**Predicting Approval of Loan Application**



In this blog-post, I will go through the whole process of creating a machine learning model on the famous Loan Prediction dataset. It provides information on the status of loan application, summerised according to Gender, Marital Status, Dependents, Education, Applicant Income, Loan Amount, Credit History etc.

**Problem Statement:**

**About Company**   
Dream Housing Finance company deals in all home loans. They have presence across all urban, semi urban and rural areas. Customer first apply for home loan after that company validates the customer eligibility for loan.

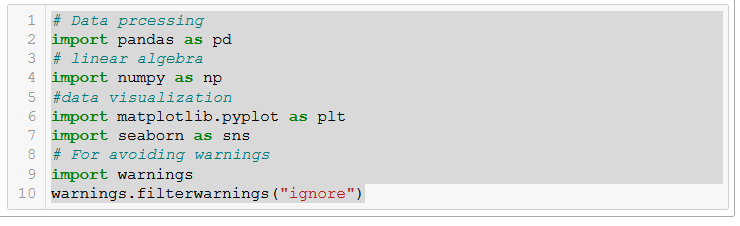
**Problem**   
Company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process, they have given a problem to identify the customers segments, those are eligible for loan amount so that they can specifically target these customers. Here they have provided a partial data set.

**Dataset Description:**

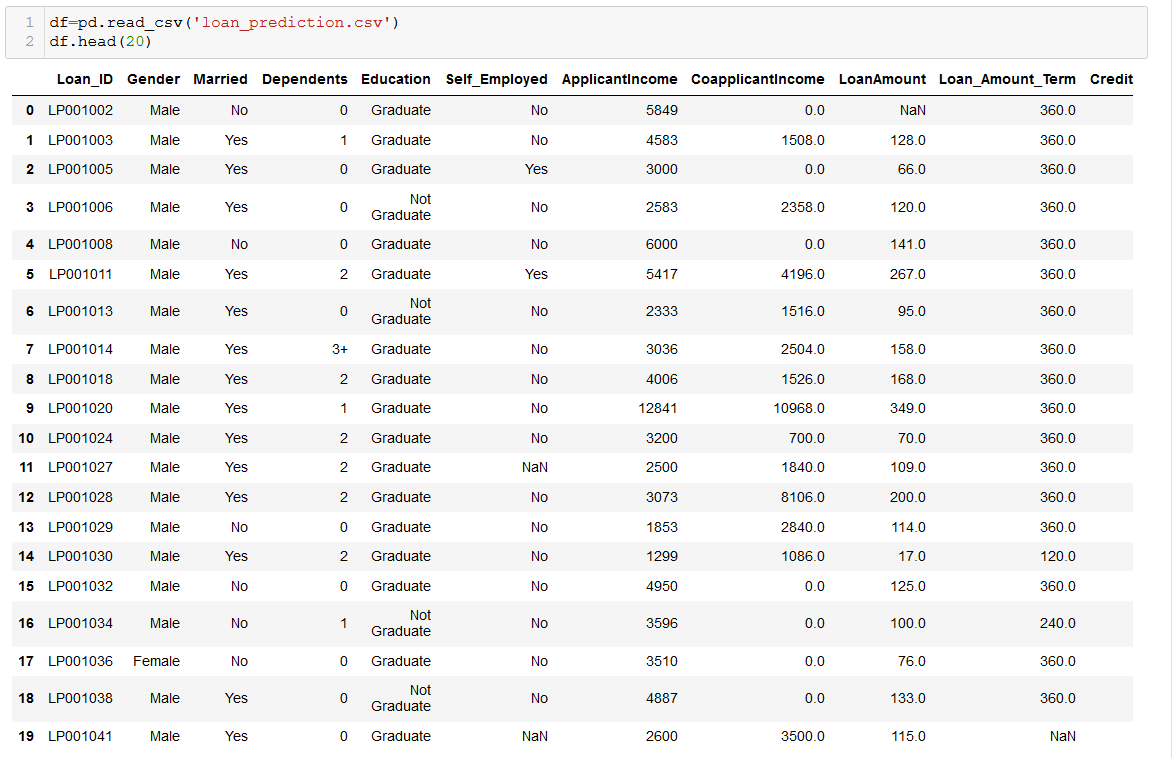
|  |  |
| --- | --- |
| **Variable** | **Description** |
| **Loan\_ID** | **Unique Loan ID** |
| **Gender** | **Male/Female** |
| **Married** | **Applicant married (Yes/No)** |
| **Dependents** | **Number of dependents** |
| **Education** | **Applicant Education (Grad/ U. Grad)** |
| **Self\_Employed** | **Self employed (Yes /No)** |
| **ApplicantIncome** | **Applicant income** |
| **CoapplicantIncome** | **Coapplicant income** |
| **LoanAmount** | **Loan amound in thousands** |
| **Loan\_Amount\_Term** | **Term of loan in months** |
| **Credit\_history** | **Credit history meets guidelines** |
| **Property\_Area** | **Urban/ Semi Urban/ Rural** |
| **Loan\_Status** | **Loan approved (Yes / No)** |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Importing the Libraries

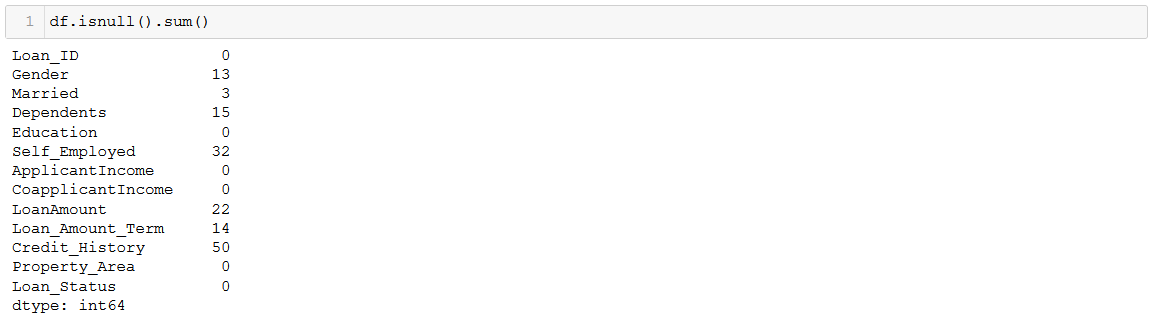
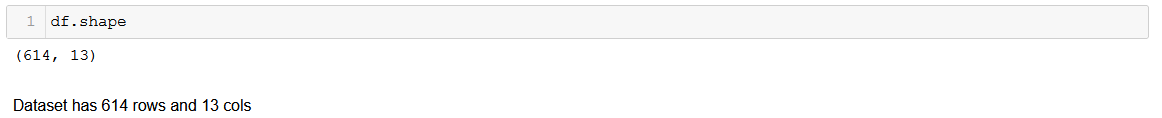


Getting the Data



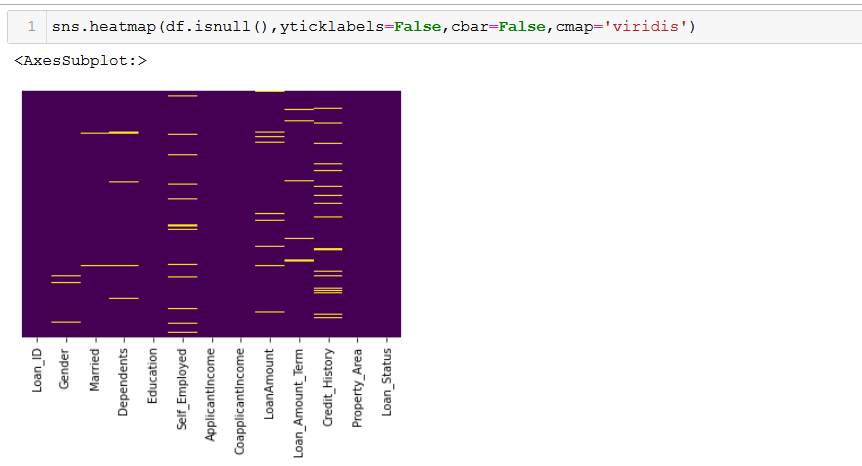
Data Preprocessing

Stage 1 : Replacement Missing values

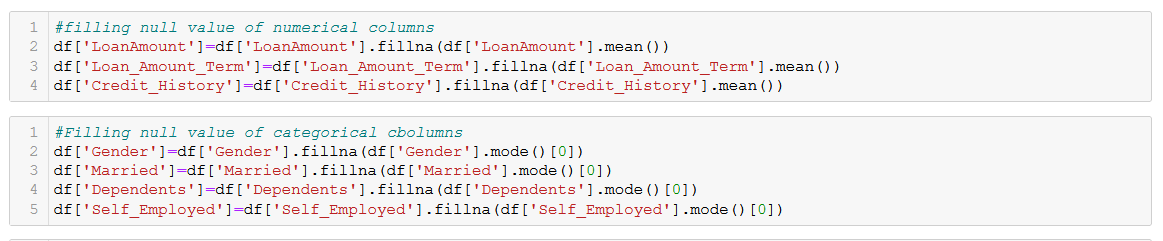


The data set has 614 examples and 12 features + the target variable. There missing values in few of the features. They are as follows

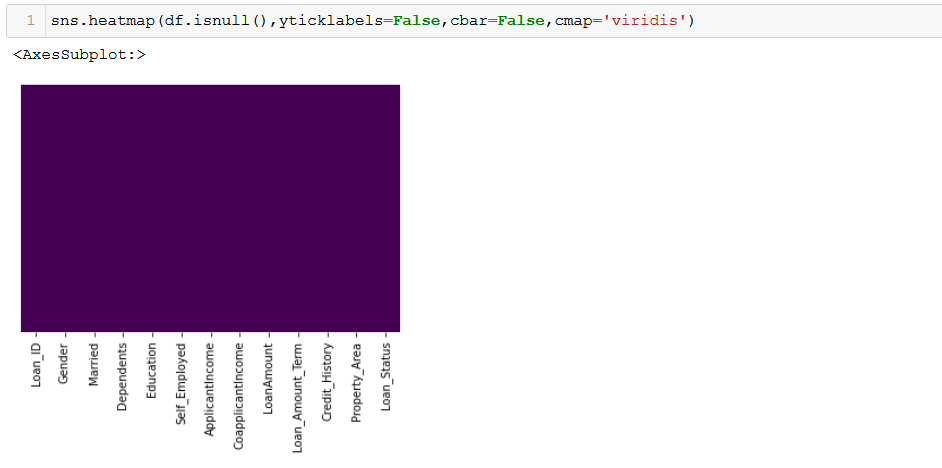
|  |  |
| --- | --- |
| Gender | 13 |
| Married | 3 |
| Dependents | 15 |
| Self\_Employed | 32 |
| LoanAmount | 22 |
| Loan\_Amount\_Team | 14 |
| Credit\_History | 50 |



We will fill/impute the null values of the dataset. For integer type of features we will use the Mean of the feature and for categorical data we will use Mode of the Feature.

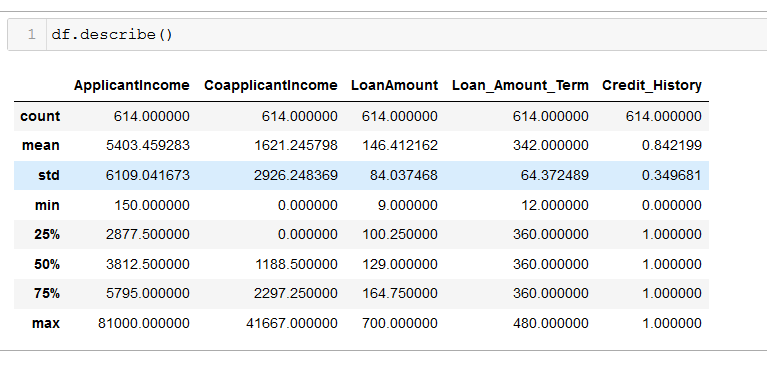


So we have replaced the null values using pandas mean() and mode() function.



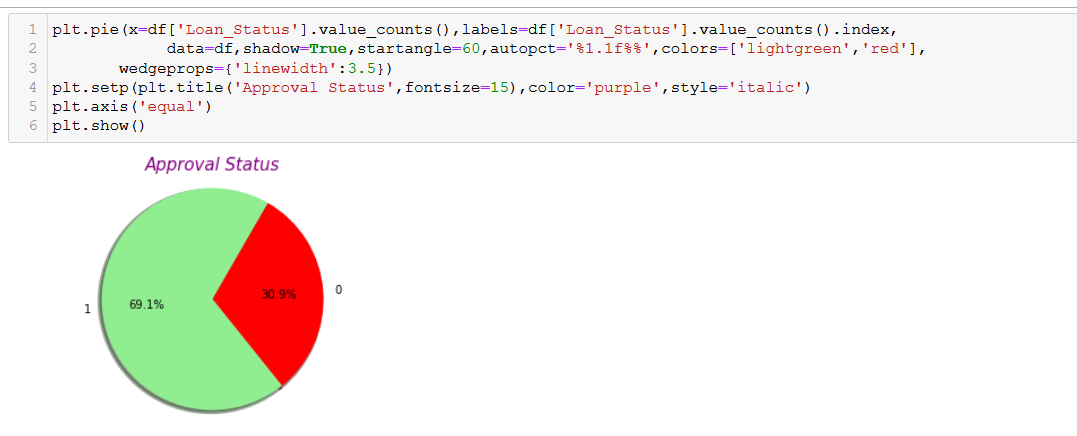
After imputing we found all the null values have been filled.

**Exploratory Data Analysis / Remarks**

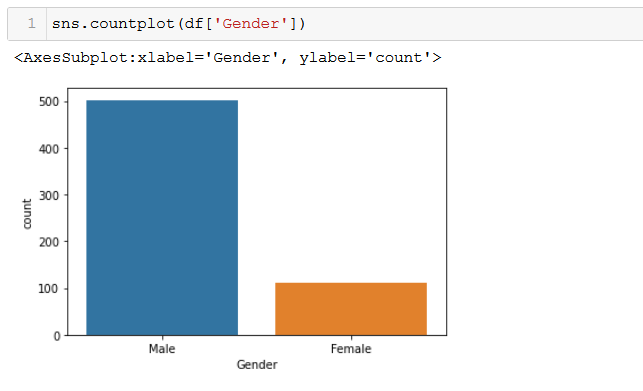


The describe function shows that average applicant income and loan amount is $ 5403

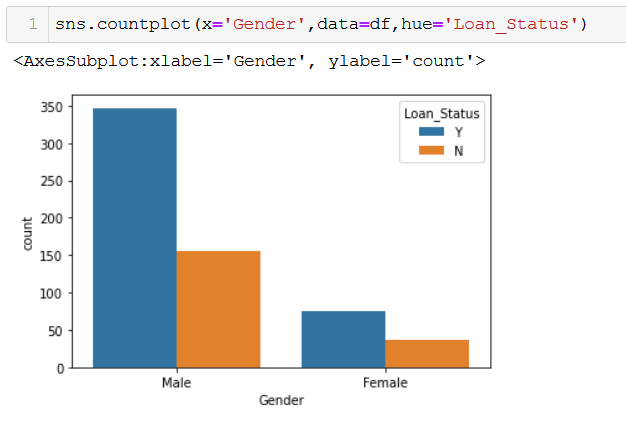
and $ 146 respectively. The Maximum loan amount is $ 700.



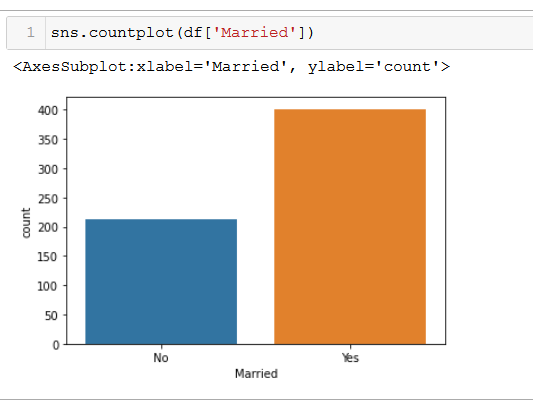
Using matplotlib pie function we came to know that 69.1 % loan approved and 30.9 % loan has been rejected.



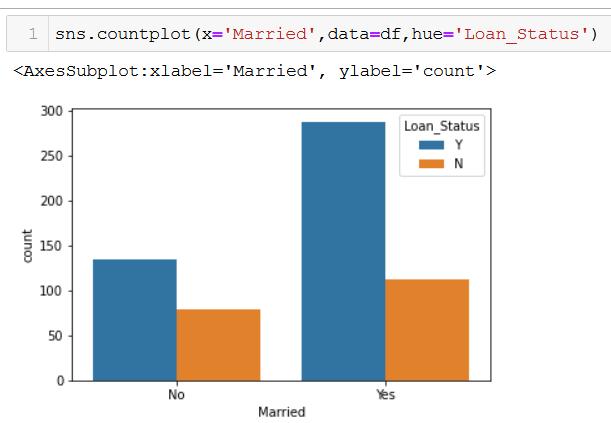
Male applicant are much more than female applicant.



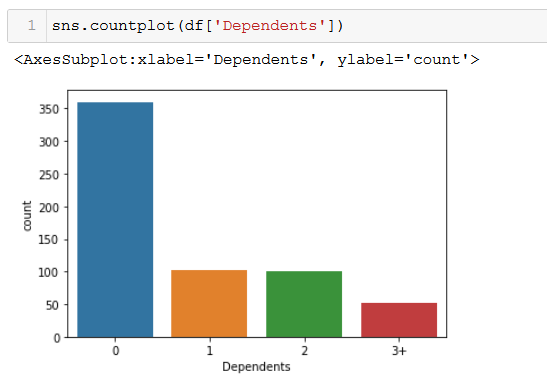
Male applicant has higher rate of approval



Majority of applicant are married.



Married applicant has higher rate of approval.



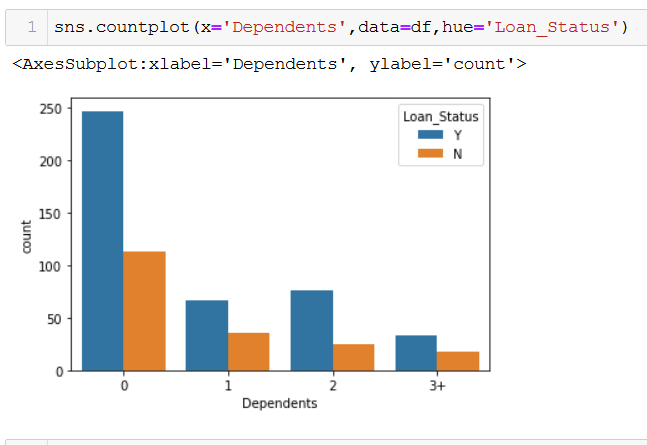
Using the above visualization we observed the following points

(i) Most of the applicants has no dependents

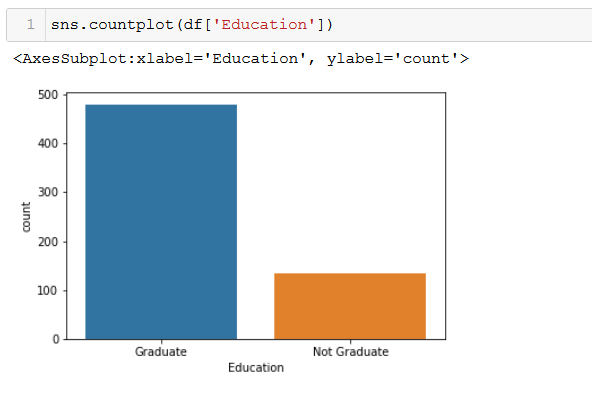
(ii) Around 100 applicants has 1 dependents and same counts is applicable for

2 dependents

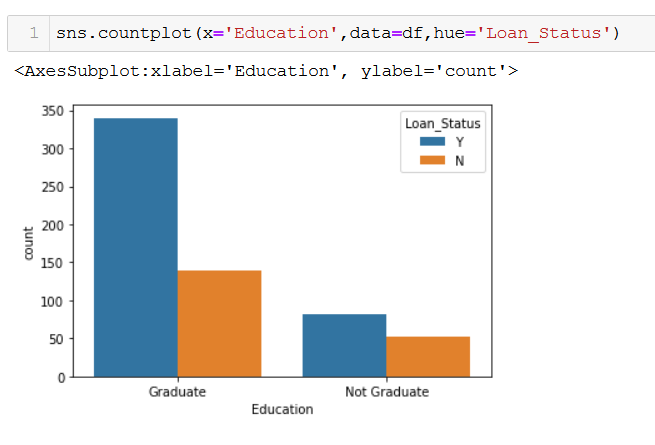
(iii) Atmost 50 applicants has 3+ dependents



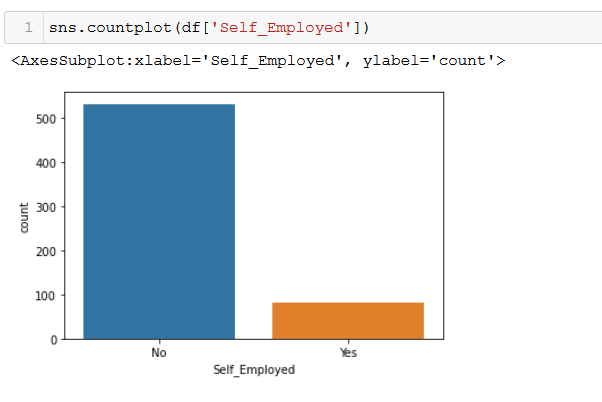
Without dependents are more preferable



**Majority of the people are graduate and 130 applicants are Not Graduate.**



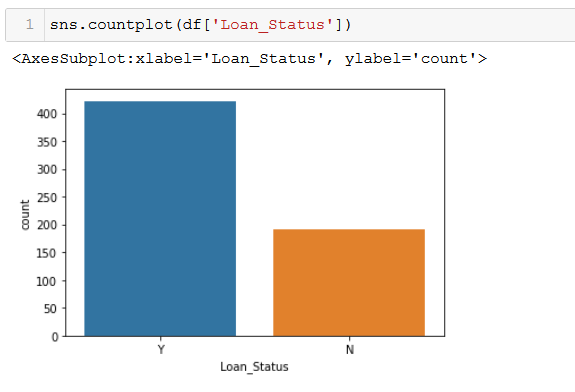
Graduate applicant has higher approval rate.



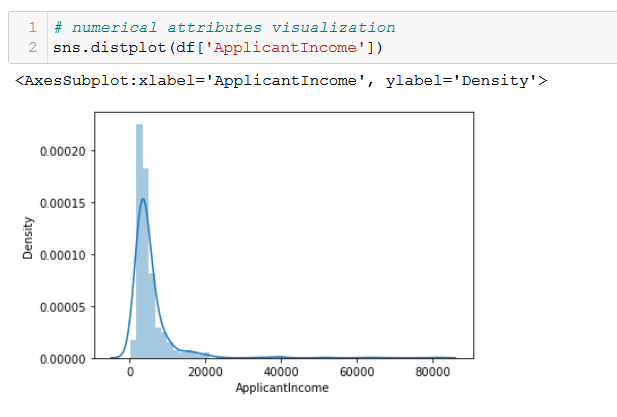
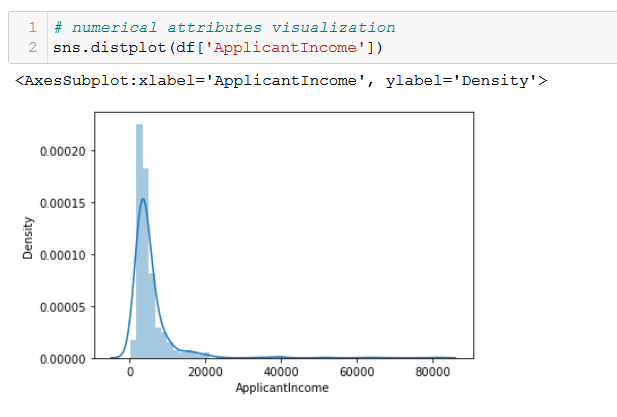
Maximum loan applicant are not Self Employed.



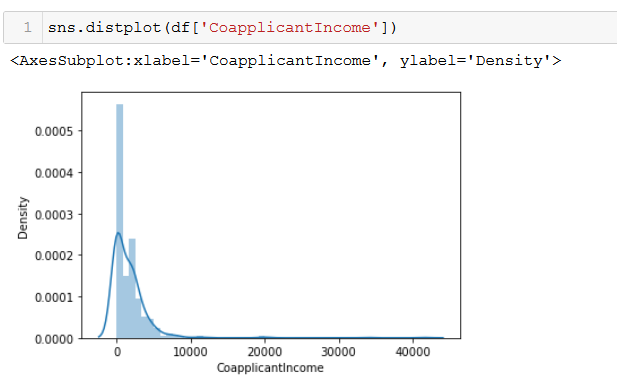
Maximum loan applicant are not Self Employed.



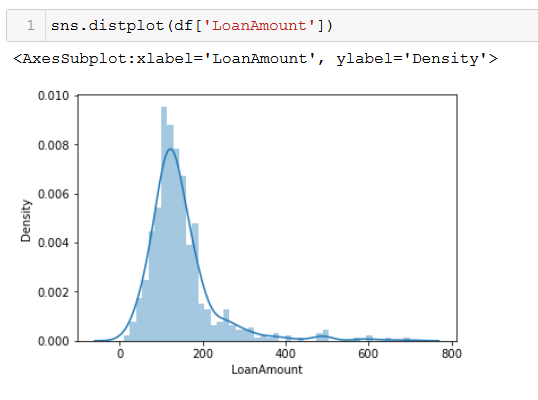
The approval status of loan is at 2:1 ratio, 2- Approved, 1- Not Approved.



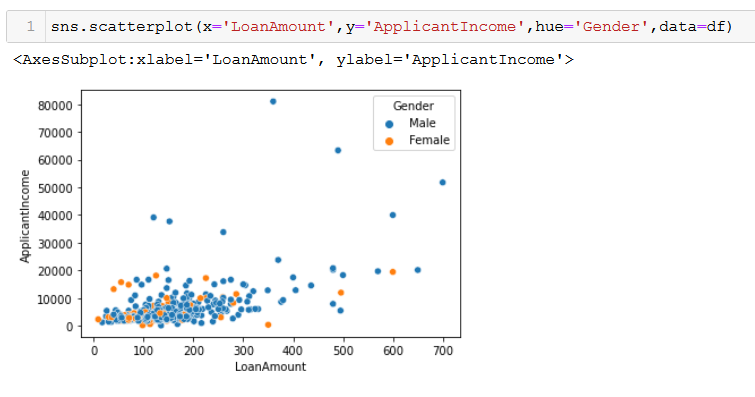
Major of the applicant income in between 0 to 20,000/- very few are above 20000



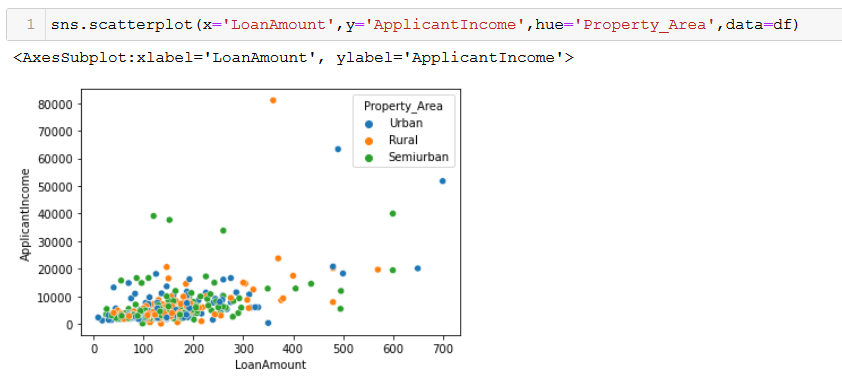
Coapplicant has imcome 0-10000 in major cases



Maximum loan amount range in 10-200 $.



If applicant income is more then loan amount is also more.



In almost all area the loan amount is equally distributed



The observation found from above visualisation are listed below:

(i) Married Male and Female has more probability of approved loan.

(ii) Male applicant are higher than female applicant

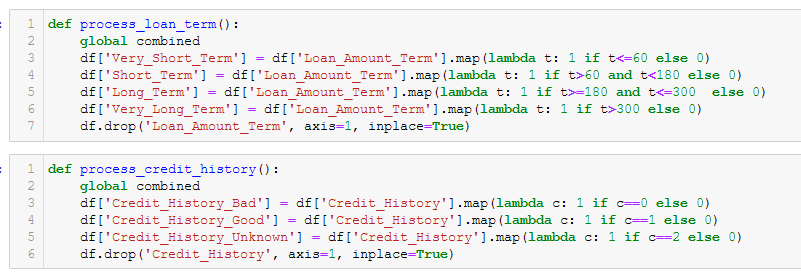
(iii) There are more 0 Dependents applicant

(iv) Graduate applicants numbers are more than ungraduate in approved loan

(v) Applicants with 1.0 Credit history has highest loan approval

**Creating new Features:**

1. Total Income of applicant



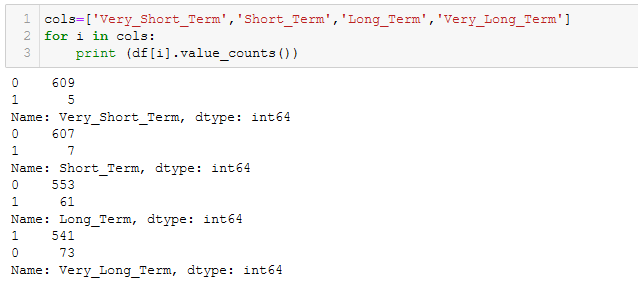
1. Very Short Term Loan amount
2. Short Term Loan amount
3. Long Term Loan amount
4. Very Long Term amount

Dropped the Loan\_Amount\_Term features as we have created 4 new features using the same.

1. Cedit History Bad
2. Credit History Good
3. Credit History Unknown

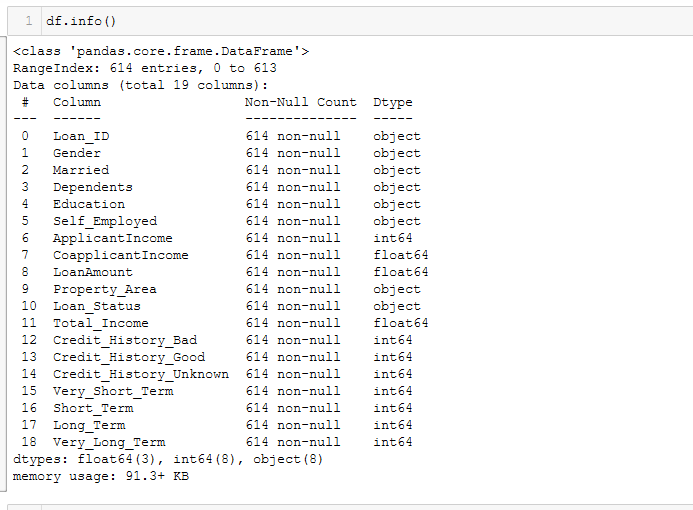
Dropped the Credit history features as we have created 3 new features using





(i) Very Short term loan applicants are less in counts 5

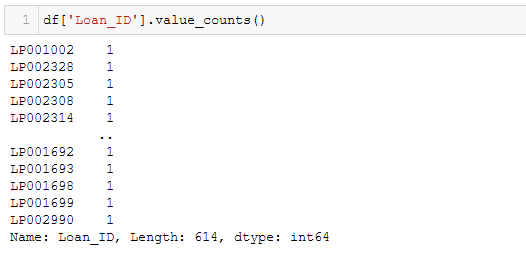
(ii) Maximum are very long term applicant that is 541



Using the pandas info() function we can observe that there dew object/String type of data which have to encoded into integer for correlation and Model Building.

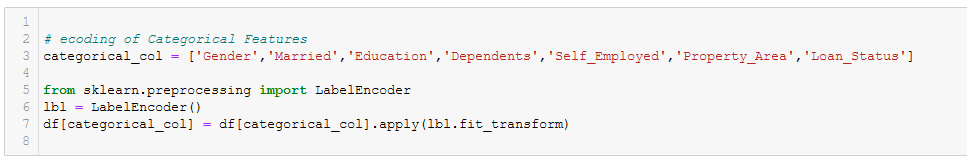
Date Preprocessing Pipelines:

Stage 2: Dropping of unwanted feature

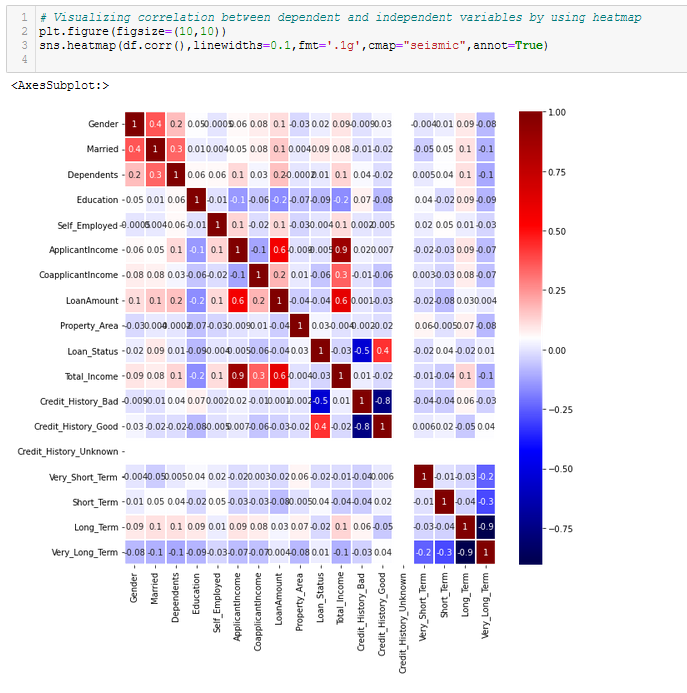


We found the Load\_ID column is not so important considered to application approval status. So we dropped it.

Stage 3 : Encoding of data



The object/ categorical features are almost in ordinal category so we have used the Label Encoder to encode the features.

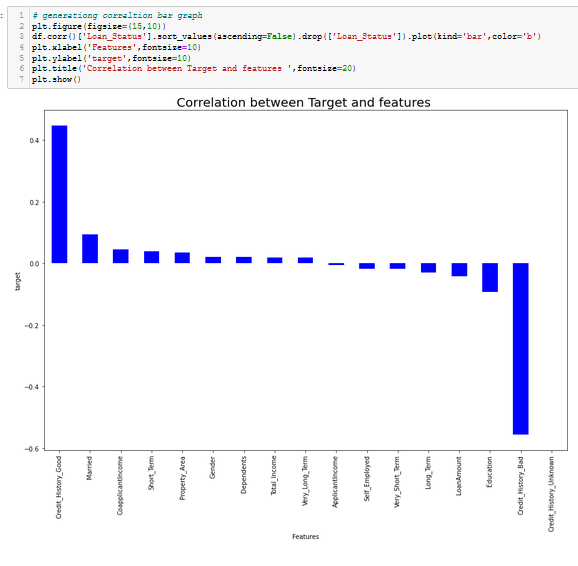


We found the below mentioned observation from the visualization using correlation

(i) Loan Status is positively correlated with Good Credit History

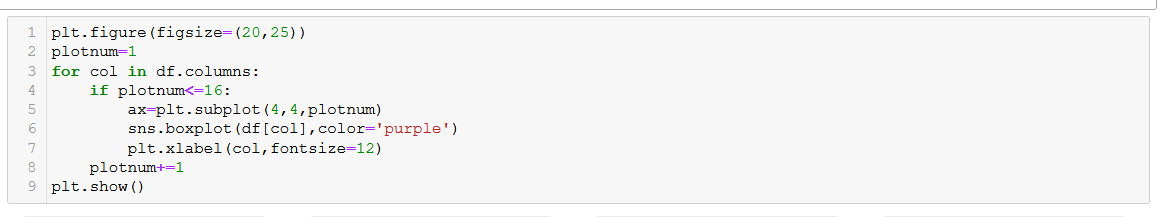
and Negatively Correlated with Bad Credit History

(ii) Loan Amount is highly correlated with Total Income

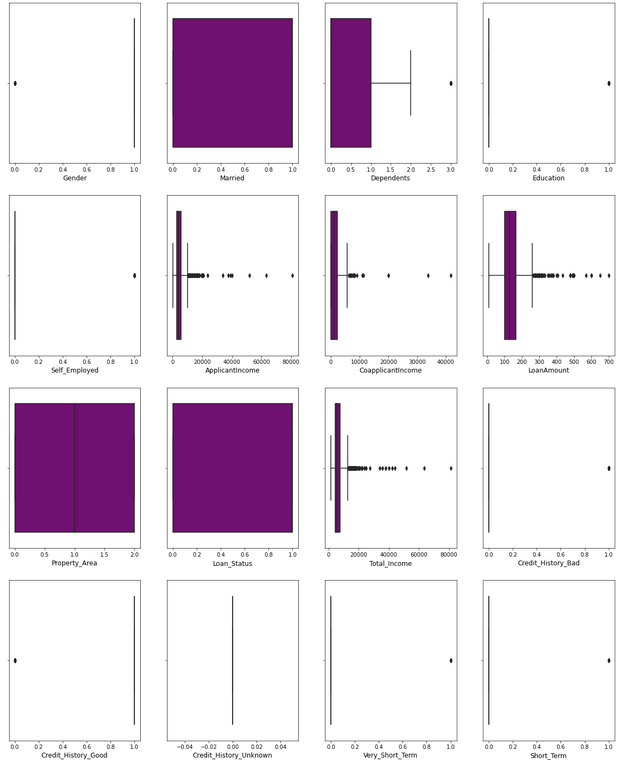


We found the same observation as earlier using the correlation score based on target values.

Stage 4: Removing of outliers

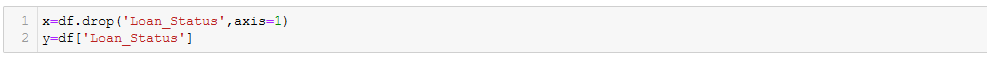
****

We have removed the outliers using Z-Score threshold value of 3. The loss of data is only 4%. Which is acceptable. The screenshot is given below.

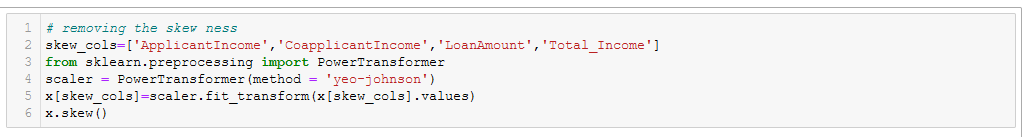
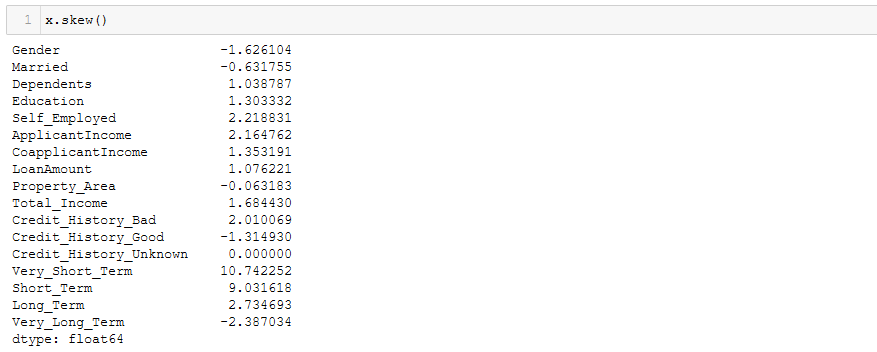




Stage 5 : Splitting of feature and Target

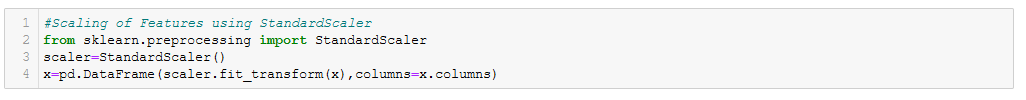


Stage 6 : Removing the skewness and normalising



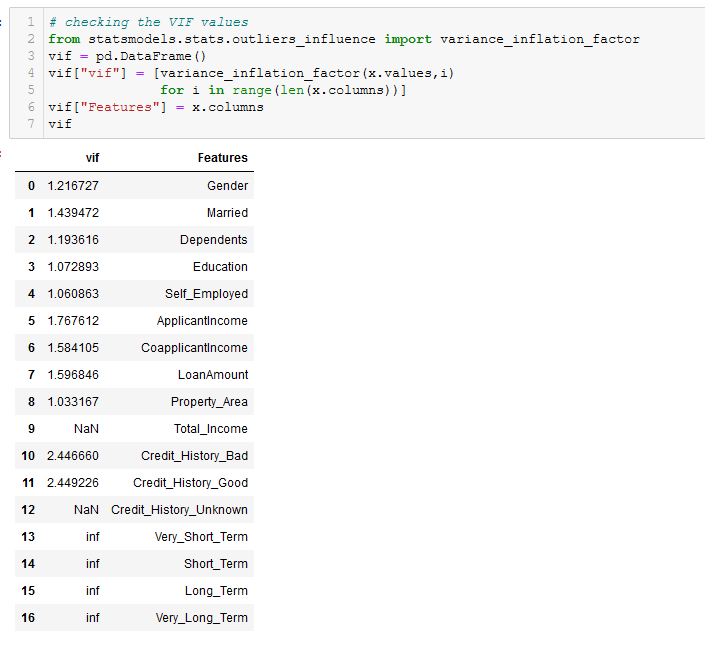
We have considered the integer type of data for skewness removal. Removed the skewness using yeo-johnson method. Positive points of yeo-johnson is it remove the skewness if at all there is availability of negative/ zero values in the features.

Stage 7:



Scaled the features for best performance of the model using the sklearn standard scaler.

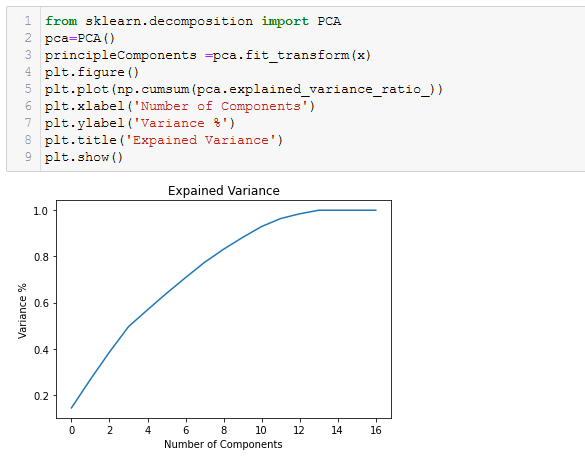
Stage 8: Checking for availability of Multicollenearity



Multicollinearity occurs when two or more independent variables are highly correlated with one another.

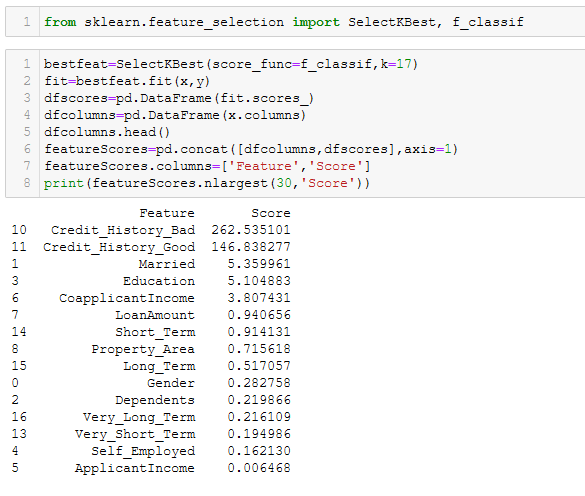
All the VIF values are less than 10 so we can assume, there is no multicollinearity.

Stage 9 : Determining number of important features using Principle Component Analysis



14 columns are sufficient for prediction with 98% accuracy.

Stage 10 : Selection of best features using KBest Score

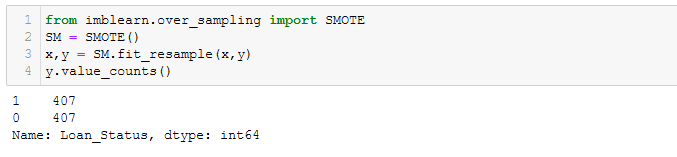


We will continue with out dropping any feature as the number of features are less

Stage 11 : Target class imbalance

An imbalanced classification problem is an example of a classification problem where the distribution of examples across the known classes is biased or skewed. The distribution can vary from a slight bias to a severe imbalance where there is one example in the minority class for hundreds, thousands, or millions of examples in the majority class or classes.

Imbalanced classifications pose a challenge for predictive modeling as most of the machine learning algorithms used for classification were designed around the assumption of an equal number of examples for each class. This results in models that have poor predictive performance, specifically for the minority class. This is a problem because typically, the minority class is more important and therefore the problem is more sensitive to classification errors for the minority class than the majority class.

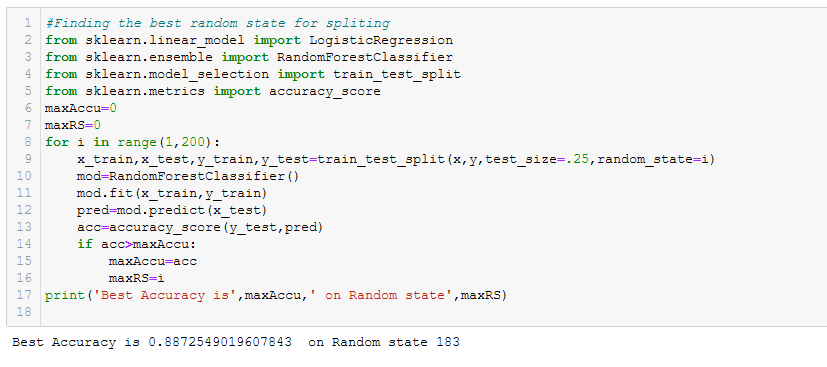


We have balanced the target using Synthetic Minority Over-sampling Technique(SMOTE).

**Building Machine Learning Model**

As the target value is not continuous in nature so this can be considered as classification problem.

Stage 1 : Determining the best random state for splitting of train and test dataset



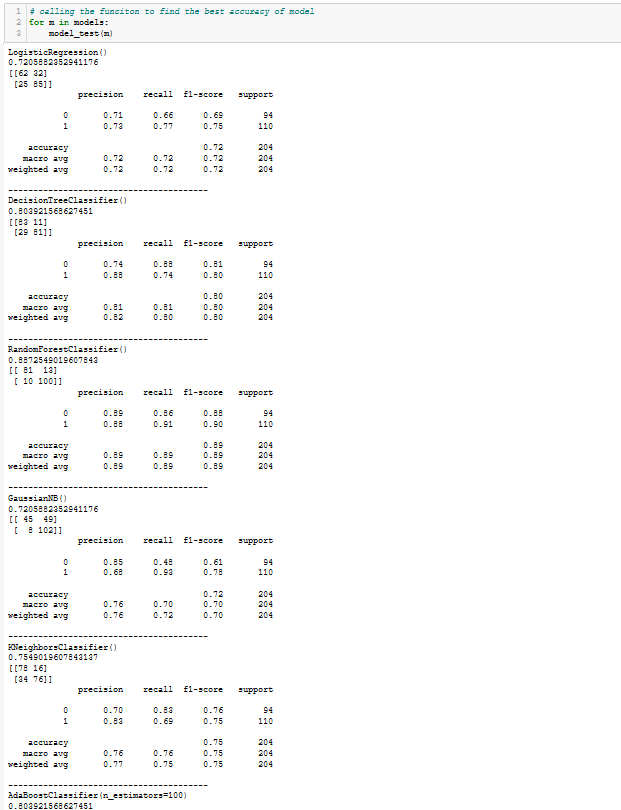
We have found that the 88% accuracy can be achieved using random state 183.

Stage 2 : Loading of classification models





Stage 3 : Testing the accuracy of each model

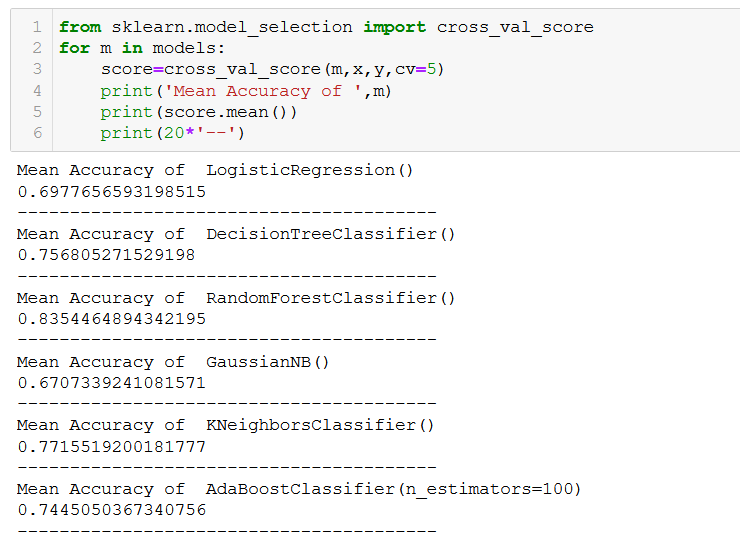


We found the Random forest model is best model which is performing with 88% accuracy with the dataset.

Stage 4 : Cross validation

Cross-validation is a statistical method used to estimate the skill of machine learning models.

It is used in applied machine learning to compare and select a model for a given predictive modeling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.



As per cross validation score also the highest score is from Random Forest Classifier

is the best model for building.

## What is Random Forest ?

Random Forest is a supervised learning algorithm. Like you can already see from it’s name, it creates a forest and makes it somehow random. The „forest“ it builds, is an ensemble of Decision Trees, most of the time trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

To say it in simple words: Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

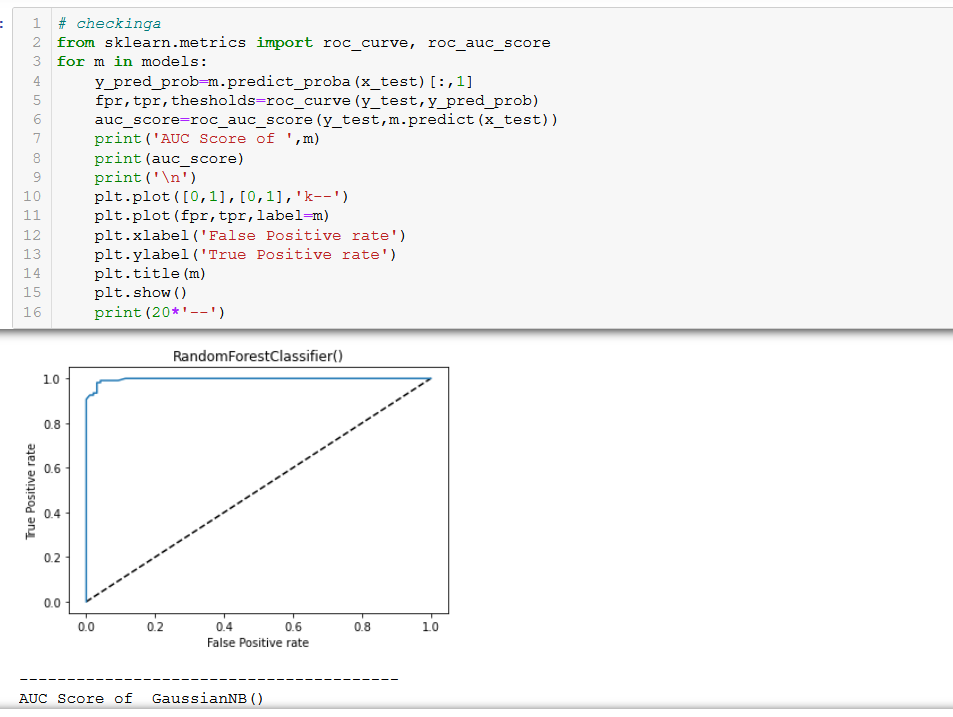
One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems. With a few exceptions a random-forest classifier has all the hyperparameters of a decision-tree classifier and also all the hyperparameters of a bagging classifier, to control the ensemble itself.

The random-forest algorithm brings extra randomness into the model, when it is growing the trees. Instead of searching for the best feature while splitting a node, it searches for the best feature among a random subset of features. This process creates a wide diversity, which generally results in a better model. Therefore when you are growing a tree in random forest, only a random subset of the features is considered for splitting a node. You can even make trees more random, by using random thresholds on top of it, for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

Stage 5 : Checking AUC RUC Curve

The **Receiver Operator Characteristic (ROC)** curve is an evaluation metric for binary classification problems. It is a probability curve that plots the **TPR** against **FPR** at various threshold values and essentially **separates the ‘signal’ from the ‘noise’**. 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 curve.

The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.



As per AUC, ROC score the Random State Classifier has covered 96 % training

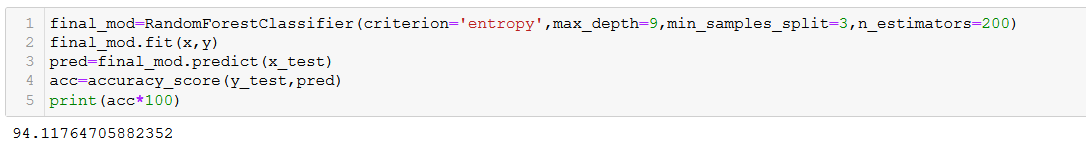
Stage 6 : Hyper Parameter Tuning

Below you can see the code of the hyperparamter tuning for the parameters criterion, min\_samples\_leaf, min\_samples\_split and n\_estimators.



Now that we have a proper model, we can start evaluating it’s performace in a more accurate way.

Stage 6 : Saving the model



We have saved the model for future usage and implementation. The final model is performing at 94% accuracy, earlier it was 88% . After Hyper Tuning process the model has increased its accuracy impressive way.

# Summary

We started with the data exploration where we got a feeling for the dataset, checked about missing data and learned which features are important. During this process we used seaborn and matplotlib to do the visualizations. We made important business remarks based on the visualization. During the data preprocessing part, we computed missing values, converted features into numeric ones, grouped values into categories and created a few new features. Afterwards we started training 6 different machine learning models, picked one of them (random forest) and applied cross validation on it. Then we discussed how random forest works, took a look at the importance it assigns to the different features and tuned it’s performance through optimizing it’s hyperparameter values. Finally we saved the model for future usage.

Concluding Remarks:

The loan approval chances is more in following cases:

1. Applicant income is high
2. Coapplicant income is there
3. Credit score is high
4. Applicant is male
5. Applicant is married
6. Applicant don’t have dependents
7. Applicant is Self employed
8. Applicant belong to rural property area
9. Applicant is Graduate
10. Applicant belongs to rural property area