechprebal30    14.779875
cnt_ma_rech90            3.425254
fr_ma_rech90             2.285423
sumamnt_ma_rech90        4.897950
medianamnt_ma_rech90     3.752706
medianmarechprebal90    44.880503
cnt_da_rech30           17.818364
fr_da_rech30            14.776430
cnt_da_rech90           27.267278
fr_da_rech90            28.988083
cnt_loans30              2.713421
amnt_loans30             2.975719
maxamnt_loans30         17.658052
medianamnt_loans30       4.551043
cnt_loans90             16.594408
amnt_loans90             3.150006
maxamnt_loans90          1.678304
medianamnt_loans90       4.895720
payback30                8.310695
payback90                6.899951
pdate                    0.116409
```

By looking into the above graph’s, it seems that the dataset might contains some outliers and skewness also exists in the dataset.

TO avoid these, we consider best features for model building using Feature Selection method. In this method we are considering top 30 features best for modelling and framed as a New Dataset using f_classif functions.

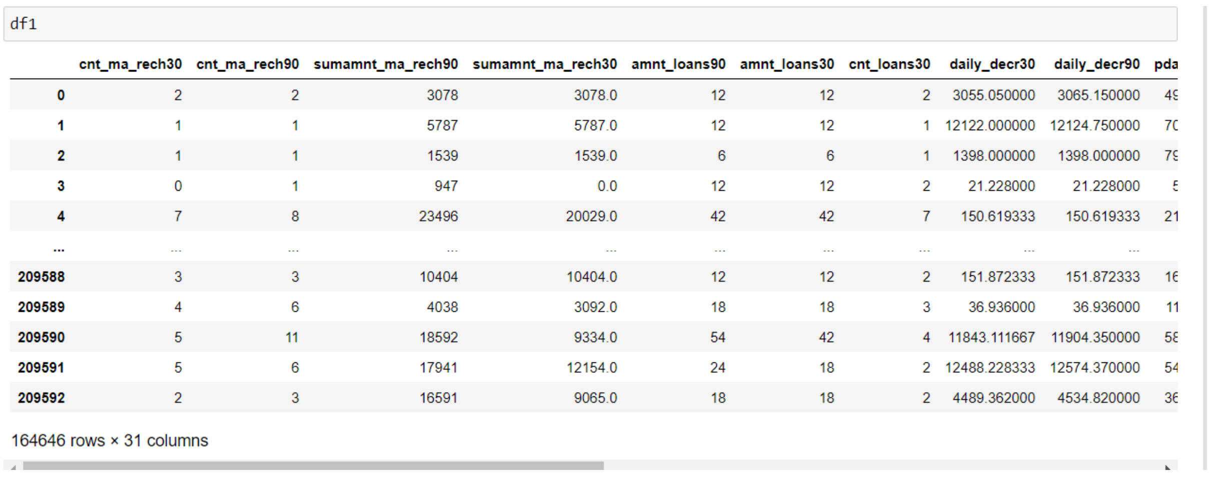
Feature selection method:



```
from sklearn.feature_selection import SelectKBest,f_classif

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

best_features=SelectKBest(score_func=f_classif,k=30)
fit=best_features.fit(x,y)
df_scores=pd.DataFrame(fit.scores_)
df_columns=pd.DataFrame(x.columns)

feature_scores=pd.concat([df_columns,df_scores],axis=1)
feature_scores.columns=['Feature_name','Score']

feature_scores.nlargest(30,'Score')
```

	Feature_name	Score
10	cnt_ma_rech30	12510.083303
15	cnt_ma_rech90	12405.460816
17	sumamnt_ma_rech90	9268.913603
12	sumamnt_ma_rech30	8992.357422
29	amnt_loans90	8713.713240
25	amnt_loans30	8486.771736
24	cnt_loans30	8398.510078
3	daily_decr30	6109.541601
4	daily_decr90	5950.192201
34	pdate	5698.062427

Zscore technique:This technique is used to remove outliers and frames a another dataset with dimension(164646,31)



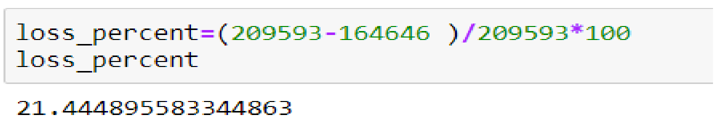
```
df1
```

	cnt_ma_rech30	cnt_ma_rech90	sumamnt_ma_rech90	sumamnt_ma_rech30	amnt_loans90	amnt_loans30	cnt_loans30	daily_decr30	daily_decr90	pda
0	2	2	3078	3078.0	12	12	2	3055.050000	3065.150000	49
1	1	1	5787	5787.0	12	12	1	12122.000000	12124.750000	70
2	1	1	1539	1539.0	6	6	1	1398.000000	1398.000000	79
3	0	1	947	0.0	12	12	2	21.228000	21.228000	5
4	7	8	23496	20029.0	42	42	7	150.619333	150.619333	21
...	...	...	...	...	...	...	...	...	...	
209588	3	3	10404	10404.0	12	12	2	151.872333	151.872333	16
209589	4	6	4038	3092.0	18	18	3	36.936000	36.936000	11
209590	5	11	18592	9334.0	54	42	4	11843.111667	11904.350000	58
209591	5	6	17941	12154.0	24	18	2	12488.228333	12574.370000	54
209592	2	3	16591	9065.0	18	18	2	4489.362000	4534.820000	36

164646 rows × 31 columns

Percentage data loss:

Difference between the data before applying the Zscore and after applying the Zscore method gives us the percentage of loss of data



```
loss_percent=(209593-164646 )/209593*100
loss_percent
```

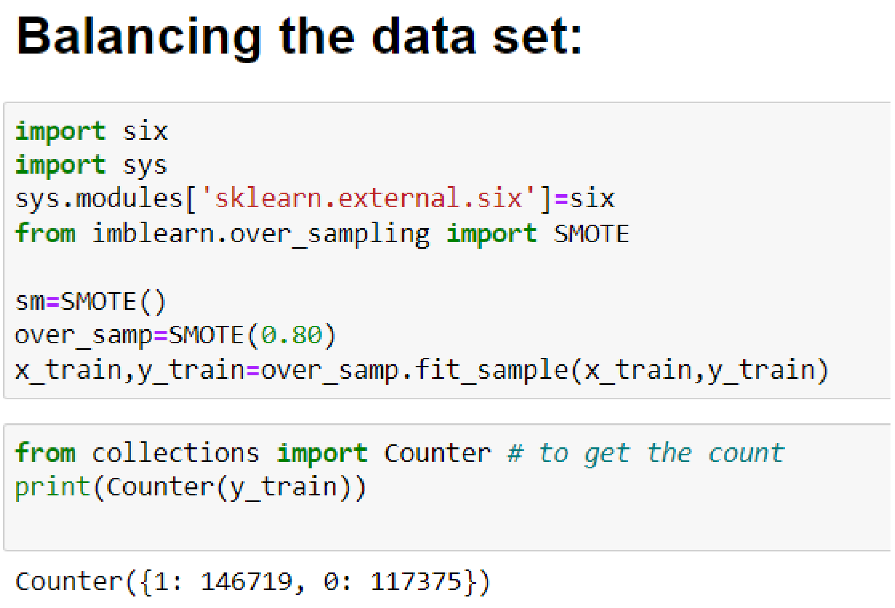
```
21.444895583344863
```

since there is huge data loss we consider the data set before using zscore technique to build the model.

Model Building:

Once splitting the features and target variable into x and y and creating a train test split .

We Balance the data using SMOTE technique:



Balancing the data set:

```
import six
import sys
sys.modules['sklearn.external.six']=six
from imblearn.over_sampling import SMOTE

sm=SMOTE()
over_samp=SMOTE(0.80)
x_train,y_train=over_samp.fit_sample(x_train,y_train)
```

```
from collections import Counter # to get the count
print(Counter(y_train))
```

```
Counter({1: 146719, 0: 117375})
```

Once the dataset is ready to be trained we use different machine learning techniques for the prediction of target variable:

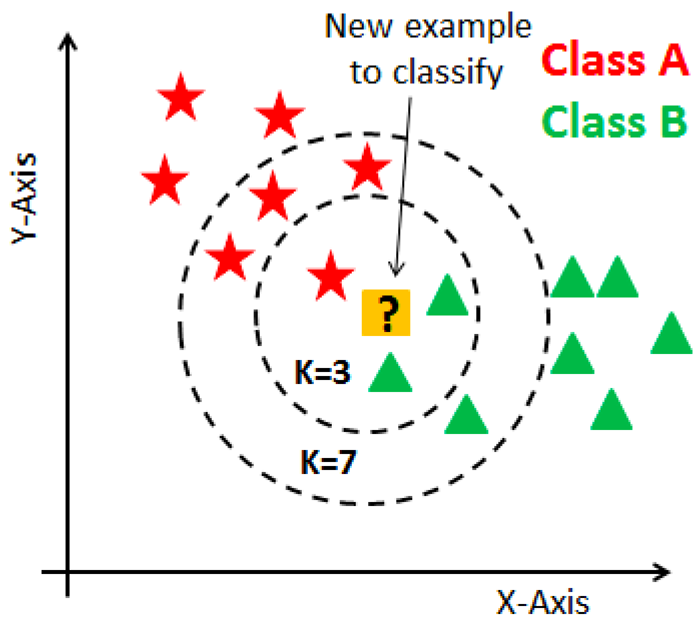
1)**KNN Classifier**:

K nearest neighbors (KNN) is a supervised machine learning algorithm. A supervised machine learning algorithm's goal is to learn a function such that f(X) = Y where X is the input, and Y is the output. KNN can be used both for classification as well as regression. In this article, we will only talk about classification. Although for regression, there is just a minute change.

The properties of KNN is that it is a lazy learning algorithm and a non-parametric method.

Lazy learning means the algorithm takes almost zero time to learn because it only stores the data of the training part (no learning of a function). The stored data will then be used for the evaluation of a new query point.

The non-parametric method refers to a method that does not assume any distribution. Therefore, KNN does not have to find any parameter for the distribution. While in the parametric method, the model finds new parameters, which in turn will be used for the prediction purpose. The only hyperparameter (provided by the user to the model) KNN has is K, which is the number of points that needs to be considered for comparison purpose.



In the training phase, the model will store the data points. In the testing phase, the distance from the query point to the points from the training phase is calculated to classify each point in the test dataset. Various distances can be calculated, but the most popular one is the Euclidean distance (for smaller dimension data).

Euclidean distance between a query point (q) and a training data point (p) is defined as

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2 + (q_2-p_2)^2 + \cdots + (q_n-p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}.$$

Euclidean distance between point p and q (n-dimensional points). Source

Other distance measures such as Manhattan, Hamming, and Chebyshev distance can also be used based on the data, which is out of the scope of this article.

Let’s learn it with an example:

We have 500 N-dimensional points, with 300 being class 0 and 200 being class 1.

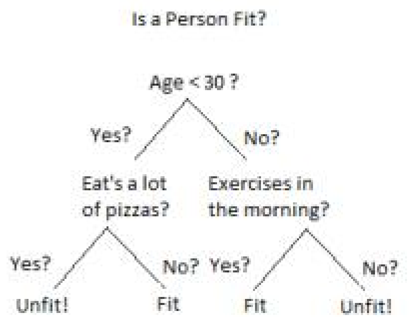
The procedure for calculating the class of query point is:

1. The distance of all the 500 points is calculated from the query point.
2. Based on the value of K, K nearest neighbors are used for the comparison purpose.
3. Let’s say K=7, 4 out of 7 points are of class 0, and 3 are of class 1. Then based on the majority, the query point p is assigned as class 0.

```
Accuracy:  0.8086309310813712
confusion_matrix:
 [[ 3752  1455]
 [ 6567 30145]]
classification report:                precision    recall  f1-score   support

           0       0.36      0.72      0.48      5207
           1       0.95      0.82      0.88     36712

    accuracy                           0.81     41919
   macro avg       0.66      0.77      0.68     41919
weighted avg       0.88      0.81      0.83     41919
```

2)**Decision Tree Classifier**:

Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the



data is split.

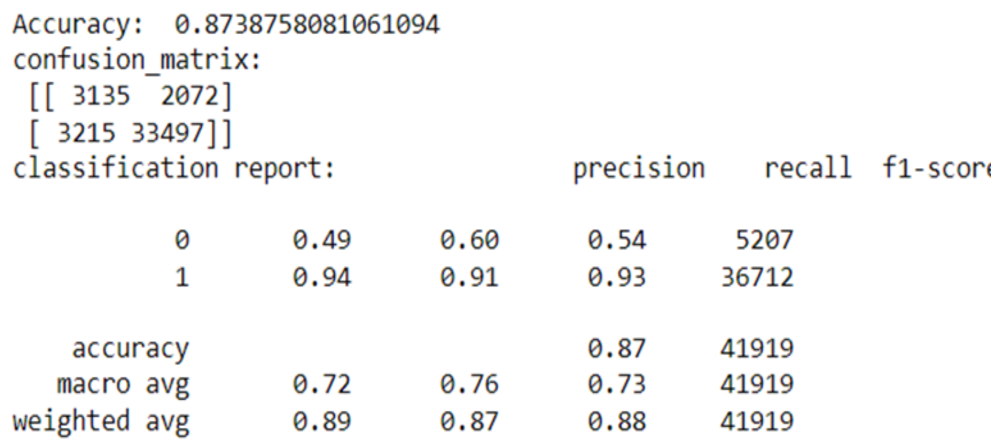
An example of a decision tree can be explained using above binary tree. Let's say you want to predict whether a person is fit given their information like age, eating habit, and physical activity, etc. The decision nodes here are questions like 'What's the age?', 'Does he exercise?', 'Does he eat a lot of pizzas'? And the leaves, which are outcomes like either 'fit', or 'unfit'. In this case this was a binary classification problem (a yes no type problem). There are two main types of Decision Trees:

1. **Classification trees** (Yes/No types)

What we've seen above is an example of classification tree, where the outcome was a variable like 'fit' or 'unfit'. Here the decision variable is **Categorical**.

2. **Regression trees** (Continuous data types)

Here the decision or the outcome variable is **Continuous**, e.g. a number like 123. **Working** Now that we know what a Decision Tree is, we'll see how it works internally. There are many algorithms out there which construct Decision Trees, but one of the best is called as **ID3 Algorithm**. ID3 Stands for **Iterative Dichotomiser 3**. Before discussing the ID3 algorithm, we'll go through few definitions.



```
Accuracy:  0.8738758081061094
confusion_matrix:
 [[ 3135  2072]
 [ 3215 33497]]
classification report:               precision    recall  f1-scor

           0       0.49      0.60      0.54      5207
           1       0.94      0.91      0.93     36712

    accuracy                           0.87     41919
   macro avg       0.72      0.76      0.73     41919
weighted avg       0.89      0.87      0.88     41919
```

3)**Random Forest Classifier**:

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

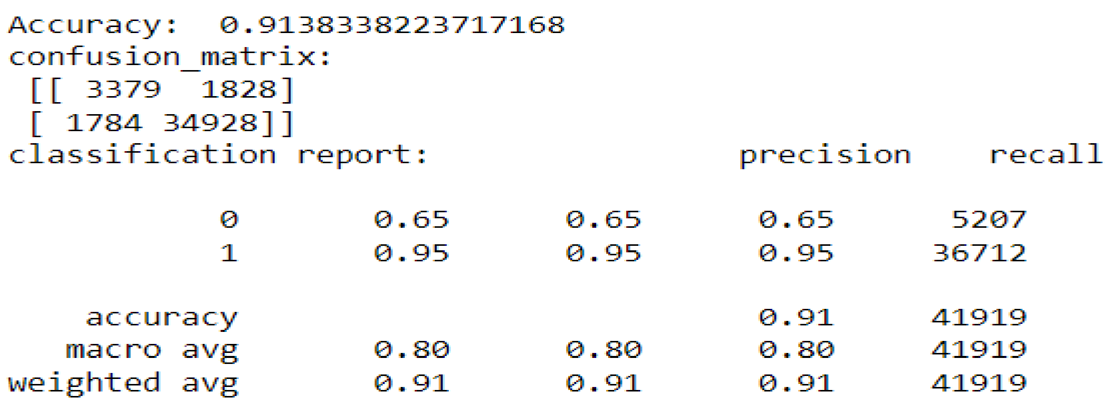
The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

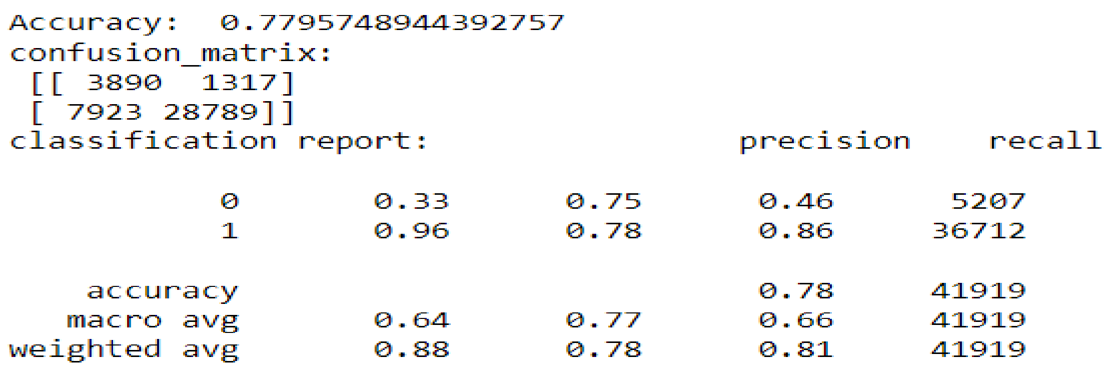
Step-4: Repeat Step 1 & 2.



```
Accuracy:  0.9138338223717168
confusion_matrix:
 [[ 3379  1828]
 [ 1784 34928]]
classification report:                    precision    recall

           0       0.65      0.65      0.65      5207
           1       0.95      0.95      0.95     36712

    accuracy                           0.91     41919
   macro avg       0.80      0.80      0.80     41919
weighted avg       0.91      0.91      0.91     41919
```

4)**Logistic Regression**:

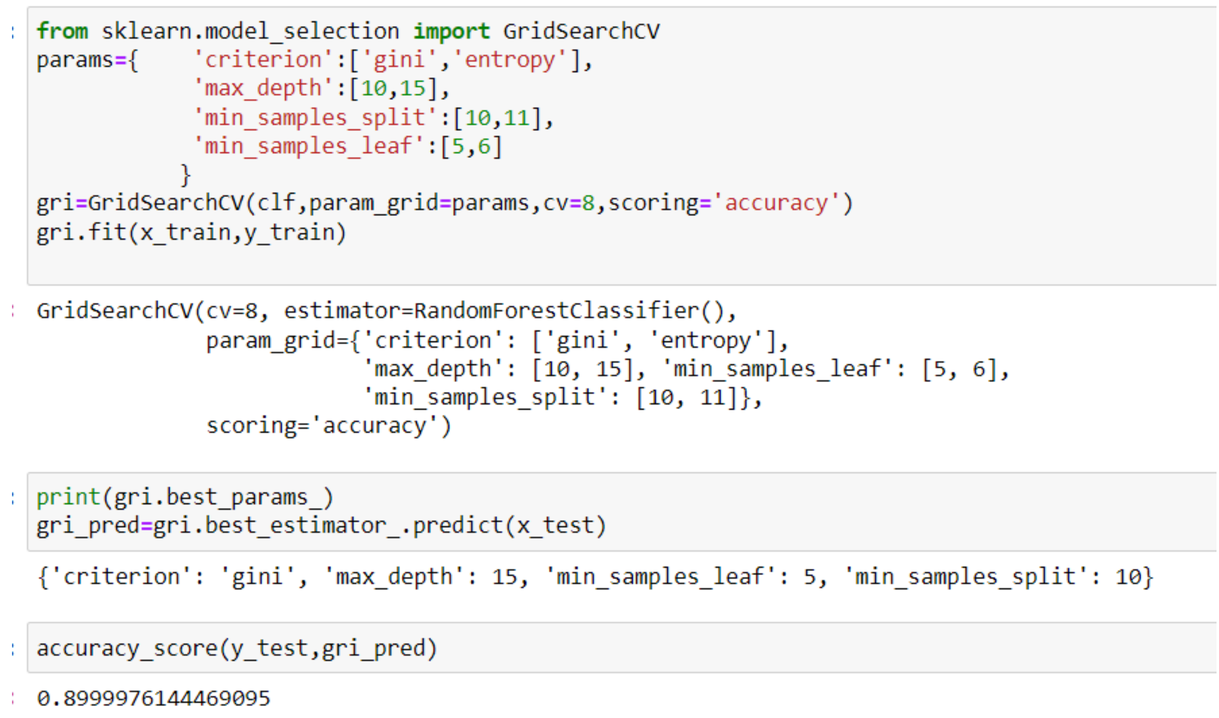
Logistic regression is an example of supervised learning. It is used to calculate or predict the probability of a binary (yes/no) event occurring.



```
Accuracy:  0.7795748944392757
confusion_matrix:
 [[ 3890  1317]
 [ 7923 28789]]
classification report:                    precision    recall

           0       0.33      0.75      0.46      5207
           1       0.96      0.78      0.86     36712

    accuracy                           0.78     41919
   macro avg       0.64      0.77      0.66     41919
weighted avg       0.88      0.78      0.81     41919
```

- **Hyperparameter tunning**:

  "Random forest Classifier" with accuracy score of 91.38% is considered to be the best model compared to the other models so this model is saved and can be used for future prediction.



```
from sklearn.model_selection import GridSearchCV
params={    'criterion':['gini','entropy'],
            'max_depth':[10,15],
            'min_samples_split':[10,11],
            'min_samples_leaf':[5,6]
           }
gri=GridSearchCV(clf,param_grid=params,cv=8,scoring='accuracy')
gri.fit(x_train,y_train)
```

```
GridSearchCV(cv=8, estimator=RandomForestClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [10, 15], 'min_samples_leaf': [5, 6],
                         'min_samples_split': [10, 11]},
             scoring='accuracy')
```

```
print(gri.best_params_)
gri_pred=gri.best_estimator_.predict(x_test)
```

```
{'criterion': 'gini', 'max_depth': 15, 'min_samples_leaf': 5, 'min_samples_split': 10}
```

```
accuracy_score(y_test,gri_pred)
```

```
0.8999976144469095
```

Conclusion:

After Hyper parameter tuning the accuracy score is predicted as 89.99% which is considered as good model, now that the model got trained we can predict the test data and any other future data using this trained model.

```python
import numpy as np
a=np.array(y_test)
predicted=np.array(gri.best_estimator_.predict(x_test))
df_conclusion=pd.DataFrame({'Original':a,'Predicted':predicted},index=range(len(a)))
df_conclusion
```

	Original	Predicted
0	1	1
1	1	1
2	1	1
3	1	1
4	1	0
...	...	...
41914	1	1
41915	1	1
41916	1	1
41917	1	1
41918	1	1

41919 rows × 2 columns