**Loan Application Status Prediction**

**Introduction**

A loan application is used by borrowers to apply for a loan. Through the loan application, borrowers reveal key details about their finances to the lender. The loan application is crucial to determining whether the lender will grant the request for funds or credit. In this blog, we will build a machine learning 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.

**Table of Contents**

1. Problem Definition
2. Understanding the data
3. Importing the necessary libraries
4. Reading the Data
5. Data Preprocessing

* Checking the data types
* Filling the null values

1. EDA (Exploratory Data Analysis)
2. Data Pre-Processing

* Label Encoding
* Ordinal Encoding

1. Plotting Outliers
2. Correlation Matrix
3. Checking skewness of the data
4. Building Machine Learning Model
5. Standard Scaling the data
6. Using different Machine Learning algorithms
7. Hyper Parameter Tunning
8. AUC-ROC Curve
9. Conclusion

**Problem Definition**

A loan company provides loans to their customers after an intense process of verification and validation. But, still they don’t have the assurance if the applicant will be able to repay the loan. The customer first applies for loan and after that, the company validates the customers eligibility for loan.

**Understanding the data**

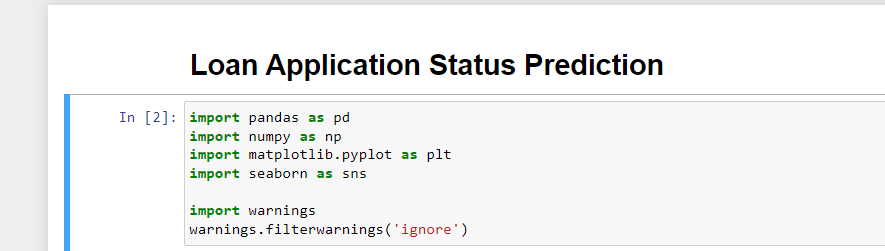
This dataset includes details of applicants who have applied for loan. The dataset includes details like credit history, loan amount, their income, dependents etc.

This dataset has 614 rows and 13 columns. Based on the problem, our target variable consist of two values, yes (if the loan application is approved) and no (if the loan application is rejected). This will make it a binary classification problem where we have to predict the target variable, Loan Status.

**Importing the necessary libraries**

We will use Jupyter Notebook to prepare our data and predict the target variable. We will need to import few libraries for this,

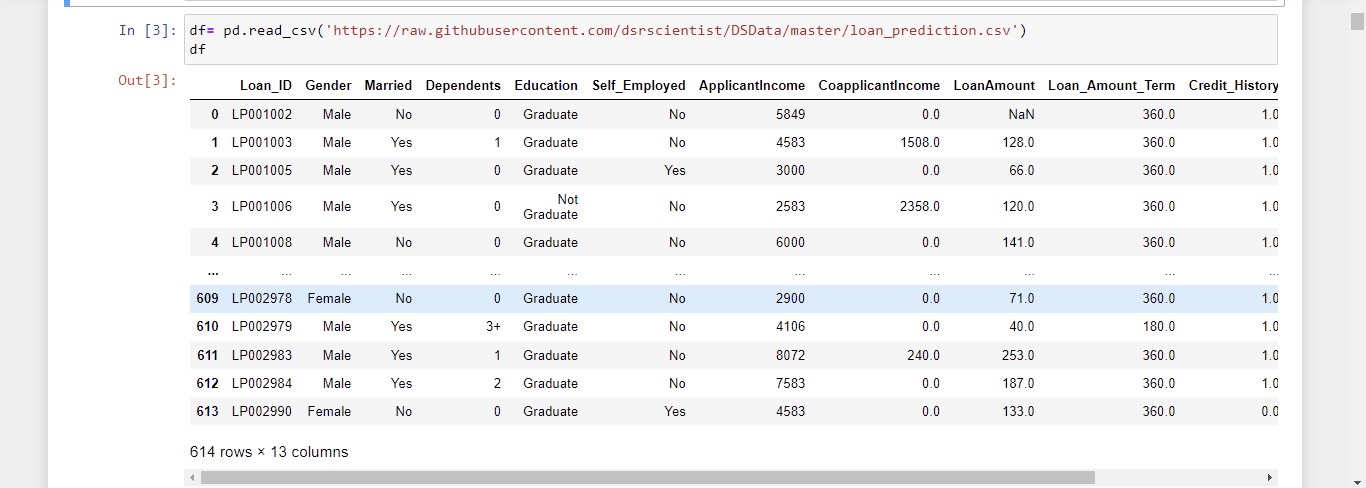
* Pandas
* Numpy
* Matplotlib
* Seaborn



**Reading the Data**

Uploading the CSV file which contains the dataset with help of pandas library, df=pd.read\_csv('https://raw.githubusercontent.com/dsrscientist/DSData/master/loan\_prediction.csv')

df



Here, the independent variables are,

* Loan\_ID, Gender, Married, Dependents, Education, Self\_Employed, ApplicantIncome, CoapplicantIncome, Loan\_Amount, Loan\_Amount\_Term,  Credit History, Property\_Area

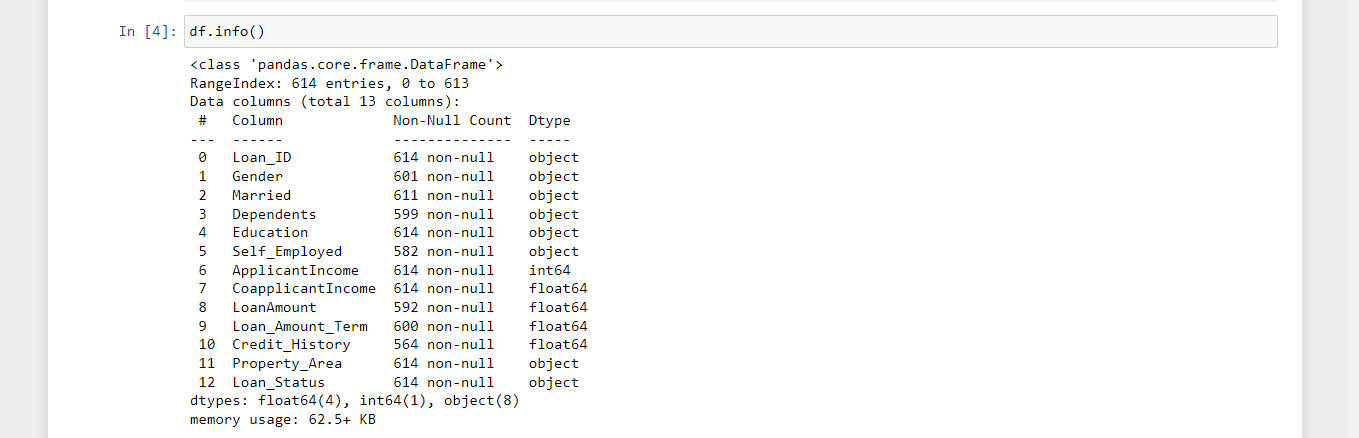
And, the dependent variables are,

* Loan\_Status

Numerical Columns consists of  Loan ID, Applicant Income, Co-applicant Income, Loan Amount, and Loan amount term

Categorical Columns consists of Gender (Male/Female), Married (Yes/No), Dependents (from 1,2,3+) , Self-Employed (No/Yes), Property Area (Rural/Semi-Urban/Urban), Education (Graduate / Not Graduate), credit history(Yes/No), Loan Status (Yes/No)

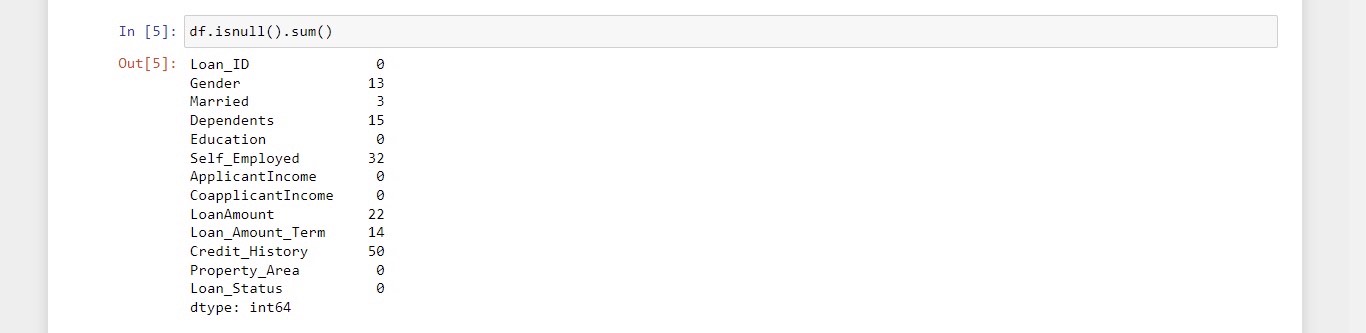
**Data Preprocessing**

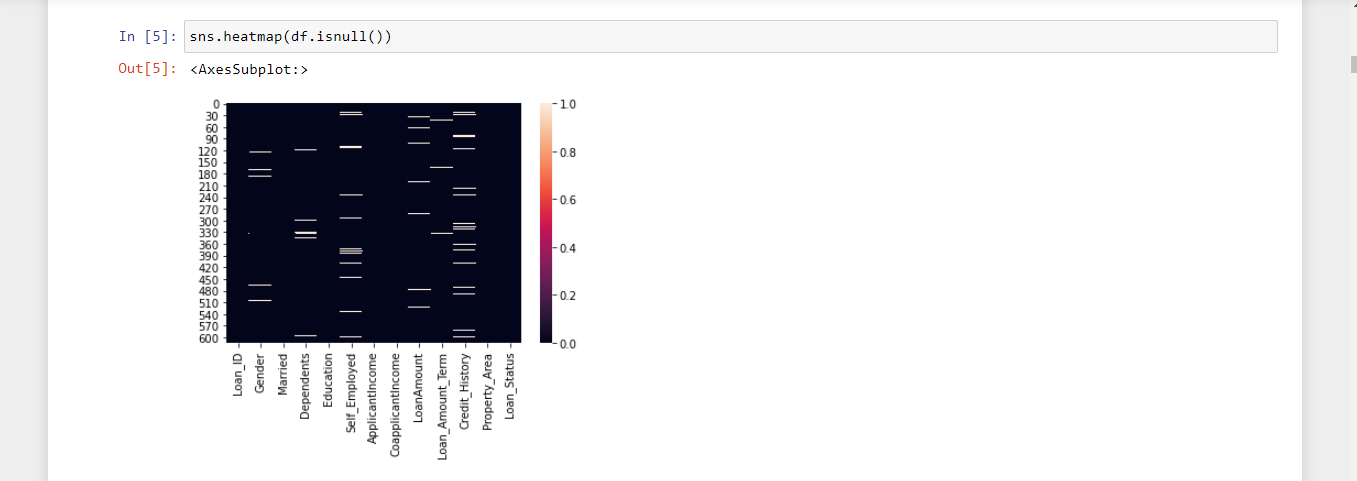


Here, there are three formats of data types:

* Float64: This says that we have 4 columns with decimal values present in it. CoapplicantIncome, LoanAmount, Loan\_Amount\_Term , Credit\_History are with float values.
* Int64: This says that we have 1 column with integer value present in it. ApplicantIncome has integer value.
* Object: This says that we have 8 columns with categorical values present in it. Loan\_ID, Gender, Dependents, Married, Property\_Area, Self\_Employed, Education and Loan\_Status are with categorical values.

Now, we will check if there’s any null value present in our data,

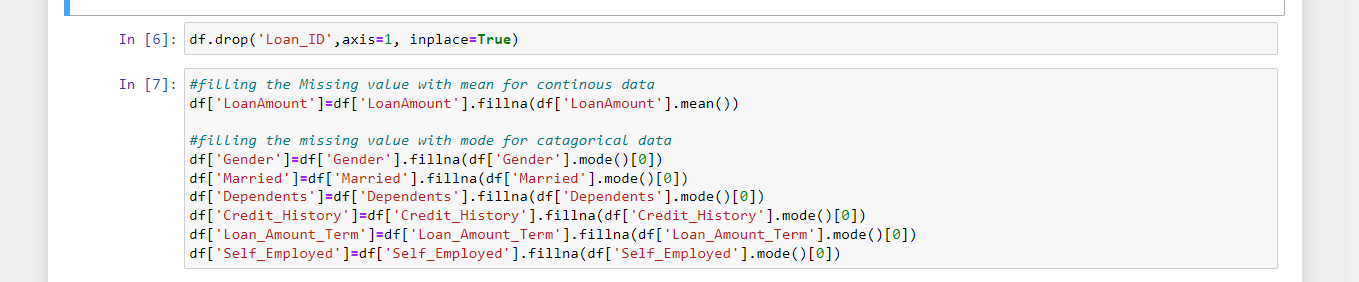


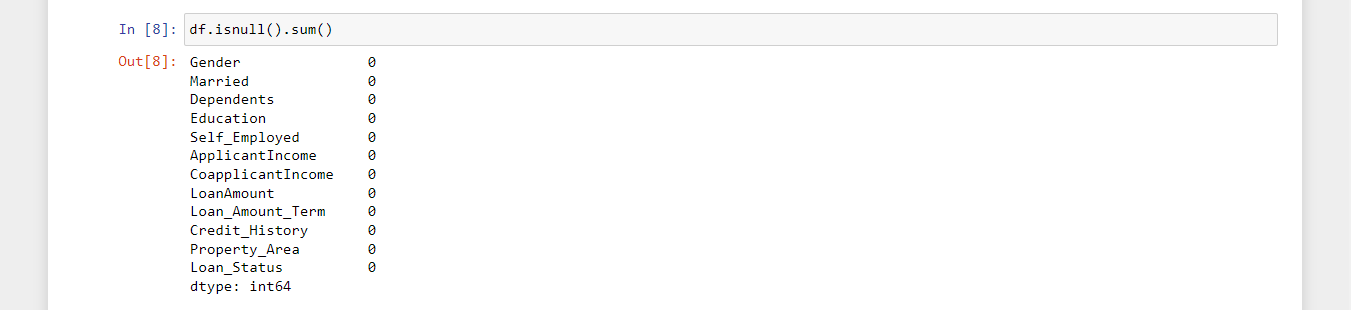


There are null values present in Gender, Married, Dependents, Self\_Employed, LoanAmount, Loan\_Amount\_Term, Credit\_History and Loan\_Status.

Let’s drop the Loan\_ID column as it is not of much use.

Then, we will fill in the null values by imputation method. We will fill the missing values with mean for continuous data and with mode for categorical data.

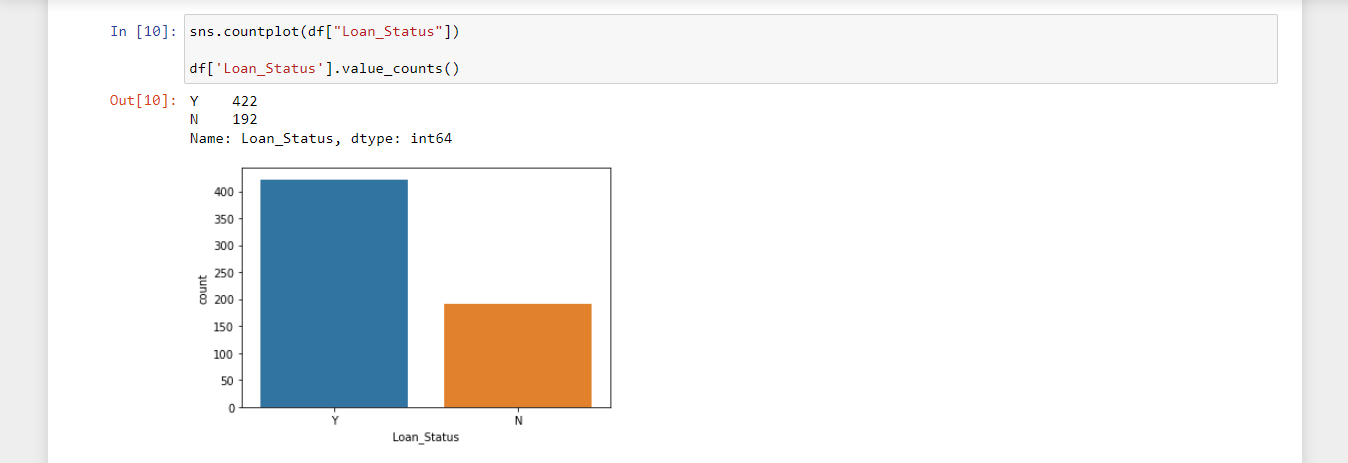


Now we have removed all the null values from our data.

**EDA (Exploratory Data Analysis )**

**Univariate Analysis**

* Loans approved Vs rejected



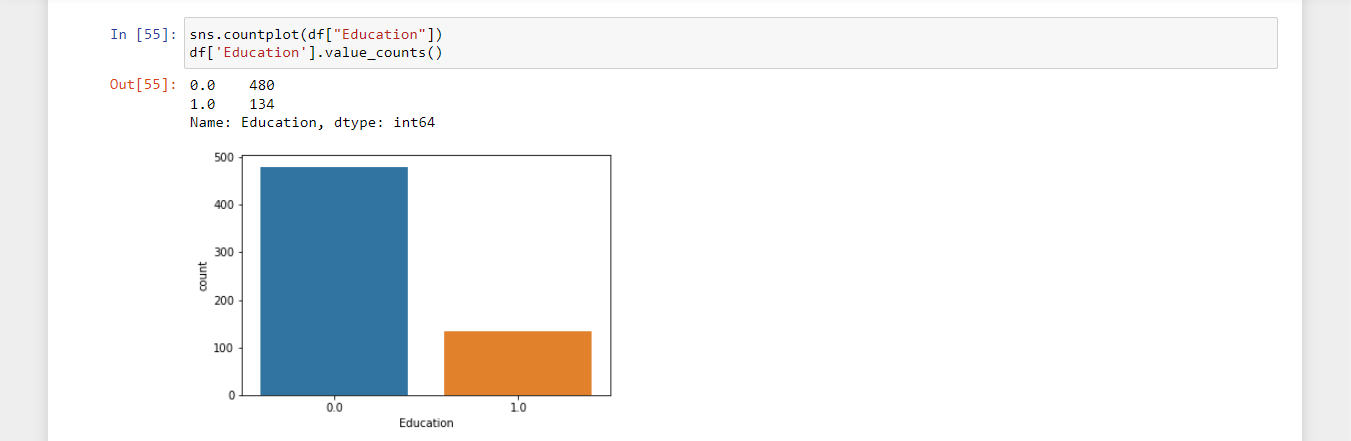
Here, the ratio of approved loan application is more than rejected ones.

* Married Vs Non Married applicants



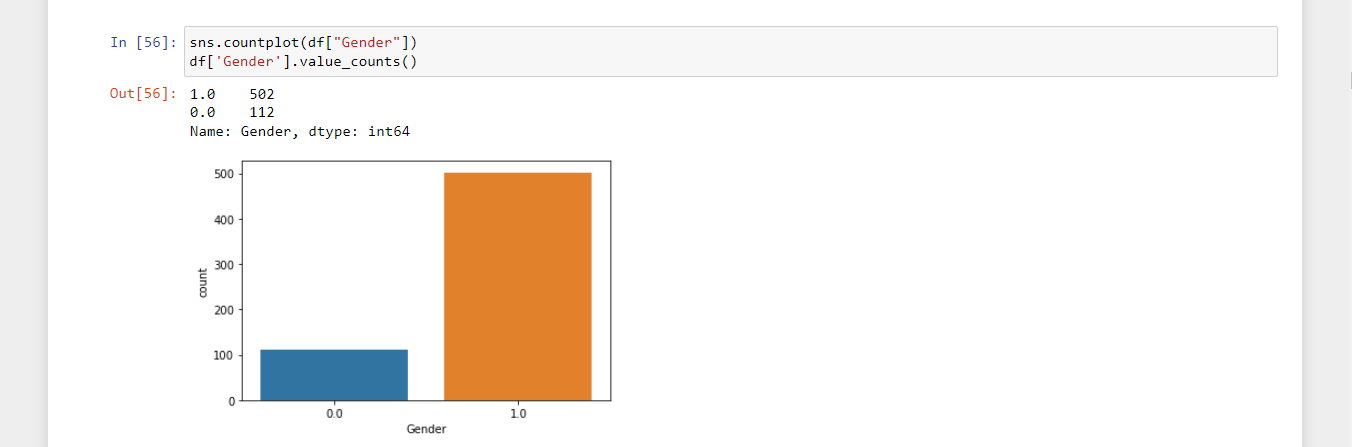
Number of Married applicants (1.0) are more than non- married (0.0) applicants.

* Graduate applicants Vs non- Graduate applicants



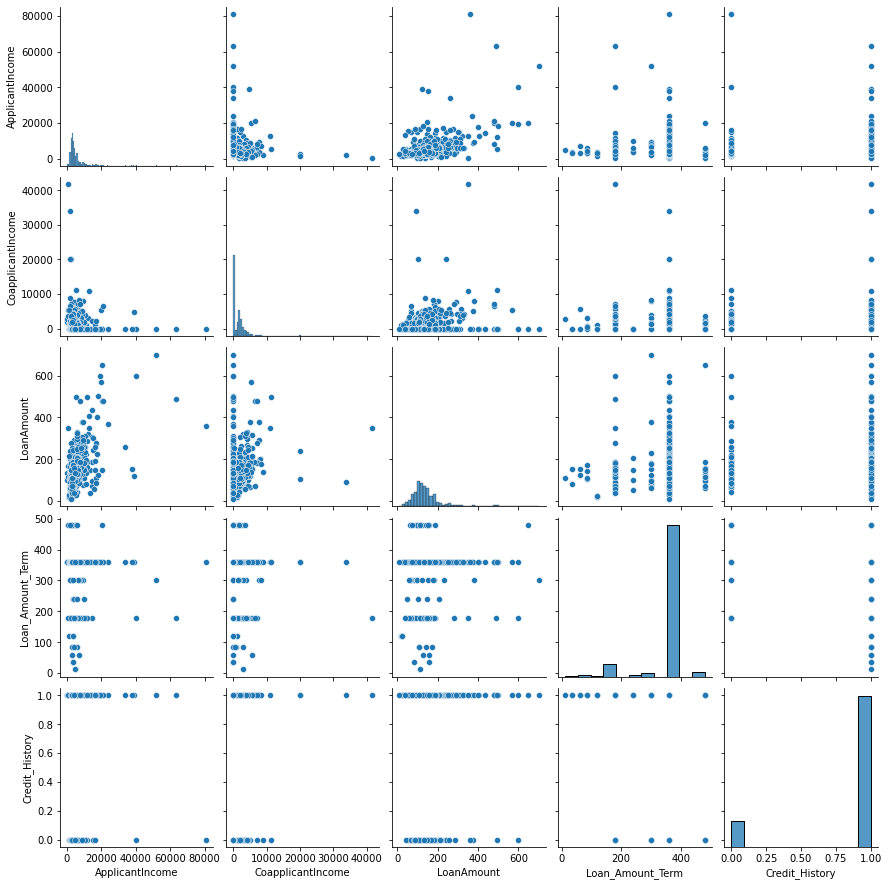
Number of Graduate applicants (0.0) are more than non- Graduate applicants (1.0).

* Male applicants Vs Female applicants



Number of Male applicants (0.0) is more than female applicants.

* **Pairplot of the dataset**

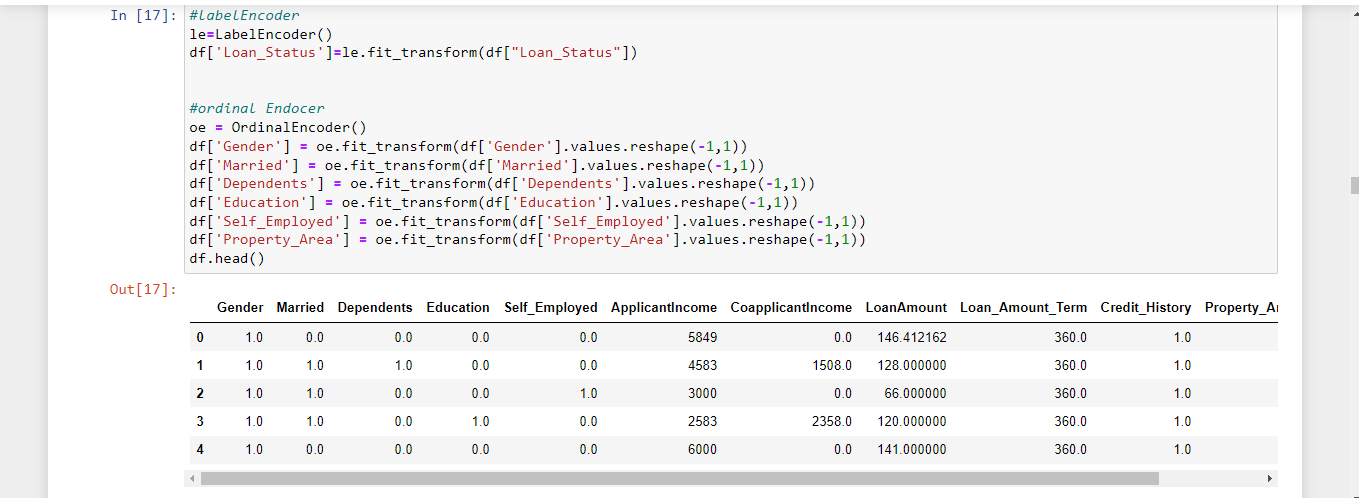
****

**Data Pre-Processing**

**Label Encoding and ordinal encoding the data**

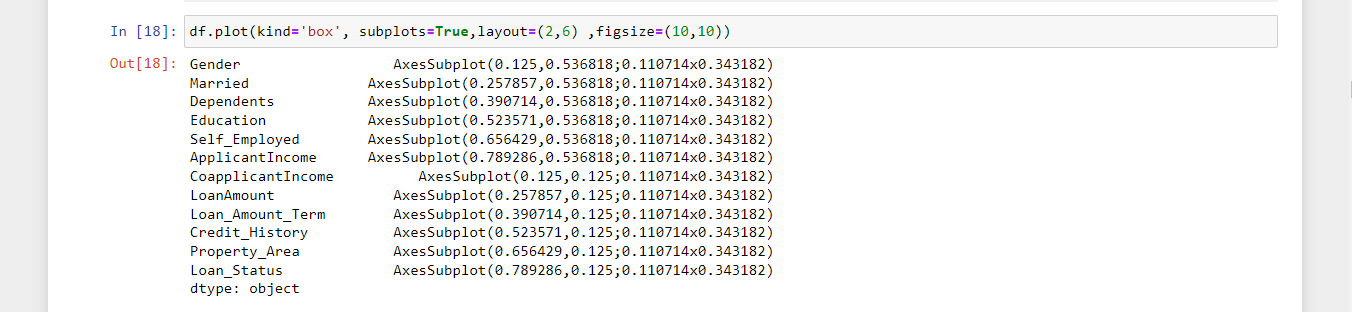
Now let’s convert the label form of the data into numeric form so that machine can read and analyse and predict the data properly. This is an important step for structured dataset in supervised learning. First, we will import the LabelEncoder and OrdinalEncoder from sklearn preprocessing.

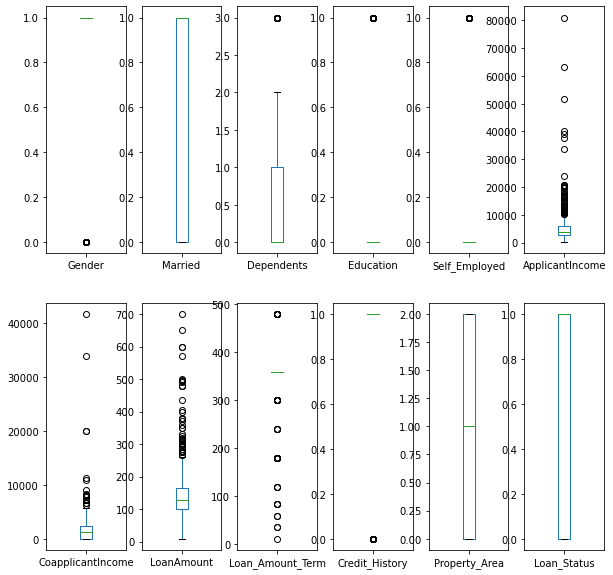




**Plotting Outliers**

Let’s now check if there are any outliers present in our data

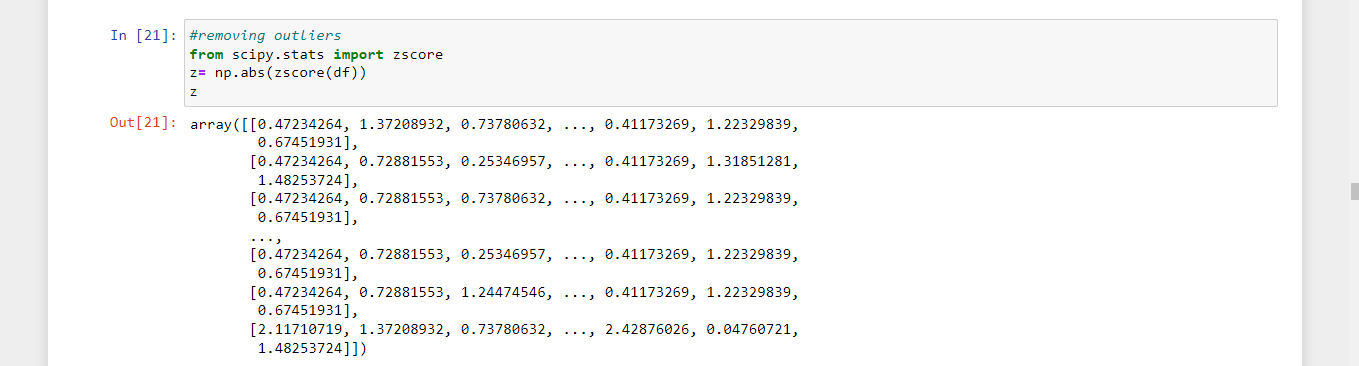




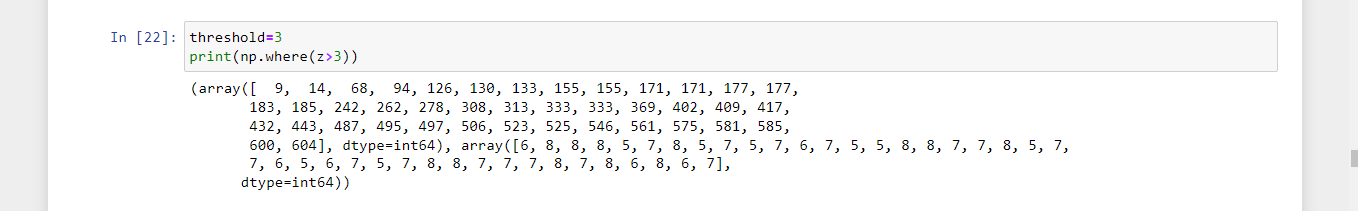
We can see from the boxplot that there are outliers present in Dependents, ApplicantIncome, CoapplicantIncome, LoanAmount, Loan\_Amount\_Term columns.

Let’s remove these outliers for our machine learning model to work smoothly and give better results. For this, we will import zscore library from scipy.stats.

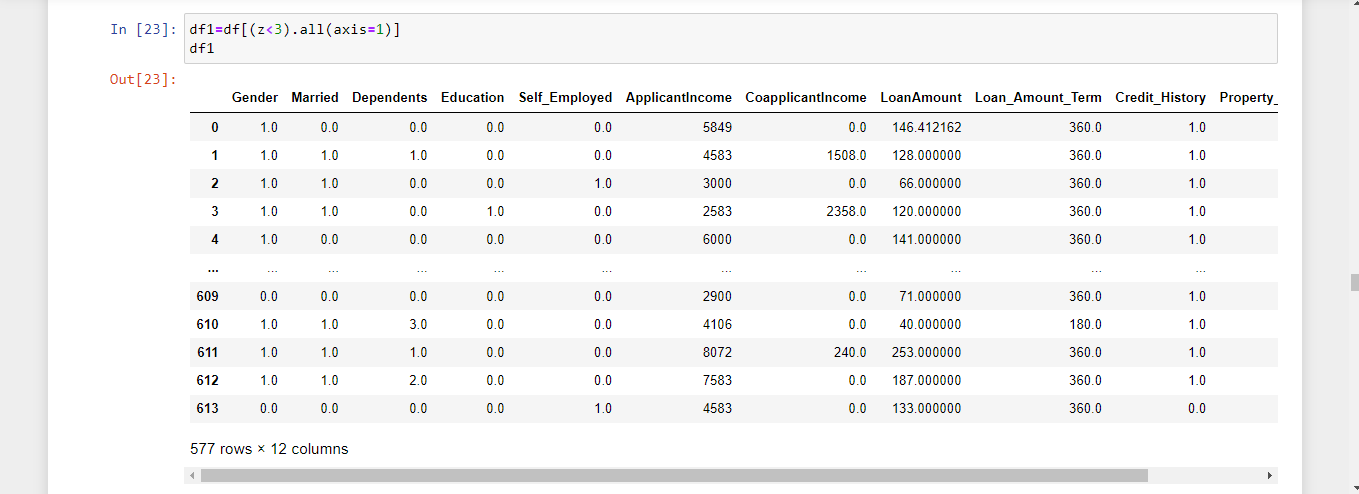
If the zscore of a data point is greater than 3, then there are outliers present as we can see in the above boxplot. We will set the threshold for zscore as 3, if there is any data point which has value greater than 3 then we will remove it. Let’s create a new dataframe (df1) for removing the outliers and storing the new data which has no outliers.



Now, we have got all the data points from our dataset. Let’s set the threshold as 3 and check out which point is an outlier



We have removed the outliers from the data and storing the new data in a new dataframe (df1)



**Correlation Matrix**

There are three types of correlation :

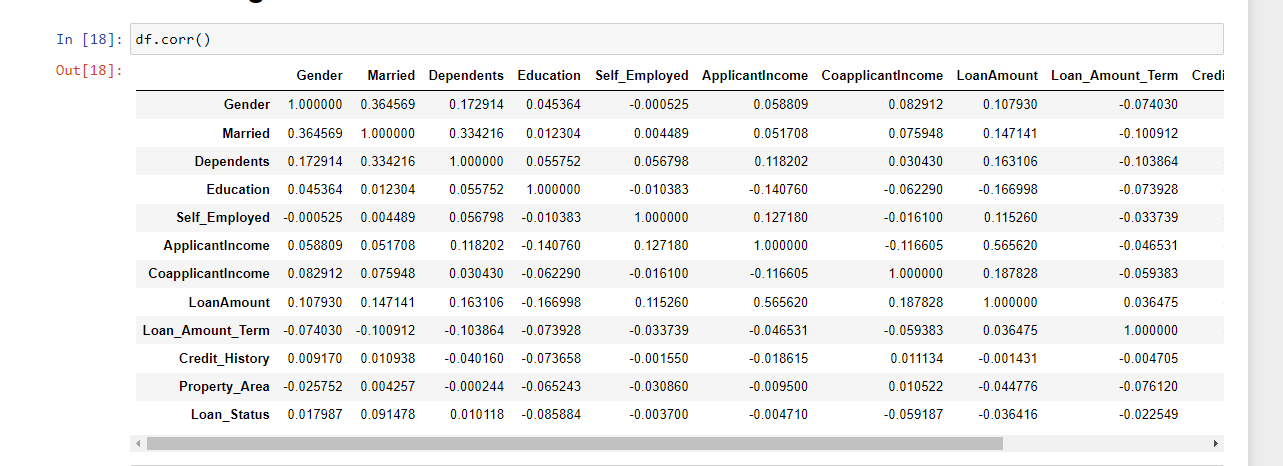
* Positive Correlation
* Negative Correlation
* Neutral Correlation

A correlation is said to be positive when both the variables move in the same direction.

A correlation is said to be negative when if one variable’s value increases, the other variable’s value decreases , in simple words, variables change in opposite direction

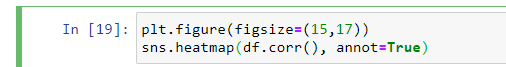
A correlation is said to be neutral or zero when the variables are unrelated, ie, there is no relationship in the change of variables.

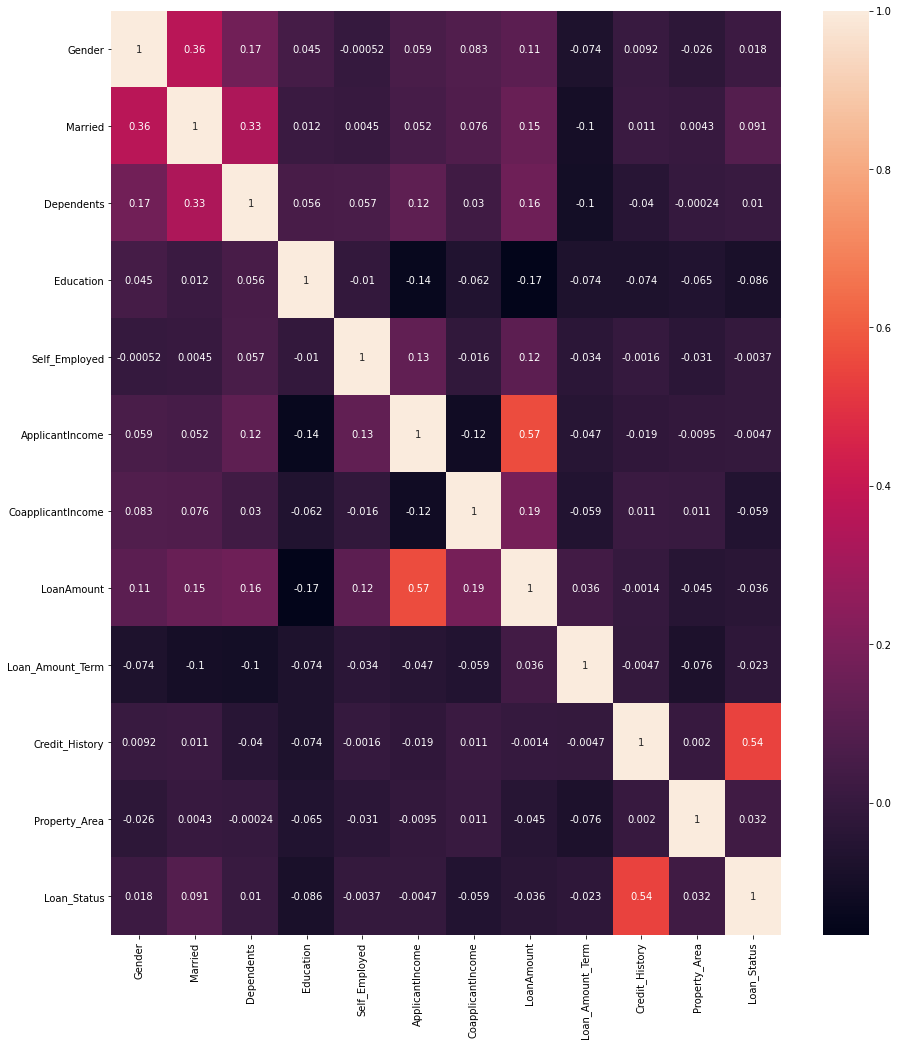
Let’s check the correlation between all the columns in our dataset



The data contains negative correlation, posititve correlation and netural correlation as well. To understand the correlation better, we will check the heatmap

Plotting the heatmap:

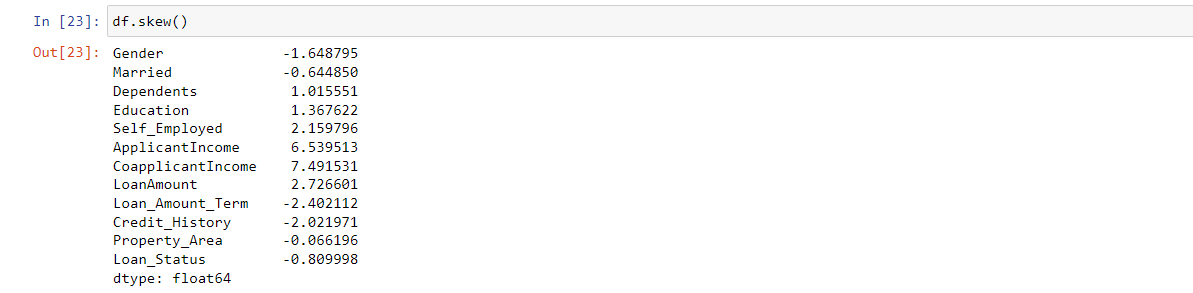




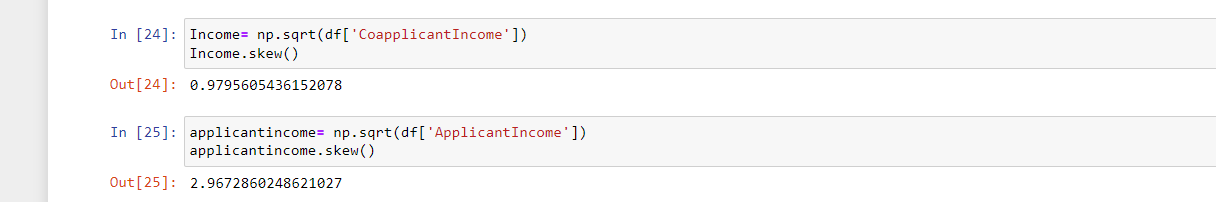
From the heatmap, we can clearly see the relation between two variables.

Checking the skewness of the data

Skewness is the measure of checking asymmetry in a distribution. If the model is highly skewed, then it may affect the prediction of whether the loan is approved or not.



Dependents, Applicant Income, Co applicant Income and Loan Amount are highly skewed columns. For the model to predict better results, we will handle the skewness by using Square Root Transformation Method.



The skewness of Co applicant Income from 7.49 reduced to 0.97 and the skewness of Applicant Income from 6.53 has reduced to 2.96.



The skewness of Dependents from 1.01 reduced to 0.56 and the skewness of Loan Amount from 2.72 has reduced to 1.31.

**Building Machine Learning Model**

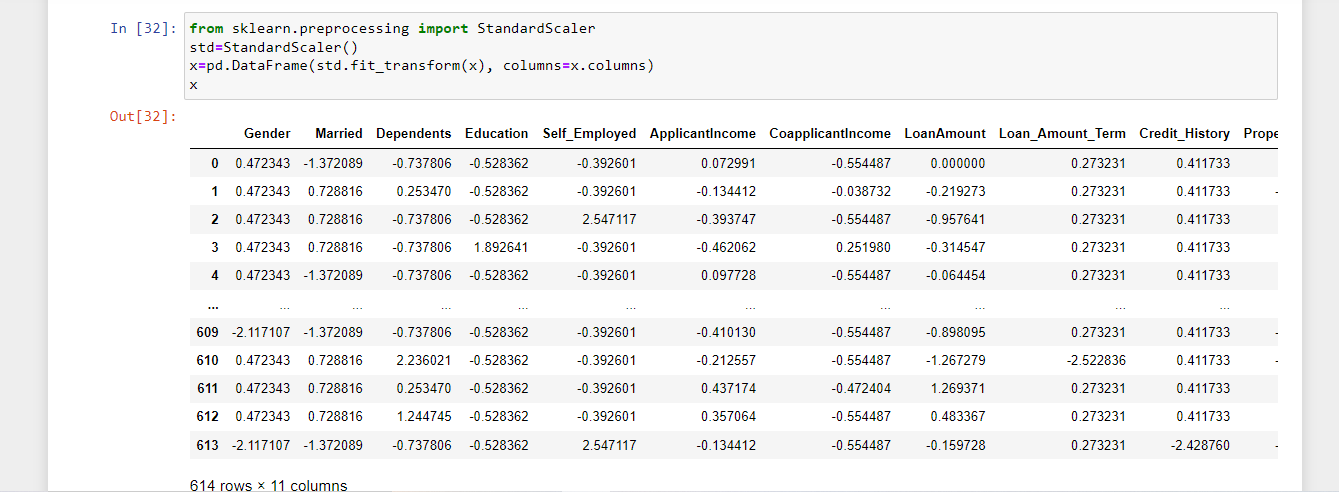
Let’s divide the model into training and testing parts. Creating X (input variables) and Y (Target variable) from the data df.



**Standard Scaling the data**

Standard Scaling helps in normalizing / standardizing the data before applying any machine learning model.

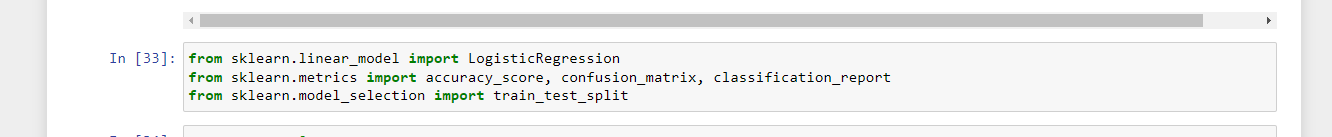
From sklearn.preprocessing, we will import the StandardScaler library



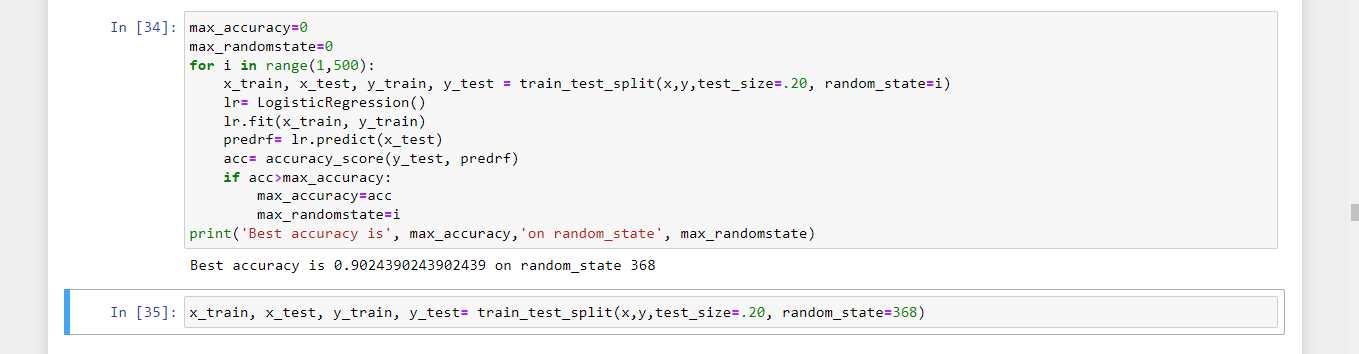
I will use train test split on the training data

Let’s import necessary libraries for machine learning.

* LogisticRegression from sklearn.linear\_model
* accuracy\_score from sklearn.metrics
* confusion\_matrix from sklearn.metrics
* classification\_report from sklearn.metrics
* train\_test\_split from sklearn.model\_selection



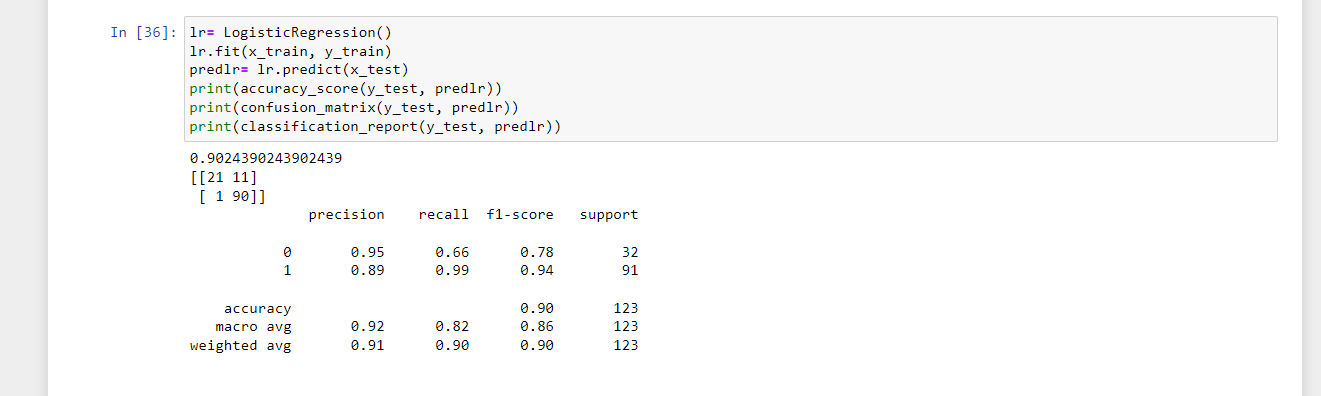
Now, I will find the best random state and it’s accuracy score fo the model to predict properly and then I will put the best random state I got in train test split on the training data forvalidation. I have splited the data into 80:30, ie, 80% data for training and 20% data for testing.



**Using different Machine Learning algorithms**

* Logistic Regression
* Support Vector Classifier
* KNeighbors Classifier
* Random Forest Classifier
* Decision Tree Classifier

**Logistic Regression**

****

Logistic Regression gives the accuracy score of 90.2%.

**Support Vector Machine (SVM)**

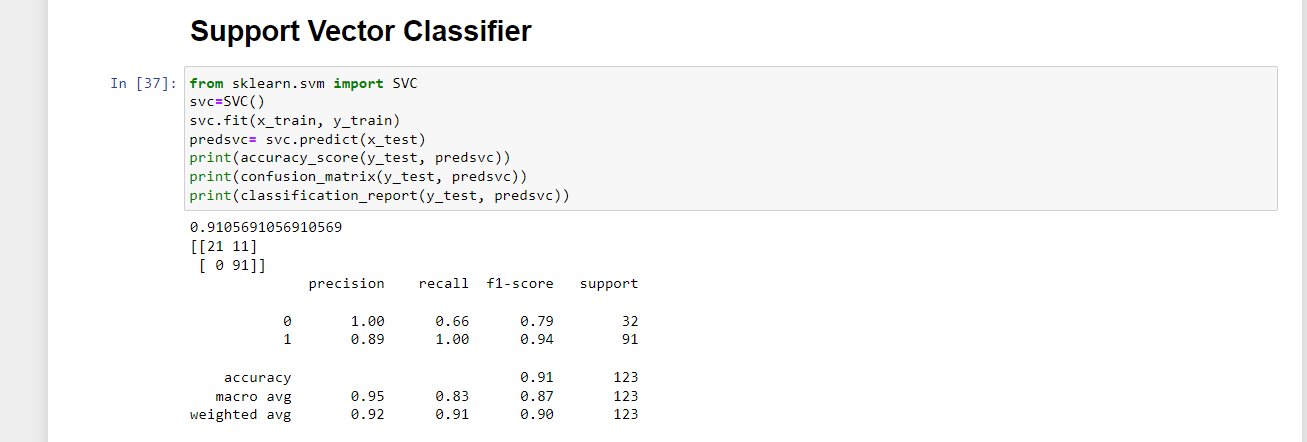
Support Vector Machine (SVM) is a supervised machine learning algorithm which is mostly used in classification problems. In this technique, we have to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. It chooses the extreme points that help in creating the hyperplane. These extreme cases are called as support vectors, and hence this algorithm is called as Support Vector Machine.

There are two types of SVM

* **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
* **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

For detailed information on SVM, visit :

<https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>



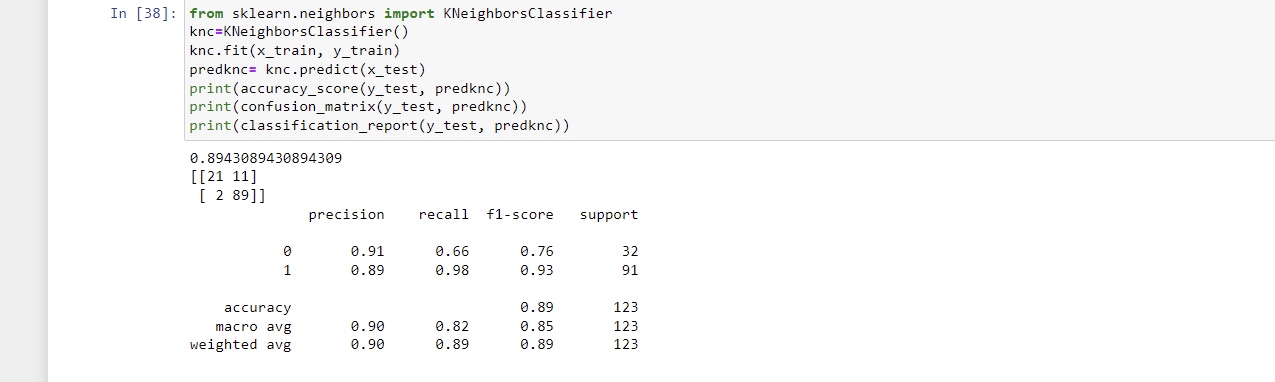
Support Vector Classifier gives the accuracy score of 91.05%.

# K-Neighbors Classifier

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique. It assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. It can be used for Regression as well as for Classification but mostly it is used for the Classification problems. It is a non-parametric algorithm, which means it does not make any assumption on underlying data. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

For detailed information on K-Neighbors Classifier , visit :

https://www.analyticsvidhya.com/blog/2021/04/simple-understanding-and-implementation-of-knn-algorithm/

****

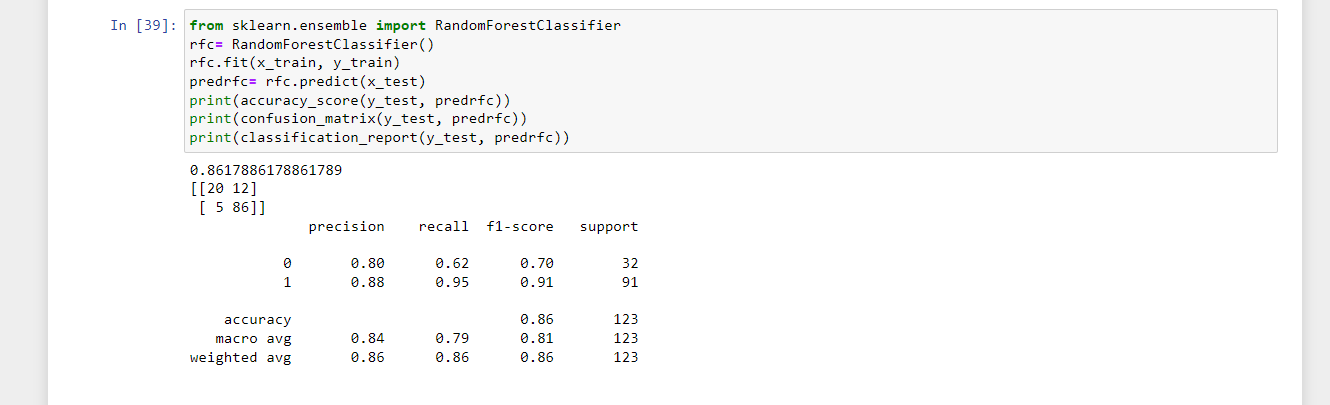
KNeighbors Classifer has accuracy score of 89.4%

**Random Forest Classifier**

Random Forest Algorithm is a very popular supervised machine learning algorithm. It is based on ensemble learning concept. It is used for both classification as well as regression problems. It 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. It takes less training time as compared to other algorithms. It predicts output with high accuracy, even for the large dataset it runs efficiently. It can also maintain accuracy when a large proportion of data is missing. 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.

For detailed information on Random Forest Classifier, visit :

https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

 For detailed information on Random Forest Classifier, visit :

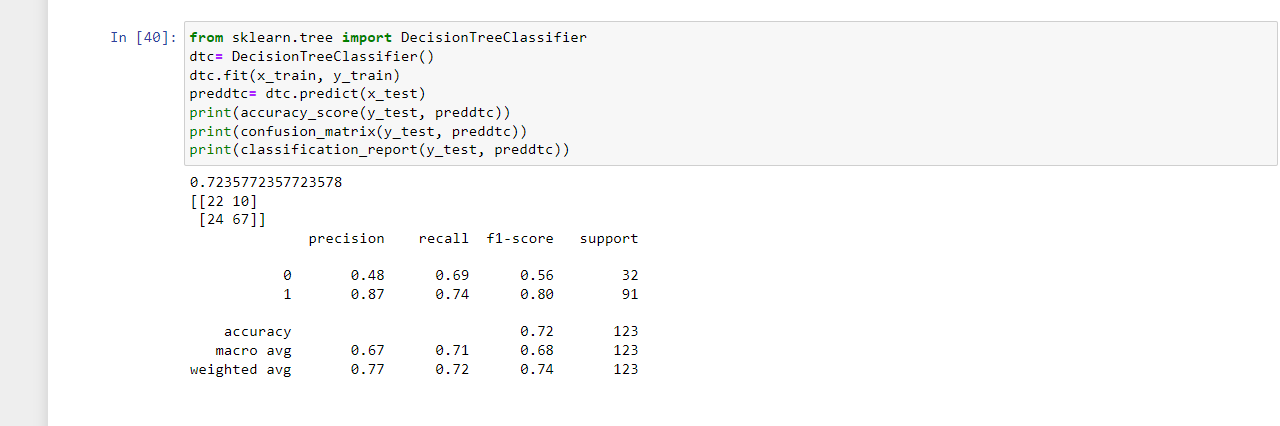
Random Forest Classifier has accuracy score of 86.1%.

**Decision Tree Classifier**

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is used for solving Classification problems. It uses a set of rules to make decisions, similarly to how humans make decisions. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

For detailed information on Decision Tree Classifier, visit :

https://towardsdatascience.com/decision-tree-classifier-explained-in-real-life-picking-a-vacation-destination-6226b2b60575



Decision Tree Classifier has accuracy score of 72.3%.

Among all the algorithms Random Forest Classifier performs best on the validation data with an accuracy score of **86.1%**.

To improve the accuracy score, I will hyper tune the score by using GridSearchCV

# Hyper Parameter Tuning

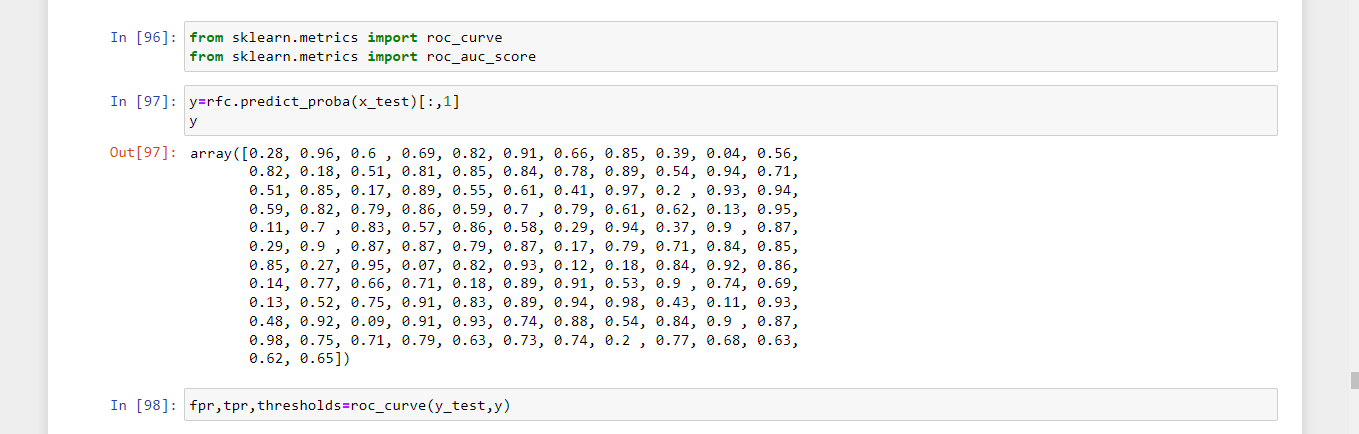
Importing GridSearchCV from sklearn.model\_selection



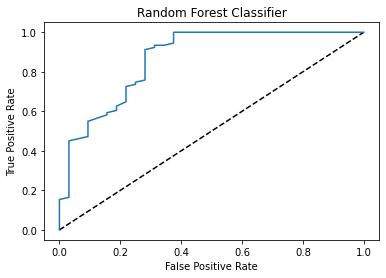
After hyper parameter tunning, the accuracy of the model has increased from 86.1% to 90.2%.

**AUC-ROC Curve**

The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC (Receiver Characteristic Operator) curve. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes. Importing the libraries, roc\_curve from sklearn.metrics for the roc-auc curve and roc\_auc\_score from sklearn.metrics for the roc score.







Now, let’s check the score of the curve,



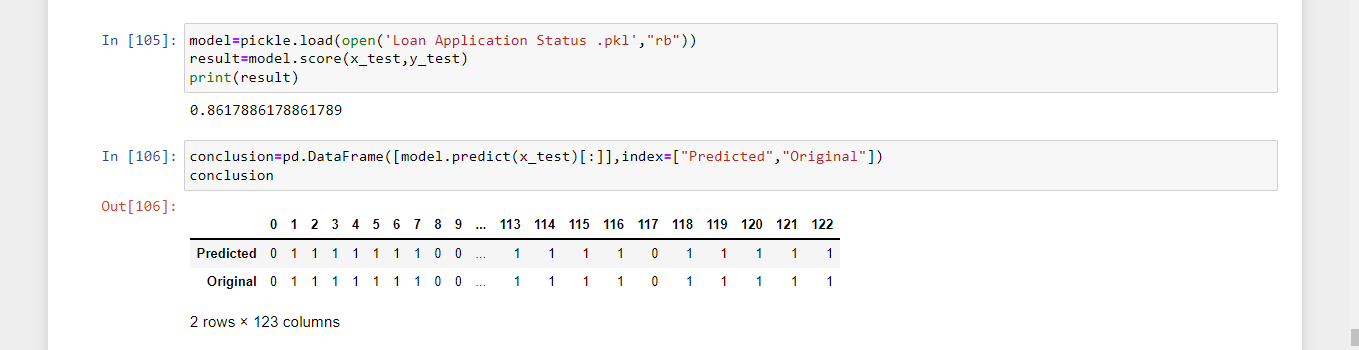
The score of the curve is 78.5%.

Now, I have saved my model in pickle format,

****

**Conclusion**

I have saved the best accuracy model, now let’s check the predicted and original values of our data.



The model predicted 86.1% data correctly.