**Loan Application Status Prediction using Machine learning**

Build predictive model to approve loan to the right applicants.



The use and development of computer systems that are able to learn and

adapt without following explicit instructions, by using algorithms and

statistical models to analyse and draw inferences from patterns in data.

Loans are the core business of banks. The main profit comes directly from the loan’s interest. The loan companies grant a loan after an intensive process of verification and validation. However, they still don’t have assurance if the applicant is able to repay the loan with no difficulties.

In this tutorial, we’ll build a predictive model to predict if an applicant is able to repay the lending company or not. We will prepare the data using Jupyter Notebook and use various models to predict the target variable.

We will be using Python for this course along with the below-listed libraries.

**Specifications**

* Python
* pandas
* seaborn
* sklearn

**Load the dataset in jupyter notebook**

IMPORTING PACAKES

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

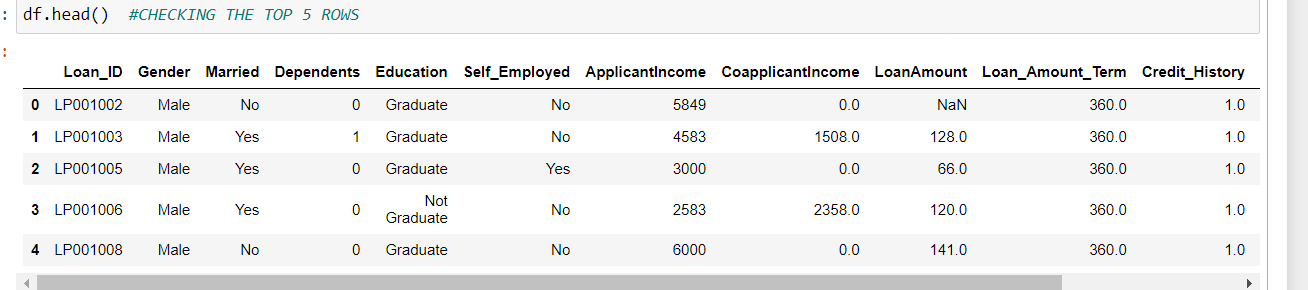
import seaborn as sns

import warnings

warnings.filterwarnings('ignore’)

Read the data with pandas dataframe ,using pd.read\_csv (https://raw.githubusercontent.com/dsrscientist/DSData/master/loan\_prediction.csv )



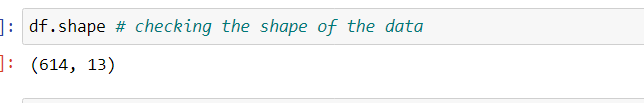


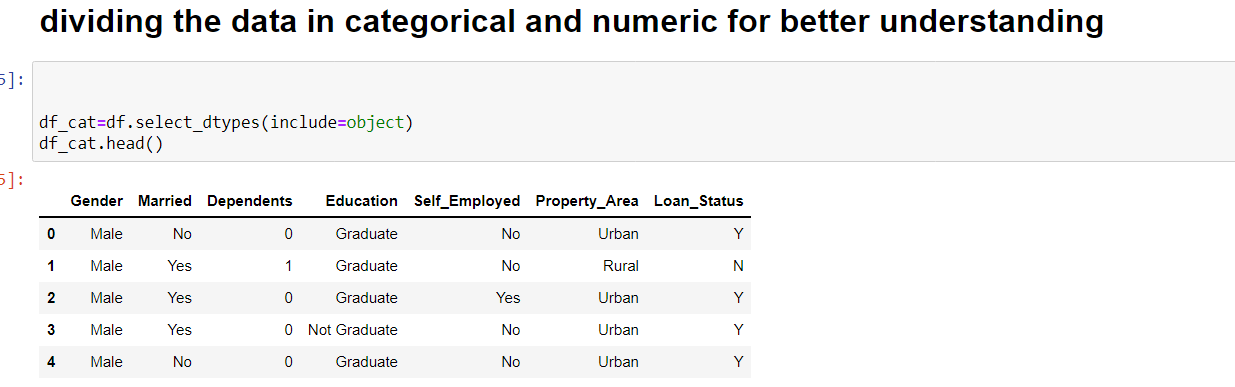
Check the size of data ,understanding the features of the dataset is important ,as domain knowledge is very important to get good results.

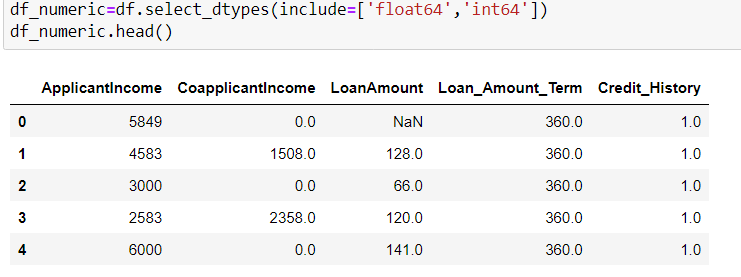
In this data set we have 12 independent variables and 1 target variable that is Loan\_Status**.**

**EXPLORATORY DATA ANAYLISIS**

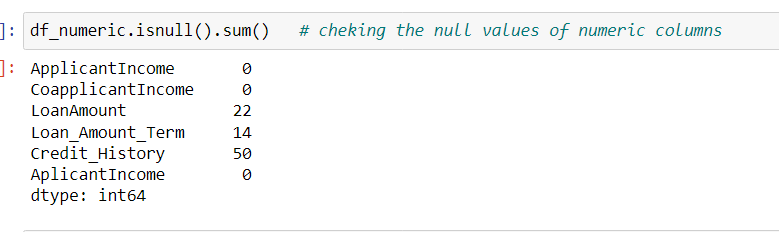
To begin with check the shape of the dataset , I like to divide the dataset into numeric and categorical columns to be able to understand the data better and perform analysis

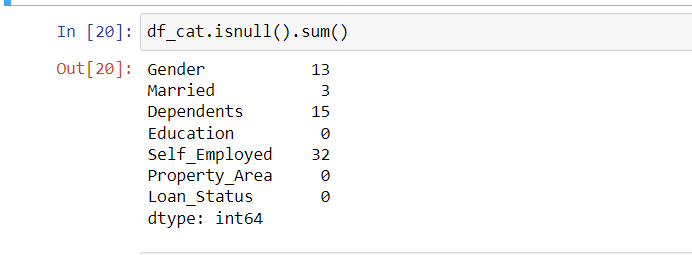




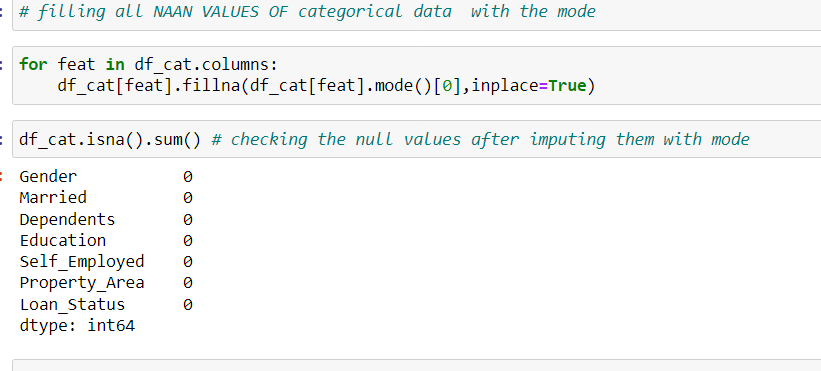


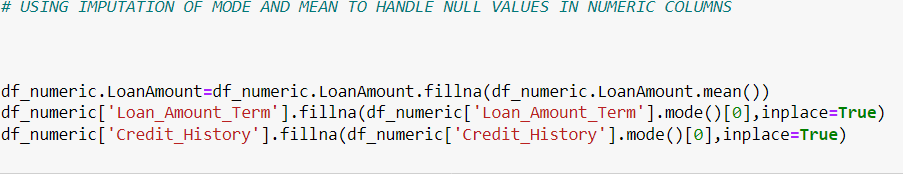
Check if there are any null values ,if present treat them with imputation techniques like mean ,median ,mode.



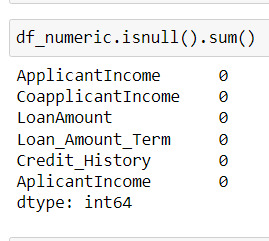


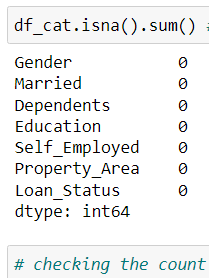
In loan application status dataset I have used mode to impute the null values of categorical data And mean for numeric data, for example-



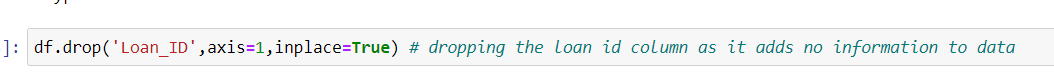


Checking the null values after applying the imputing techniques





check for the columns that are not adding any information and not helping the model, drop such columns for example-

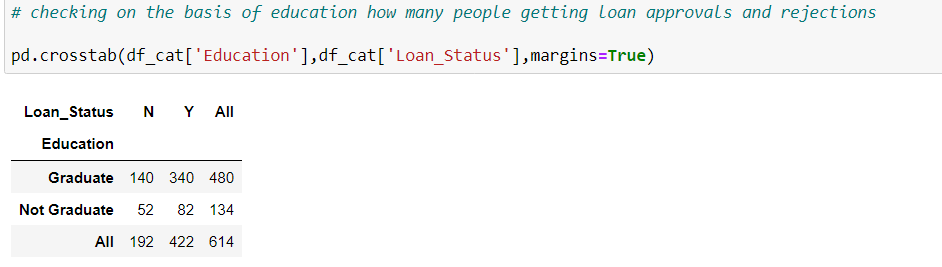


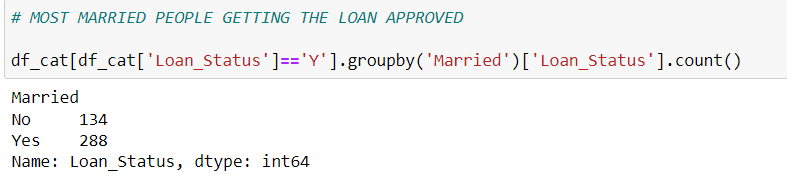
*#checking the number of male and female getting loan approvals and rejection*

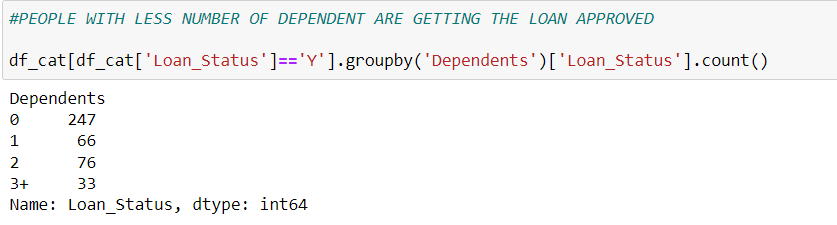
pd.crosstab(df\_cat['Gender'],df\_cat['Loan\_Status'],margins**=True**)

Out[100]:

| **Loan\_Status** | **N** | **Y** | **All** |
| --- | --- | --- | --- |
| **Gender** |  |  |  |
| **Female** | 37 | 75 | 112 |
| **Male** | 155 | 347 | 502 |
| **All** | 192 | 422 | 614 |





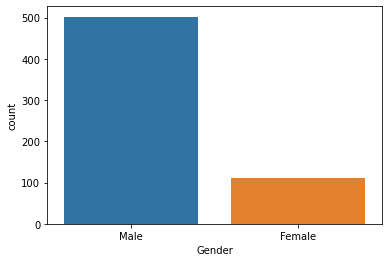


As we can observe that applicants with higher education are getting more loan approvals.Most married applicants are getting the approvals.

Applicants with less dependents are getting more approvals.

VISUALISATION OF DATA

1. Use countplot to visualise the data

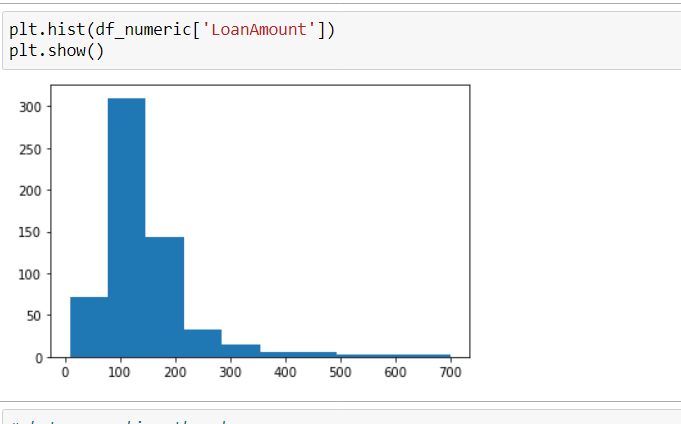






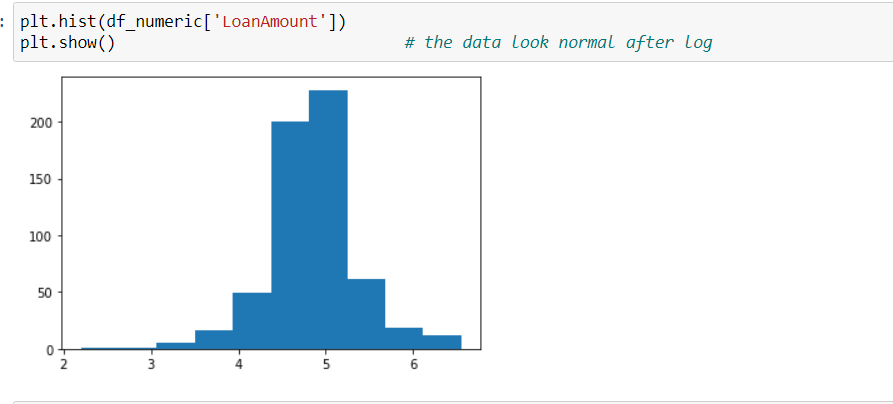
Majority of applicants are males, most of the applicants are married and very few applicants are self employed.

Using histogram

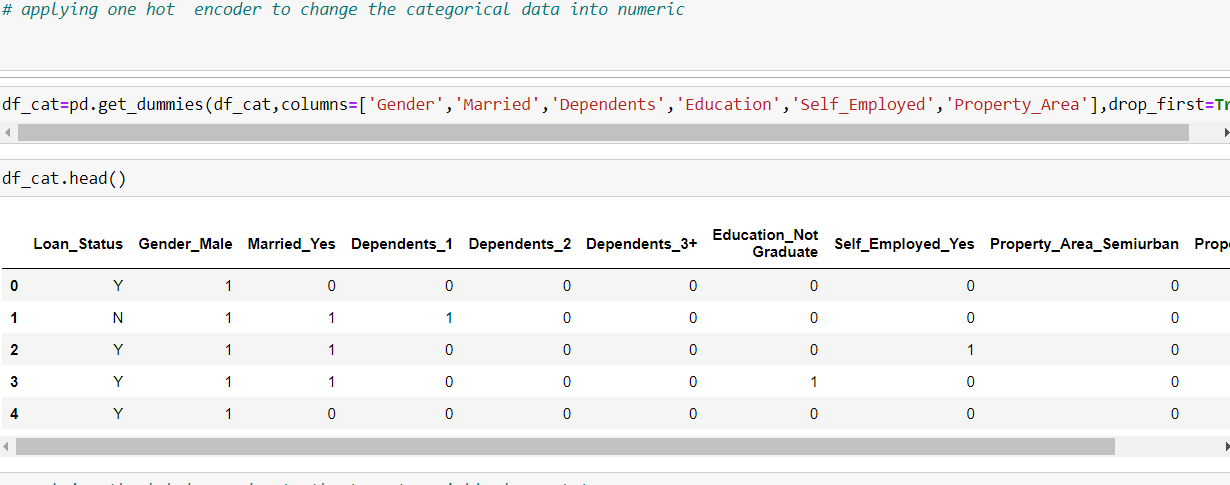


We can observe the data is right skewed ,so we can treat the skewness with log transformation to normalise the skewness.

df\_numeric['LoanAmount']=np.log(df\_numeric['LoanAmount'])

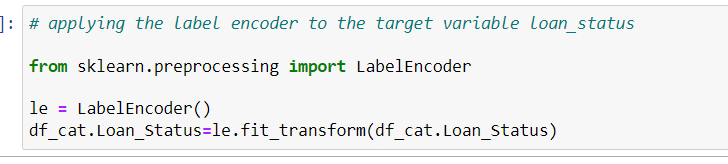


PREPROCESSING



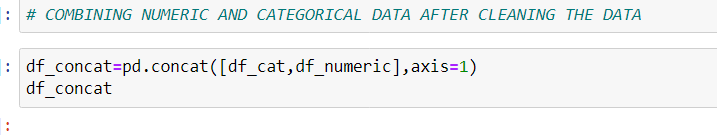
I have used one hot encoding to convert categorical data into numeric as it makes separate columns for each value in the column and model gives equal importance to all the values for accurate prediction.

On the other hand Label Encoder should be used for columns where the values are in descending order and one value has more weightage than the other as Label Encoder replaces values with number values 1,2,3,4,5 and so on, and the model end up being biased towards the greater value.

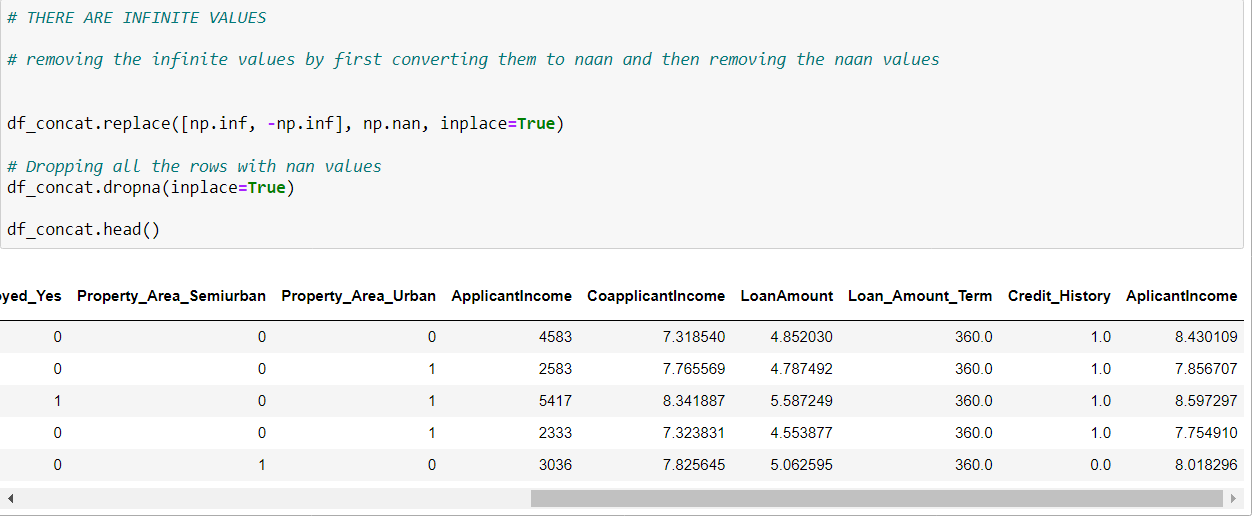


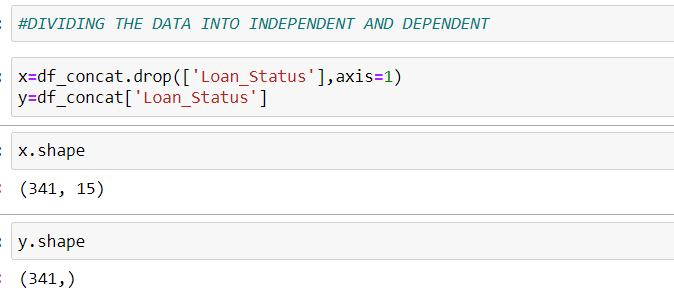
As target variable is a YES or NO so yes has more weightage than no ,therefore we should use Label Encoder.

As we have cleansed the data and done analysis now we can combined both the dataset again into one.



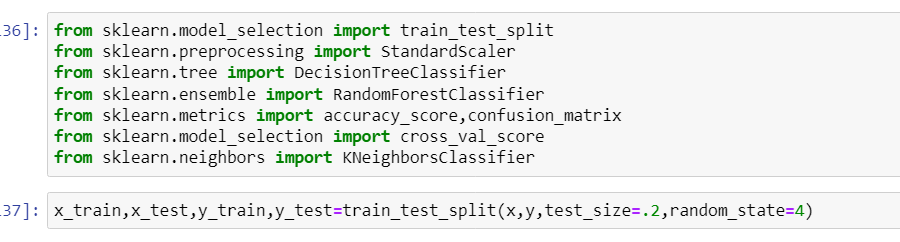
There is a possibility that after combining the data we might encounter infinite values ,We will convert the infinite values into naan and then remove the naan values.



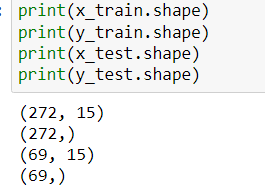


We will use the train\_test\_split function from sklearn to divide our train dataset. So, first, let us import train\_test\_split.

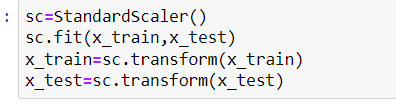
Import the required algorithms and packages



Check the shape of the splited data ,it should be same



Applying standard scaler is a good idea as this dataset has values in different units ,it will balance the data.



**MODEL BUILDING**

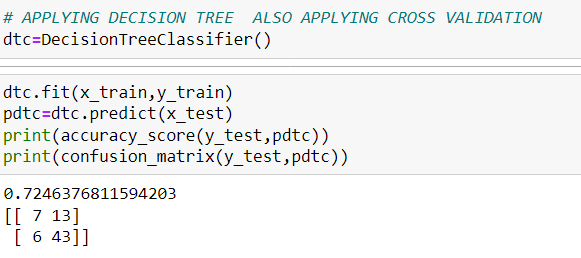
Now we are ready to apply algorithms ,as this is a classification problem so we will start with Decision tree classifier.

# Decision Tree

Decision tree is a type of supervised learning algorithm(having a pre-defined target variable) that is mostly used in classification problems. In this technique, we split the population or sample into two or more homogeneous sets(or sub-populations) based on the most significant splitter/differentiator in input variables.

Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. In other words, we can say that purity of the node increases with respect to the target variable.

For a detailed explanation visit <https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/#six>



# We can apply cross validation to make sure our model is not overfitting or underfitting

# 

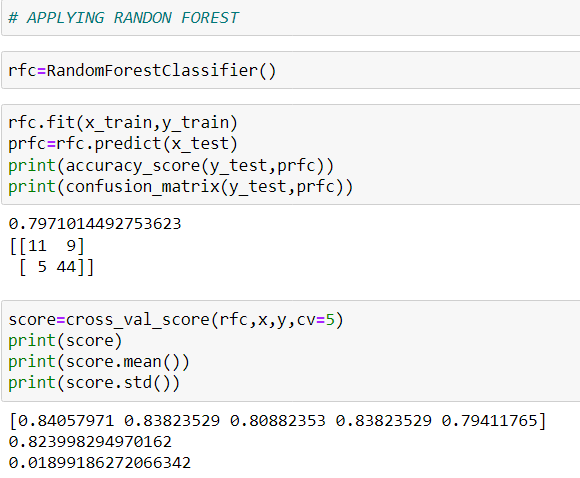
# Random Forest

RandomForest is a tree-based bootstrapping algorithm wherein a certain no. of weak learners (decision trees) are combined to make a powerful prediction model.

For every individual learner, a random sample of rows and a few randomly chosen variables are used to build a decision tree model.

Final prediction can be a function of all the predictions made by the individual learners.

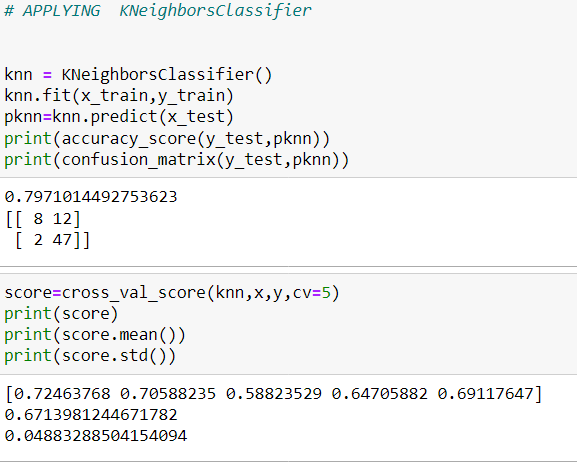
In the case of a regression problem, the final prediction can be the mean of all the predictions.



For a detailed explanation visit this article <https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/>

K-nearest neighbors

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry.



# \*CONCLUSION

MOST OF THE MALE CANDIDATES ARE GETTING LOAN APPROVALS AS COMPARE TO FEMALES EDUCATION IS PLAYING IMPORTANT ROLE IN LOAN APPROVALS ,GRADUATES ARE GETTING MORE LOANS AS COMPARE TO UNDERGRADUATE MOST MARRIED PEOPLE GETTING THE LOAN APPROVED PEOPLE WITH LESS NUMBER OF DEPENDENT ARE GETTING THE LOAN APPROVED SELF EMPLOYED PEOPLE ARE NOT GETTING GOOD RESPONSE WITH LOAN APPROVALS DATA HAD NULL VALUES ,THOSE HAVE BEEN HANDELED DATA HAS BEEN SCALED FOR BETTER PREDICTION RANDOM FOREST IS RESPONDING THE BEST