

In loan application status prediction, we are trying to analyze whether the loan for an applicant is approved or not. Loan approval is a very important process for banking organizations. The system approved or reject the loan applications. Recovery of loans is a major contributing parameter in the financial statements of a bank.

It is very difficult to predict the possibility of payment of loan by the customer. In recent years many researchers worked on loan approval prediction systems. Machine Learning (ML)techniques are very useful in predicting outcomes for large amount of data.

In this project, we are provided a dataset which has the details of the applicants who have applied for a loan policy. The dataset includes details like credit history, loan amount, their income, dependents etc.

In this example, we will be working with details of the applicants to demonstrate how we can create a predictive model that predicts if an applicant is approved to avail loan or not.

**INTRODUCTION**

Bank plays a vital role in market economy. The success or failure of organization largely depends on the industry’s ability to evaluate credit risk. Before giving the credit loan to borrowers, bank decides whether the borrower is bad (defaulter) or good (non defaulter).The prediction of borrower status i.e. in future borrower will be defaulter or non defaulter is a challenging task for any organization or bank.

Henceforth prediction of loan application status of an applicant is most important for an organization to run successfully and at the same time provide service to its customers.

1. **PROBLEM DEFINITION**

This dataset includes details of applicants who have applied for loan. The dataset includes details like credit history, loan amount, their income, dependents etc.

**Independent Variables:**

- Loan\_ID - ID that is assigned to the loan application

- Gender - Gender tells the gender of an applicant

- Married - Provides the status of whether the applicant is married or not

-Dependents - The number of dependents/ co-applicants , for eg: spouse, children or parents, are also assessed before sanctioning the loan

- Education - Info on education of the loan applicant

- Self\_Employed – Status on whether the applicant is self-employed or not

- ApplicantIncome – Income amount of an Applicant

- CoapplicantIncome – Income amount of Co-applicant

- Loan\_Amount - Amount of loan applied by an applicant

- Loan\_Amount\_Term – Term / duration in which the loan must be cleared

- Credit History – Info on repayment history of previous or existing loan

- Property\_Area – The area of which the property belongs for which the loan is applied for

**Dependent Variable (Target Variable):**

- Loan\_Status – Info on whether loan is approved or not . It has two variables Yes or No

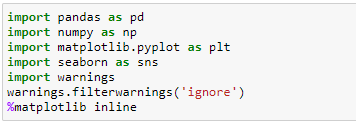
“ Yes “ meaning the loan applied by an applicant is approved, “ No ” meaning the loan applied by an applicant is not approved or denied

We have to build a model that can predict whether the loan of the applicant will be approved or not on the basis of the details provided in the dataset

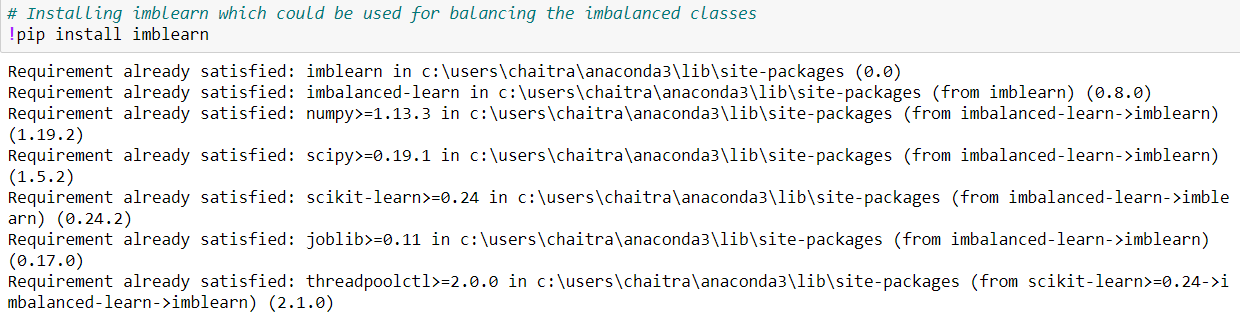
This current dataset has n=614 samples. It means the data set consists of 614 applicant details and 13 features. It is not stated if this data is from multiple banks or just one bank

Importing libraries and dataset: -

Here we are importing Pandas, NumPy. We are also importing Matplotlib and Seaborn for data visualization

****

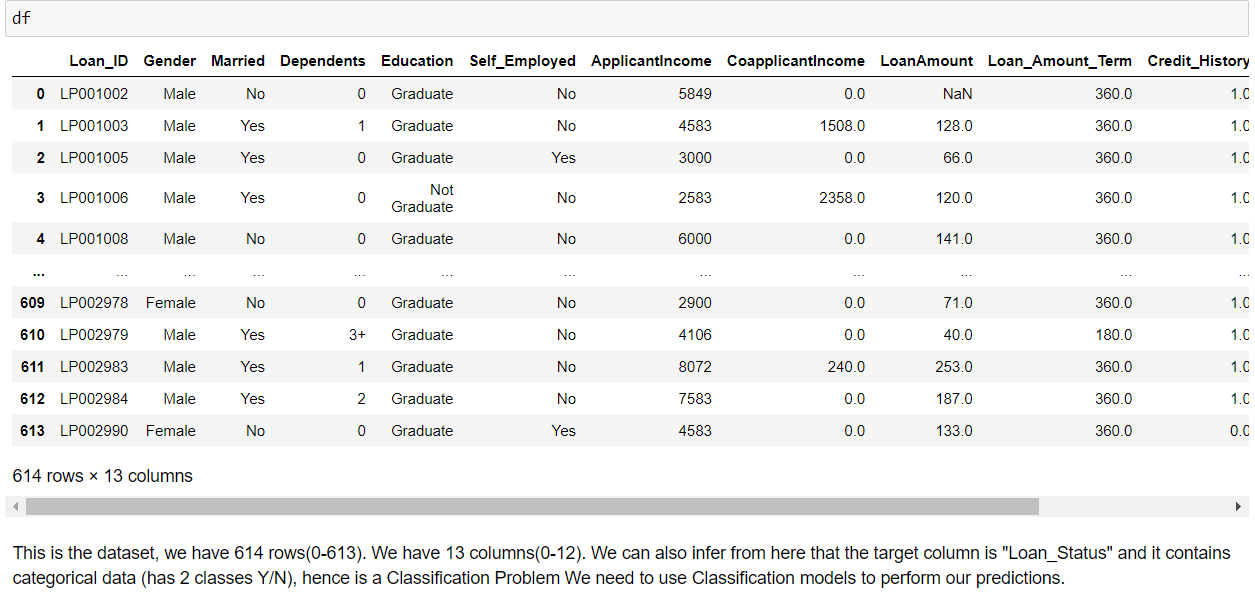
Let’s import imblearn which is used for balancing the classes. In this project we have 2 classes in the target label i.e., yes or no

****

Now let’s import the dataset. Here the dataset is downloaded from github.



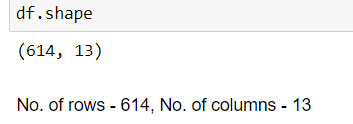
Now let’s check if the data frame is imported successfully



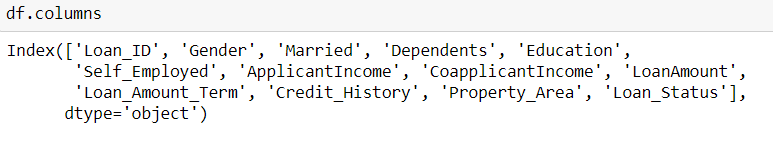
Df is imported and we can see that it contains 614 rows and 13 columns within it. Our target column is Loan\_Status and it has 2 classes “yes” or “no” .Hence a classification problem. We need to use Classification models or algorithms to perform our predictions in here.

1. **Data Analysis**

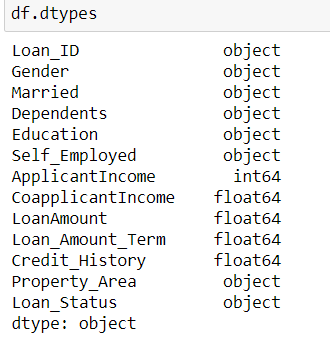
Let’s check for its shape using df.shape()



Gives info on number of rows and columns present



Gives info on the column headings of our dataset



The independent variables "CoapplicantIncome", "LoanAmount", "Loan\_Amount\_Term" and "Credit\_History" contains float values and its datatype is float.

The independent variable "Loan\_ID" has alphanumeric values and is identified as object datatype. The independent variable "Dependents" numeric plus arithematic symbol as values and is identified as object.

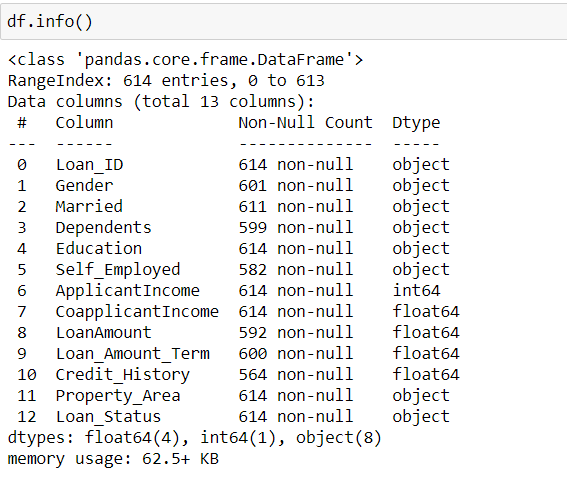
The independent variables , "Gender", "Married", "Education", "Self\_Employed", "Property\_Area" and the dependent variable or the target column "Loan\_Status" has string values and is of string datatype.

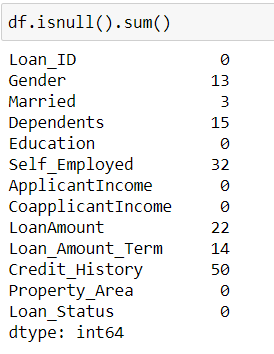
Now let’s try to find more info on our dataset

Here we can use the function df.info()

From df.info() we can infer that the columns "Loan\_ID", "Education", "ApplicantIncome", "CoapplicantIncome", "Property\_Area" and "Loan\_Status" contains no null values present in them.

All the rest columns in our dataset contains null values, so we need to treat them

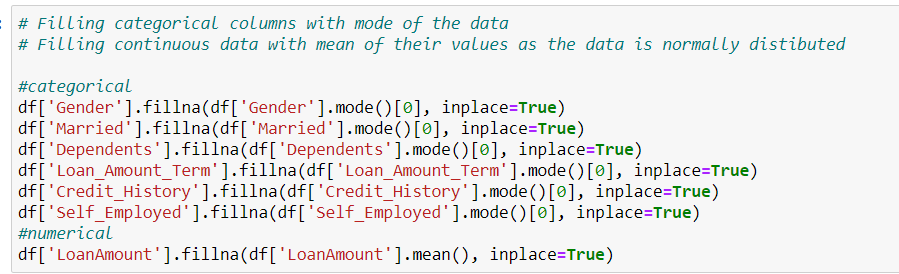




From here we can infer that we have missing values present in few columns. The columns that have missing values present in them are "Gender", "Married", "Dependents", "Self\_Employed ", "LoanAmount", "Loan\_Amount\_Term", "Credit\_History". The solution to this is that we need to fill some values in them.

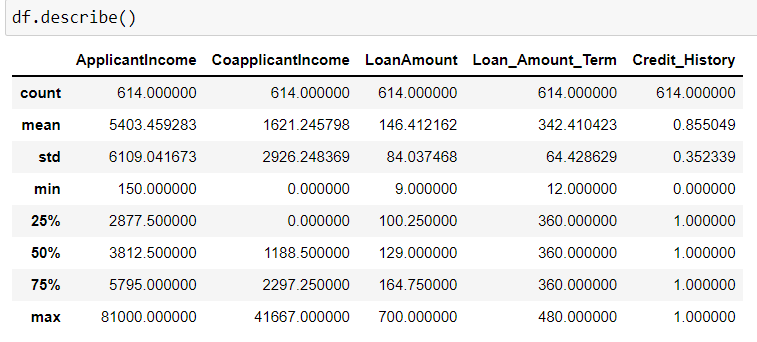
**Missing value treatment**

Missing value can be treated by filling them with the values. Categorical columns to be filled with mode of the values in that column and for continuous data which are normally distributed we can use mean of the values and for the continuous data which are not normally distributed we need to go for median of the values.



1. **Exploratory Data Analysis**

Let’s check for df.describe()

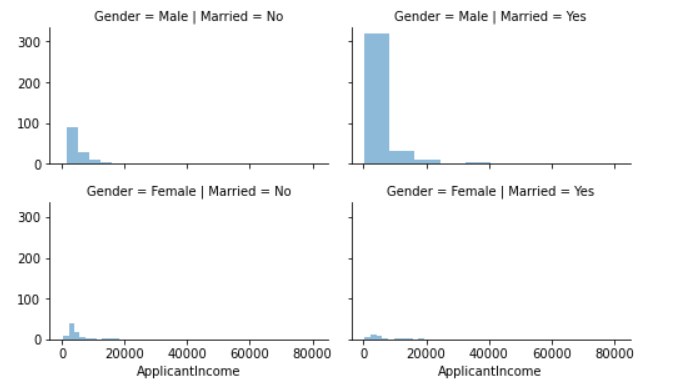


From df.describe() we can see the details about various statistical data like Count,Mean,Standard Deviation,Max Value,Min Value for our dataset. So from min and max values, we basically get the range. There seems to be some outliers for the Applicant Income , Coapplicant income and Loan Amount,

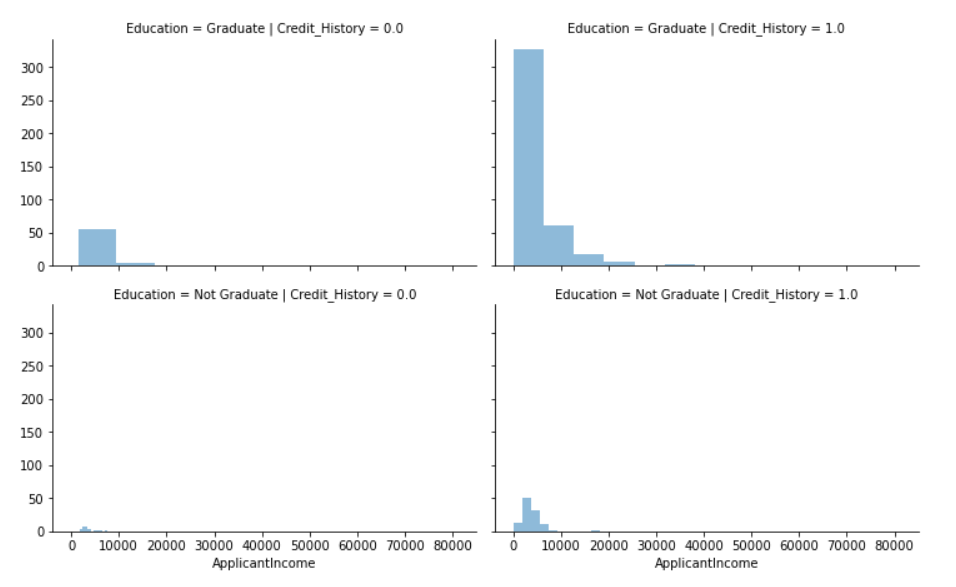
We also see that about 85% applicants have a credit\_history. Because the mean of Credit\_History field is 0.85 and it has either (1 for having a credit history or 0 for not). We can min value of "CoapplicantIncome" and "Credit\_History" is zero.

**Data Analysis and Visualizatiton**

Males generally have the highest income. Males who are married have greater income that unmarried male. Married females have even lesser income in comparision with unmarried females. Analysis for very important features can be visually seen below.

****

Credit history is very important , let’s analyze the education, applicant income and applicant’s credit history

****

Graduates with good credit history depicts a good income. Also, not a graduate and have a good credit history can be traced to having a better income than the ones with no degree.

**4 . Pre-processing**

For the heatmap to be plotted , we need all the values in numeric data type and aswell as for our model to identify the features all string and unidentified datatypes needs to be converted to identifiable datatype

For this we use label encoder and the process is called as label encoding

Using label encoder we are converting all the categorical string type values into python identifiable datatypes

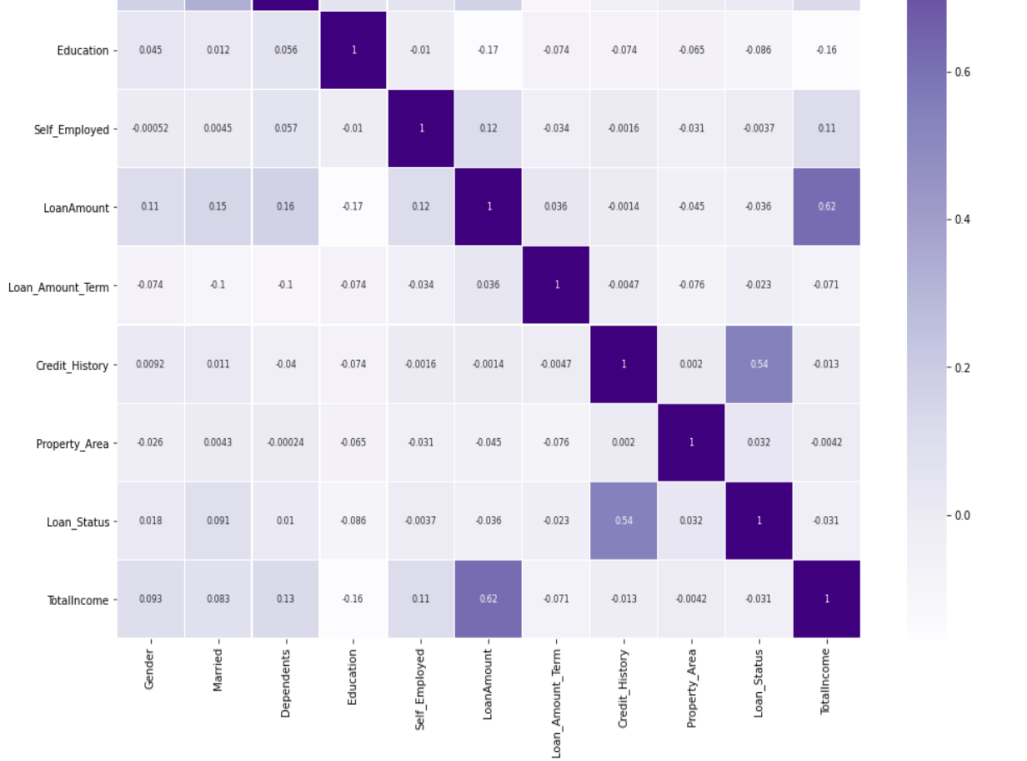


Now all the columns of our dataset are identifiable and are in either float or int datatype.

**Correlation**: -

Correlation coefficients are used to measure the strength of the relationship between two variables. This measures the strength and direction of a linear relationship between two variables. Values always range between -1 (strong negative relationship) and +1 (strong positive relationship).

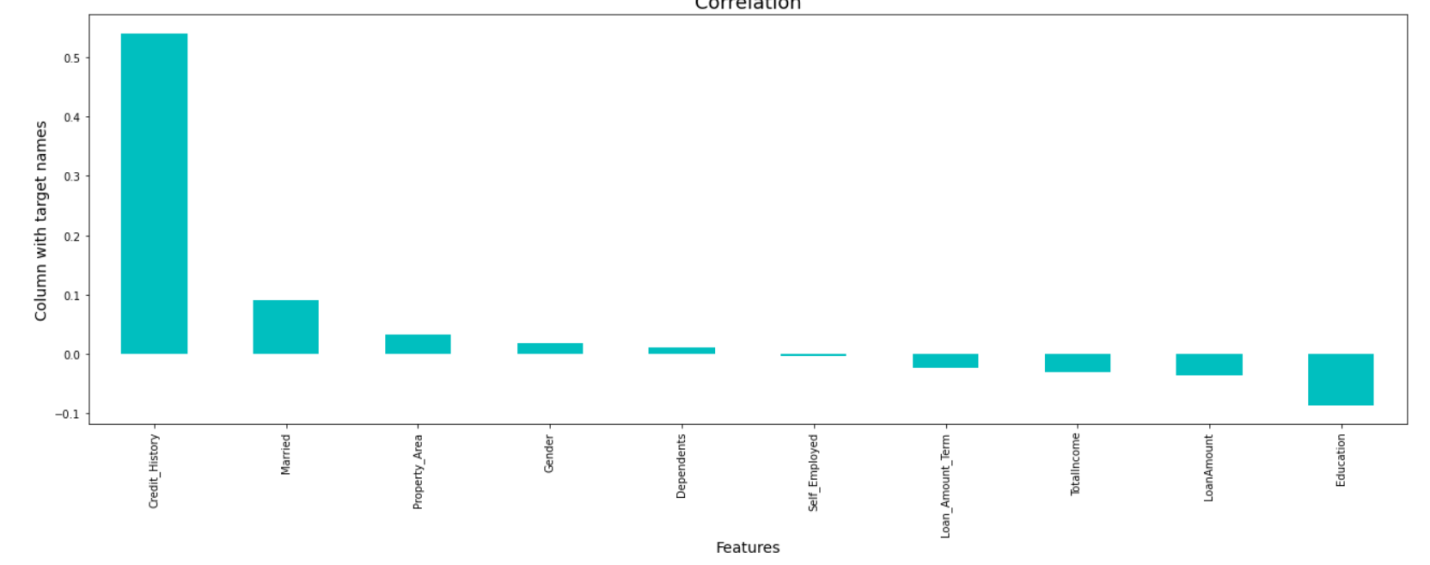
We shall check the correlation of the variables. From the heatmap we can make some observations as such.



From this heatmap we have got the correlation values and we can see that the column "Credit\_History" has high positive correlation with the target column compared to all other columns. So if the Credit\_History is high, the chance of getting loan approved is high

The column "Self\_Employed" has weak negative correlation value(-0.0037) with the target variable(not a very important factor because the correlation value is close to zero)

The other columns with negative correlation with the target variable is "Education", "Loan\_Amount", "Total\_Income" and "Loan\_Amount\_Term".



Here we can infer that the columns "Credit\_History", "Married", "Property\_Area", "Gender" and "Dependents" are positively correlated with our target column and the variables "Self\_Employed", "Loan\_Amount\_Term", "TotalIncome", "LoanAmount" and "Education" are negatively correlated

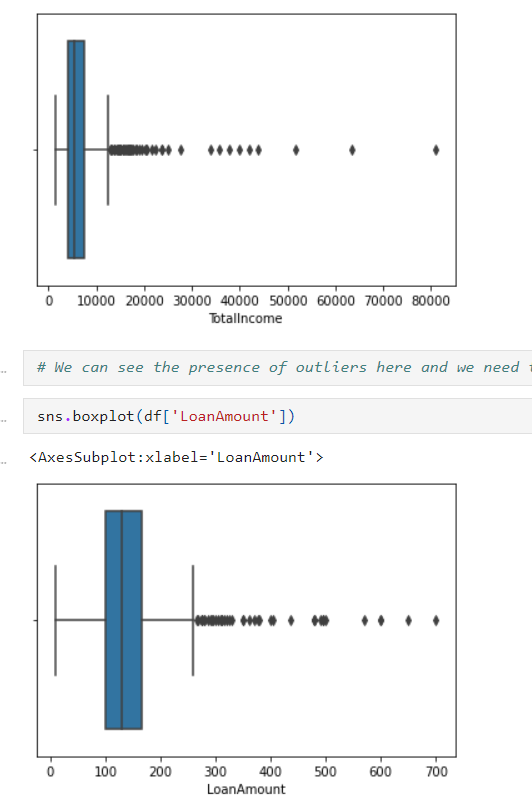
Credit\_History has high positive correlation i.e., 0.54 and is an important feature in prediction of Loan Status

**Checking for outliers in our dataset**

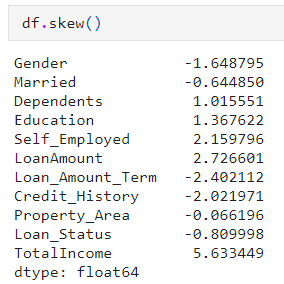
Often there exist data objects that do not comply with the general behaviour or model of the data. Such data objects, which are grossly different from or inconsistent with the remaining set of data, are called outliers. Many data mining algorithms try to minimize the influence of outliers or eliminate them all together. This, however, could result in the loss of important hidden information. In other words, the outliers may be of particular interest, such as in the case of fraud detection, where outliers may indicate fraudulent activity

We need not check for outliers or skewness for categorical columns. So hence let’s check for continuous ones

The best way to check for the presence of outliers is plotting graphs preferably boxplot



Since the outliers are present , skewness also would be present . Let’s check for it

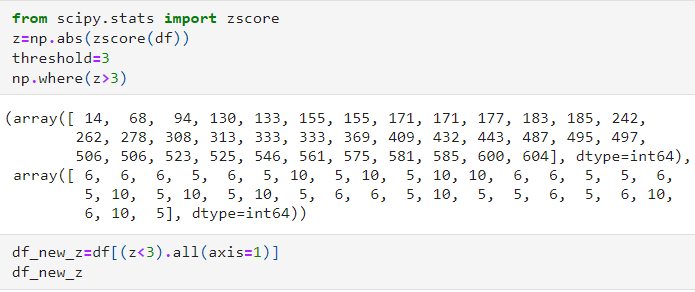


-(0.5) to +(0.5), the skewness within this range is acceptable

We have skewness of few columns outside the above mentioned range and they need to be treated. The columns with high skewness are "Loan Amount" and "TotalIncome".

**Considering to remove the outliers**

Z-score is used here to remove the outliers thereby handling skewness.



Now the outliers and skewness is removed from the dataset .

**EDA Concluding Remark**

Now the data is analysed, have performed necessary modifications to dataset such as label encoding for changing the datatype to identifiable type , missing value treatment , removal of duplicates, removing the outliers and skewness .

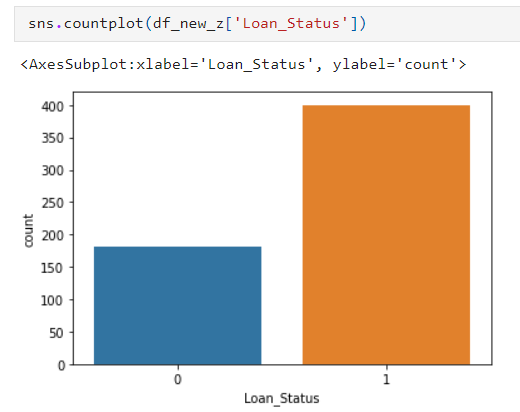
Now before building the model, lets first plot feature importance and for classification problems we need to check if the classes in the target columns are balanced or not.

If the classes are not balanced then we need to balance them using over sampling or under sampling methods

Then the entire dataset must be separated as X and y. X being the features and y containing only the target label.

Then we need to perform train-test split on X and y which inturn gives us X\_train, X\_test, y\_train and y\_test data

**Analyzing the target column -** This is to check if the target classes are balanced or not



we can clearly see that the classes 0 and 1 are not balanced

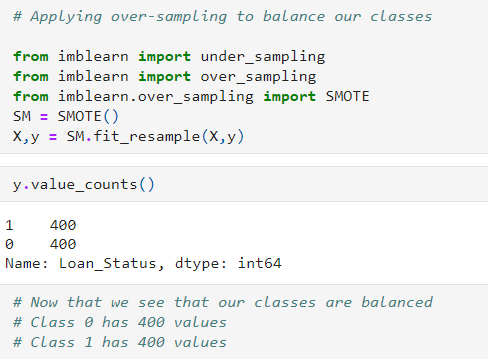
Lets try to balance them first

We need to split the data into X and y before we could try balancing the classes

Splitting the dataset into X features and y label

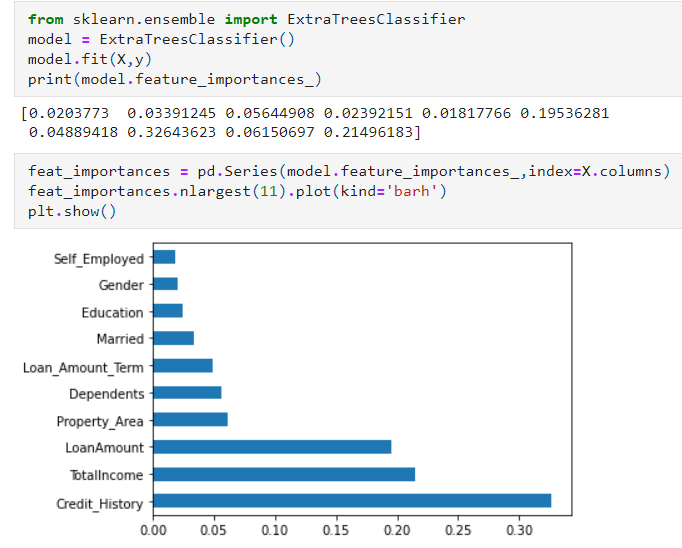
X **=** df\_new\_z**.**drop(["Loan\_Status"],axis**=**1)

y **=** df\_new\_z["Loan\_Status"]

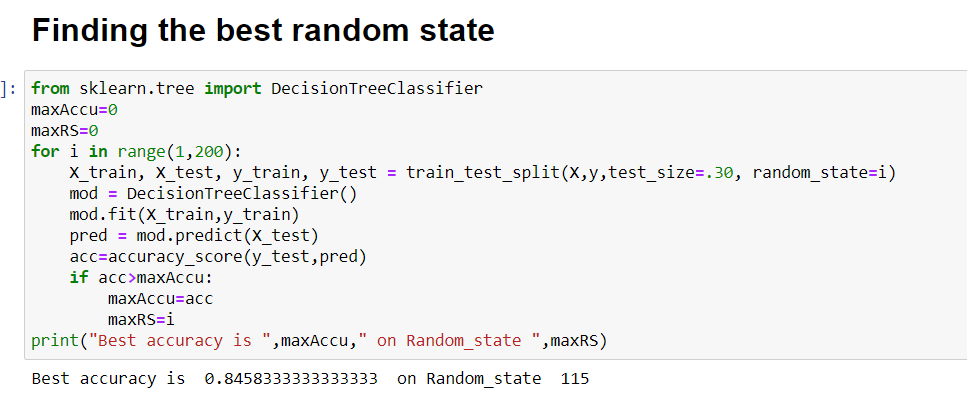


I have applied over-sampling and balanced the class using the above code. Now both the classes are balanced each having the count of 400.

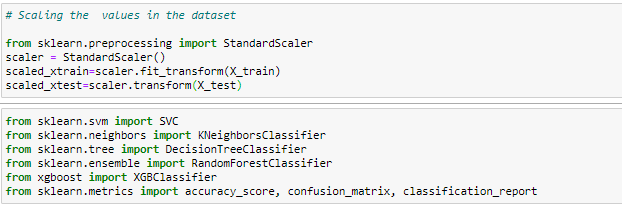
Now let’s plot the important feature using Extra Tree Classifier

  
“Credit\_History” of an applicant is a very important feature to predict the status of loan application.

Let’s find the best random state and accuracy for implementing our models



**Modelling**

****

Here we are scaling independent variables of both train and test set.

Five different classifiers are used in this project:

-SVC

-Random Forest

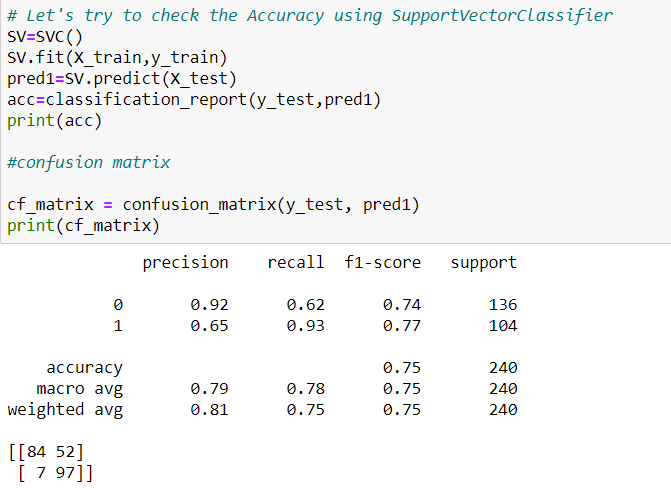
-XGBoost

-Decision Tree

Classification is defined as the act or process of putting things into groups based on ways that they are alike. Classification is the process of finding a model (or function) that describes and distinguishes data classes or concepts, for the purpose of being able to use the model to predict the class of objects whose class label is unknown.

**SVC**

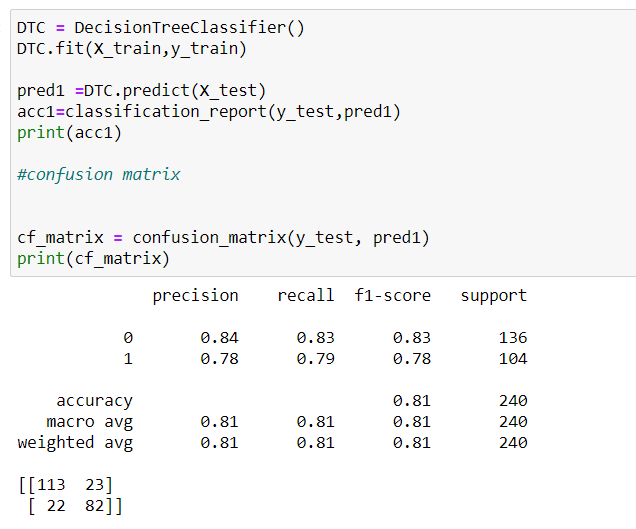
The objective of a Linear SVC (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes, your data

****

The accuracy score obtained by SupportVector Classifier is 75%

**Decision Tree**

The decision tree Algorithm belongs to the family of supervised machine learning algorithms. It can be used for both a classification problem as well as for regression problem. The goal of this algorithm is to create a model that predicts the value of a target variable, for which the decision tree uses the tree representation to solve the problem in which the leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

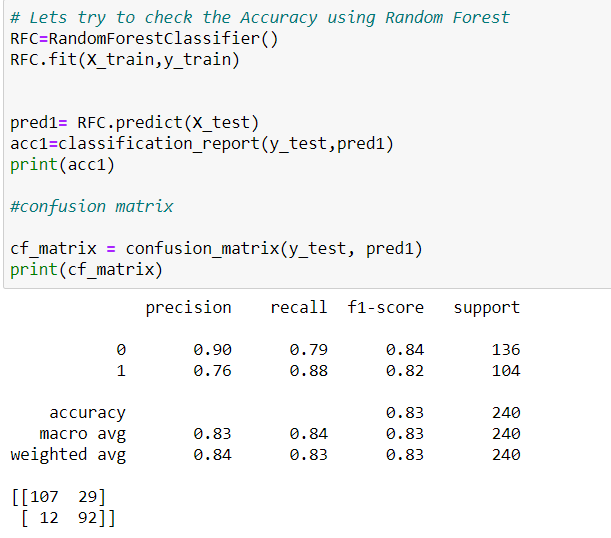


The accuracy score obtained by DecisionTree Classifier is 81%

**Random Forest**

Random forests (RF) are basically a bag containing n Decision Trees (DT) having a different set of hyper-parameters and trained on different subsets of data. Let’s say I have 100 decision trees in my Random Forest. These decision trees have a different set of hyper-parameters and a different subset of training data, so the decision or the prediction given by these trees can vary a lot.

Let’s consider that I have somehow trained all these 100 trees with their respective subset of data. Now I will ask all the hundred trees in my bag that what is their prediction on my test data. Now we need to take only one decision on one example or one test data, we do it by taking a simple vote. We go with what the majority of the trees have predicted for that example.



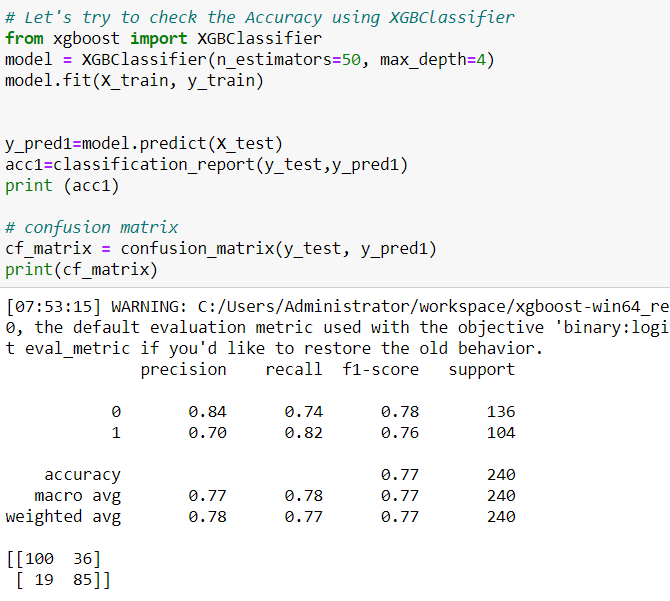
The accuracy score obtained by RandomForest Classifier is 83%

**XGBoost**

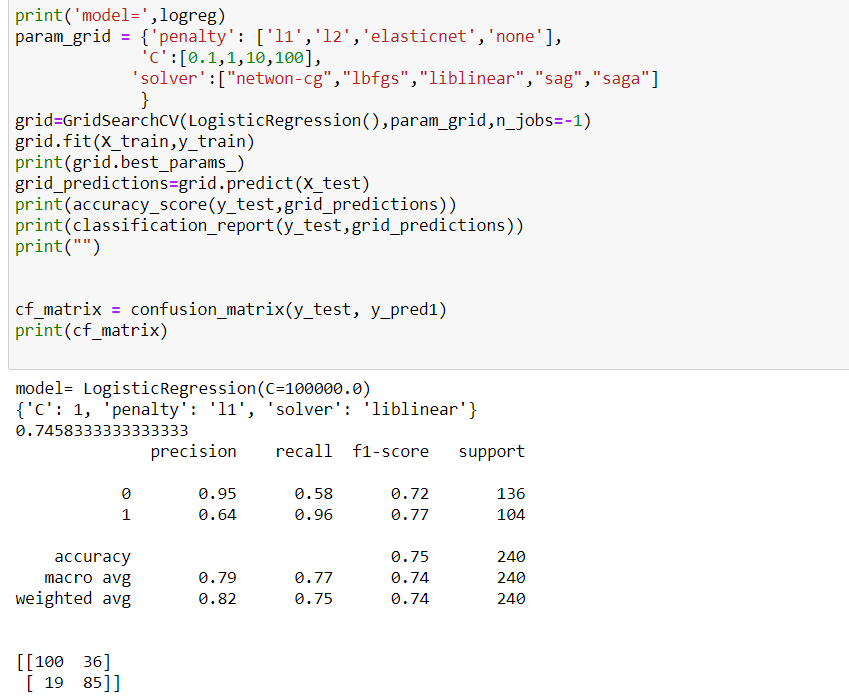
It is also known as “Extreme Gradient Boosting”. It carries out the gradient boosting decision tree algorithm. It has several different names like gradient boosting, gradient boosting machine, etc.

Boosting is nothing but ensemble techniques where previous model errors are resolved in the new models. These models are added straight until no other improvement is seen. One of the best examples of

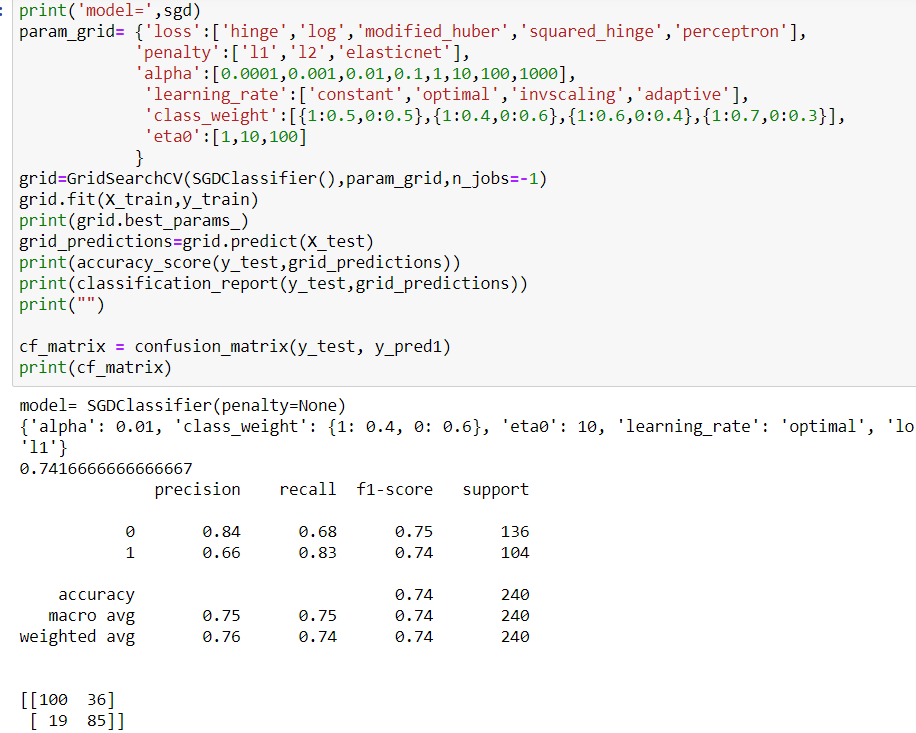
such an algorithm is the AdaBoost algorithm. Gradient boosting is a method where the new models are created that computes the error in the previous model and then leftover is added to make the final prediction



The accuracy score obtained by XGB Classifier is 77%



The accuracy score obtained by Logistic Regression is 75%

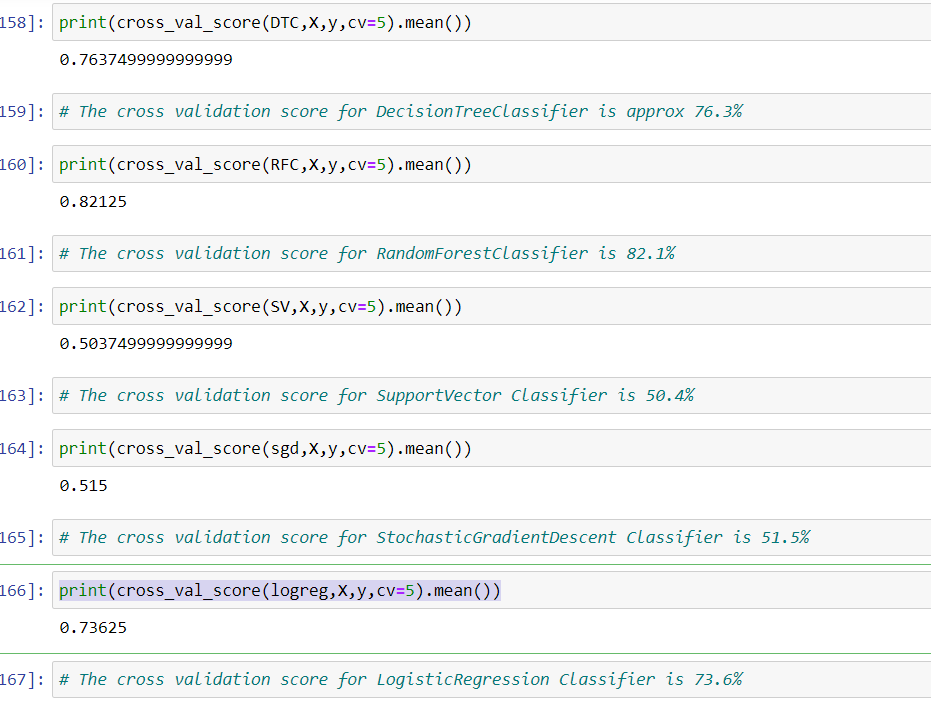


The accuracy score obtained by Stochastic Gradient Descent is 74%

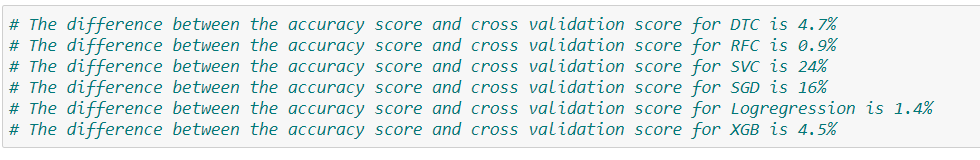
These performances maybe due to overfitting, hence finding CV (Cross validation) score is very important . The model with least difference between the performance and the CV score would be the best model.

Code for finding the cross validation is,

print(cross\_val\_score(model,X,y,cv=5).mean())

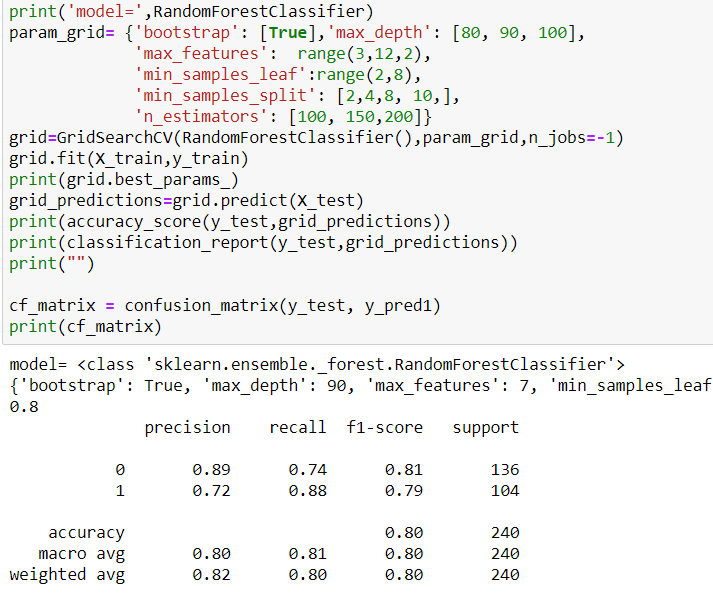


Finding the difference between the model performance ie., accuracy score and cross validation score to figure out the best model.



Our best model is RandomForest Classifier

Performing hyper-parameter tuning on best model to increase the performance.



**Conclusion**

We studied the existing load application with many features. To predict the status we used classification algorithms like decision tree classifier, random forest classifier, XGB classifier, SGD and SVC classifier. We looked at model performance metrics derived from the confusion matrix. Performance metrics such as accuracy, recall, and precision are derived from the confusion matrix. It is strong with respect to class skew, making it a reliable performance metric in predictions.

The very important task of loan approval for bank organizations is achieved through our machine learning models