**Loan Application Status Prediction**

**Author:**

**Rupali Bisen**

**Blog on Loan Application Status Prediction**

**Problem Definition:**

This data-set includes details of applicants who have applied for loan. The data-set includes details like credit history, loan amount, their income, dependents etc. Information for the data-set is taken from past clients who had their loan status approved/not-approved. Depending on past experiences model will be prepared to determine precise outcome. Following are the variables present in the data-set:

Independent Variables:

1. Loan\_ID
2. Gender
3. Married
4. Dependents
5. Education
6. Self\_Employed
7. Applicant Income
8. Co-applicant Income
9. Loan\_Amount
10. Loan\_Amount\_Term
11. Credit History
12. Property\_Area

Dependent Variable (Target Variable):

1. Loan\_Status

The principle objective is 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 data-set.

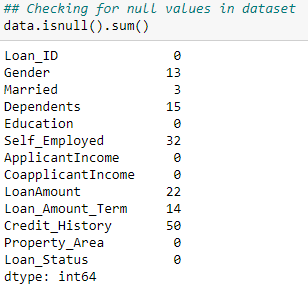
**Data Analysis:**

Firstly, load the data-set into jupyter notebook using pandas.read\_csv function. Loan application data-set has 614 rows and 13 columns. On checking for data types of attributes, the Data-set can be divided into two types:

1. Nominal/Categorical data: ['Gender', 'Married', 'Dependents', 'Education', 'Self\_Employed', 'Property\_Area', 'Loan\_Status', 'Credit\_History']

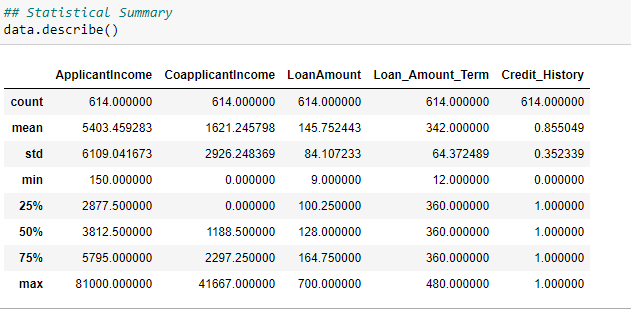
2. Ordinal/numerical data : ['Applicant\_Income','Coapplicant\_Income','Loan\_Amount','Loan\_Amount\_Term'].

On Checking for null values/ missing values in the data-set it was observed that data set has many null values .



Data-set has so many NaN values which needs to be replaced. Also, data-set has many categorical variables which needs to be converted into numeric so that machine learning algorithms can process them. Furthermore, we can see that the features have widely different ranges, that we will need to convert into roughly the same scale.

Statistics is the discipline that concerns the collection, organization, analysis, interpretation, and presentation of data.



On examining statistical summary of the data, following observations can be made:

The training set has 614 examples and 12 attributes plus 1 target variable. 8 attributes are object type, 4 are float type and 1 integer type.

1. After filling null values now total count of each attribute is 614.

2. Applicant's minimum income is 150 and maximum income is 81000.

3. Most of the coapplicants are not working. maximum income of coapplicant is 41667.

4. Minimum loan amount is 9 and maximum loan amount is 700.

5. Loan Amount term is minimum for 12 months and maximum for 480 months.

6. There is large difference in maximum and 75% value which implies data-set has many outliers.

7. Mean and median of data set also shows large variation which shows data is not normally distributed.

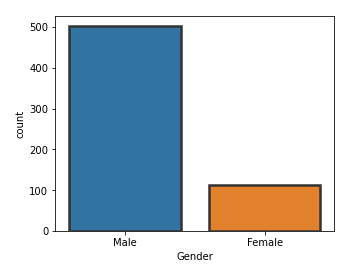
#### **Exploratory data analysis for Nominal/categorical type of data:**

**Univariate Analysis:**

Nominal/Categorical data: ['Gender', 'Married', 'Dependents', 'Education', 'Self\_Employed', 'Property\_Area', 'Loan\_Status', 'Credit\_History']

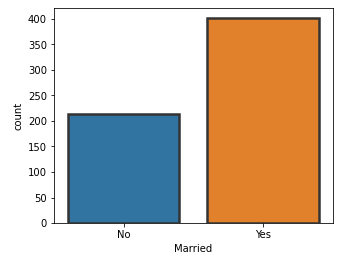
1. **Gender**



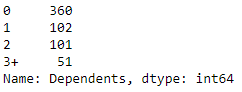


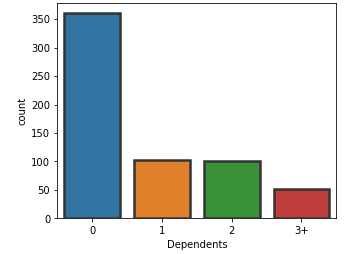
1. **Married**





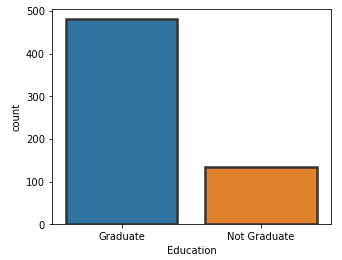
1. **Dependents**





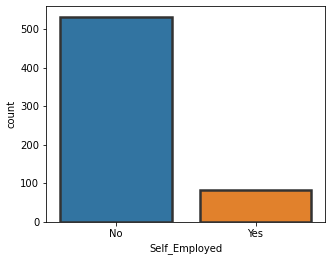
1. **Education**



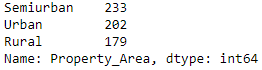


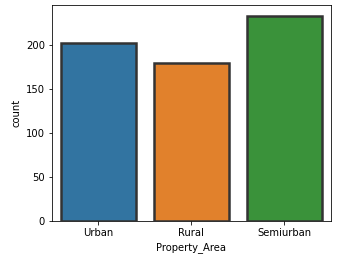
1. **Self\_Employed**





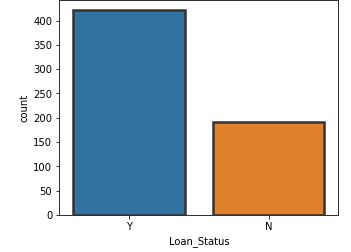
1. **Property\_Area**





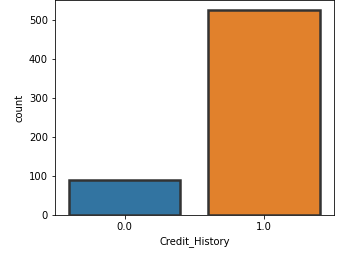
1. **Loan\_Status**





1. **Credit\_History**





Observations:

1. Data-set has more number of male applicants than female. Male applicants are around 82% compared to female applicants.

2. Most of the applicants are married. 65% more married applicants have applied for loan.

3. Applicants having 0 dependents are more. Whereas applicants having 3+ dependents are least of all.

4. About education background, around 78% of applicants are graduate.

5. 87% of applicants applied for loan are self employed.

6. Most of the applicants have property area in Semi urban region.

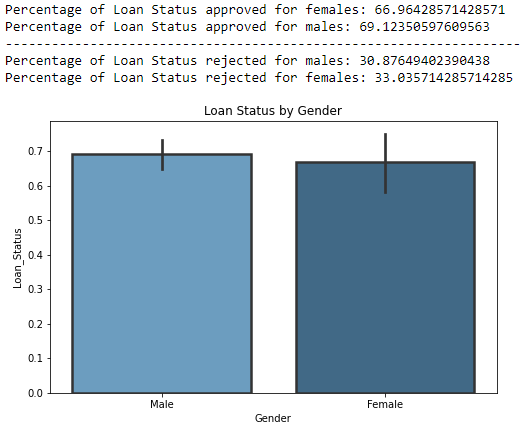
7. 38% applicants have property in semi urban region, 33% in Urban and 29% in Rural Region.

8. 85.5% applicants who have applied have credit history. Rest 14.5% are with no credit history.

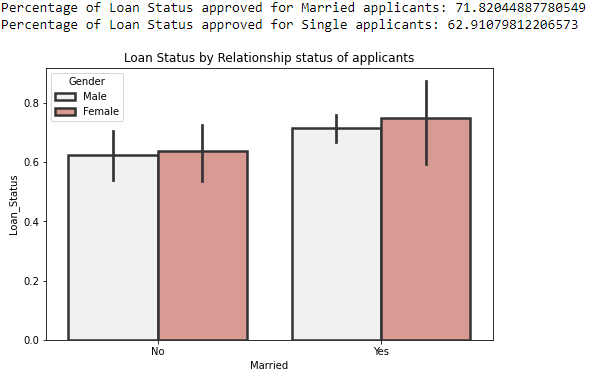
9. Target variable(Loan\_status) is unbalanced.

**Bivariate Analysis:**

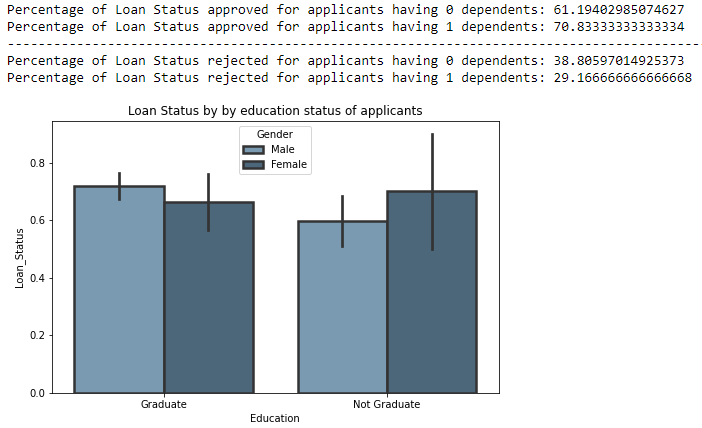
1. **Visualization of bar plot of Loan Status by Gender:**



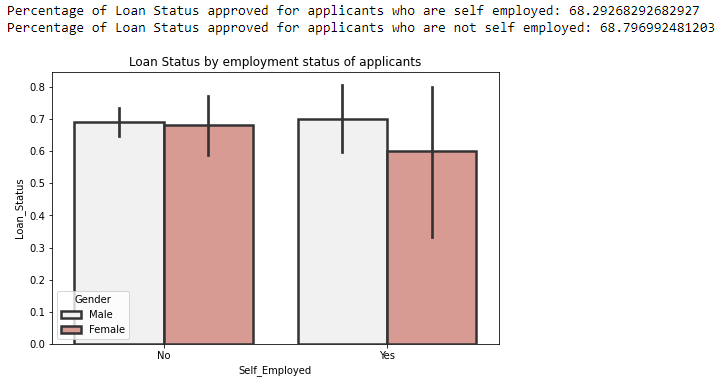
1. **Visualization of bar plot of Loan Status by relationship status:**



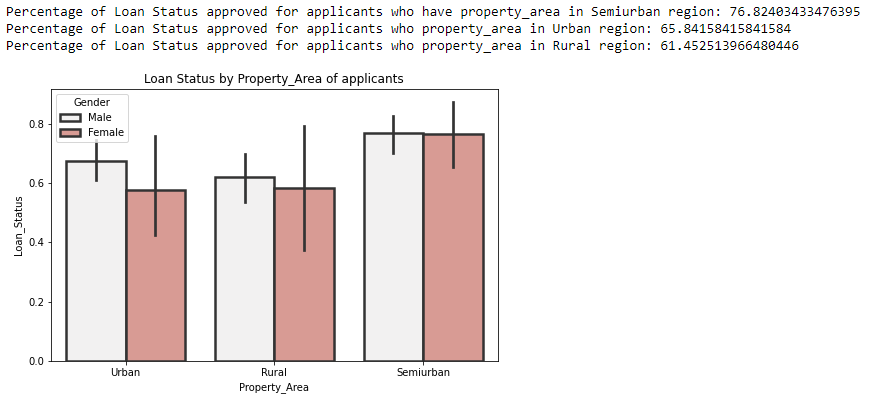
1. **Visualization of bar plot of Loan Status by education status of applicants.**



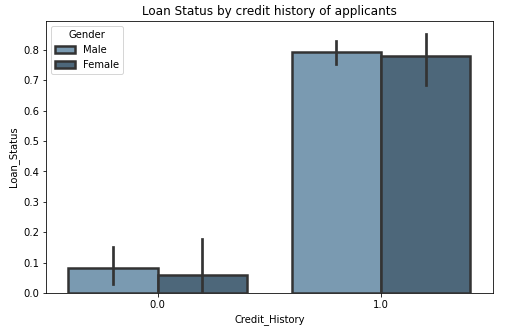
1. **Visualization of bar plot of Loan Status by employment status of applicants.**



1. **Visualization of bar plot of Loan Status by Property\_Area of applicants.**

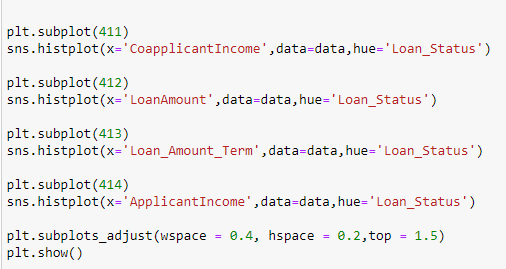


1. **Visualization of bar plot of Loan Status by credit history of applicants.**

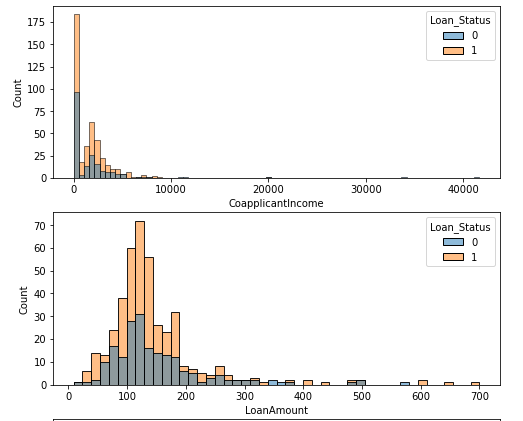


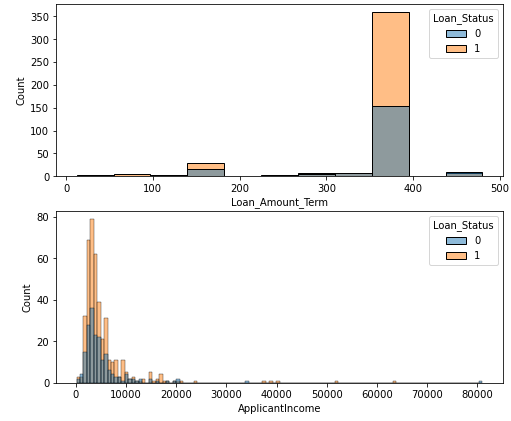
#### **Exploratory data analysis for Ordinal/numerical type of data:**

Ordinal/numerical data :['Applicant Income','Coapplicant Income','Loan Amount','Loan\_Amount\_Term']



A histogram is a great tool for quickly assessing a [probability distribution](https://en.wikipedia.org/wiki/Probability_distribution) that is intuitively understood by almost any audience. Histogram plot for numeric data is shown below:

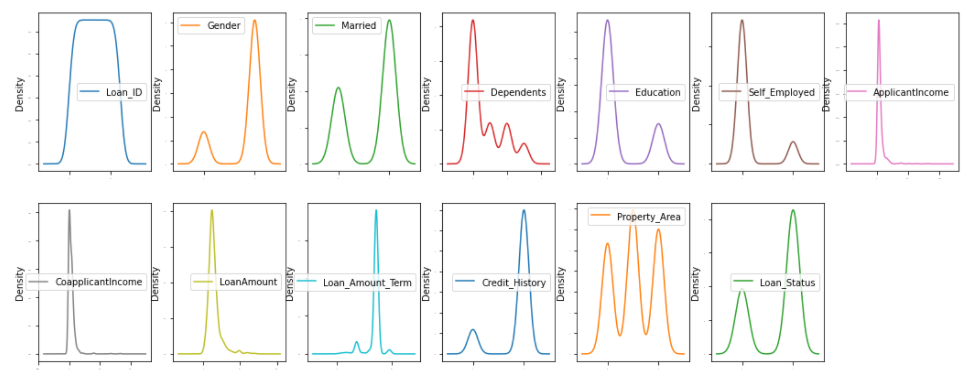




Above histogram plot shows that most of the applicants having income between 2000 to 7000 have got their loan status approved. Also applicants having less loan amount have higher chances of getting their loan approved.

**Density visualization for all attributes:**



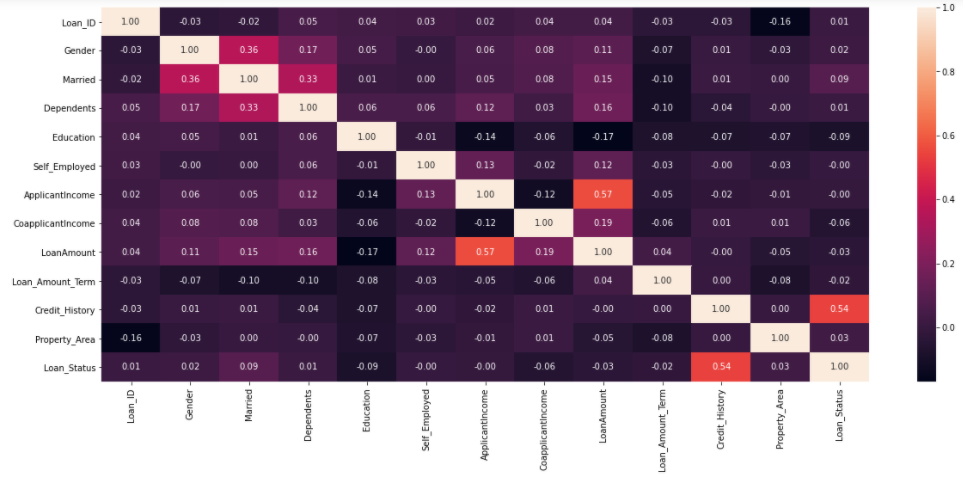


From above density plot, skewness can be observed in most of the attributes.

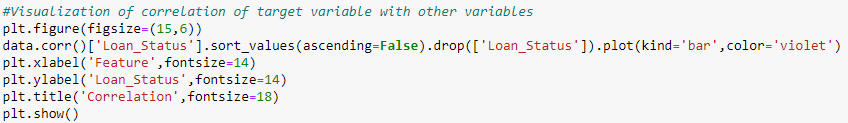
Hence checking for skewness and removing the same in data Pre-Processing.

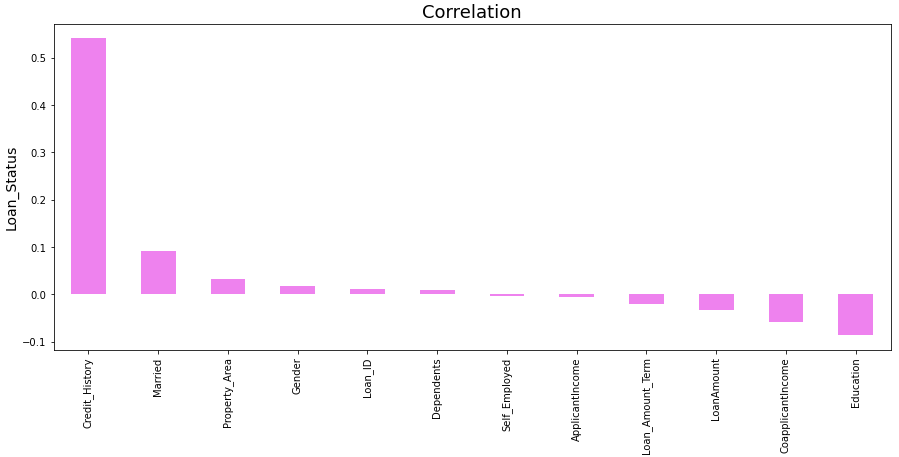
**Checking for correlation of output variable with other attributes:**





Sorting attributes based on correlation with target variable. Sorting values gives more clear idea in a simplified manner on how independent variables correlate with dependent variable. Also visualization of the same makes the picture very crystal clear.





Loan\_status has highest correlation with Credit\_History.

Loan\_Status has very less correlation with Self\_Employed.

**EDA Concluding Remark:**

Loan status approval for male and females are nearly equal. For males it is 69.32% and for females is 66.96%.

71% of married applicants have their loan status approved. More of the female married applicants have their loan status approved as compared to male married applicants. Around 70% applicants who are graduate got their loan status approved.

61% applicants who are not graduate got their loan approved. Irrespective of employment, loan is approved for applicants. Most of the male applicants who are employed get their applications approved for loan.

Most of the applications are approved for applicants who have property in semi urban region. Around 76.82% applications are approved for applicants from semi urban region.

Applicants from rural region have less loan approval(61.45%). In all the region, male applicants get their loan status approved compared to females. Applicants having credit history have high chance of getting their loan status approved.

**Pre-Processing Pipeline:**

First, we need to fill these null values as it will effect model performance.

1. Gender is categorical type of data. Hence replacing null values with it's mode.

2. Married is categorical type of data. Hence replacing null values with it's mode.

3. Dependents is categorical type of data. Hence replacing null values with it's mode.

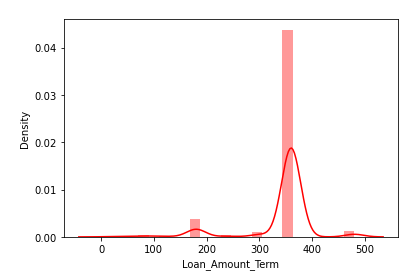
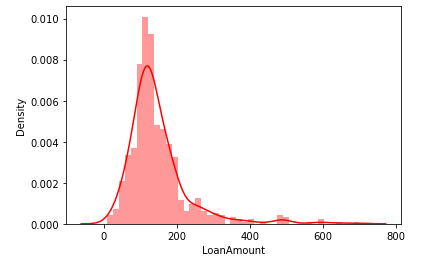
4. Self\_Employed is categorical type of data. Hence replacing null values with it's mode.

5. Loan\_Amount is a numerical type of data. Hence replacing null values with mean/median depending upon distribution of data.

6. Loan\_Amount\_Term is a numerical type of data. Hence replacing null values with mean/median depending upon distribution of data.

7. Credit\_History is categorical type of data. Hence replacing null values with it's mode.

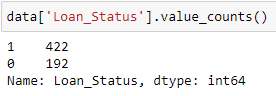
For imputing null values with mean/median it is necessary to check for distribution of data for attributes Loan\_Amount & Loan\_Amount\_Term.



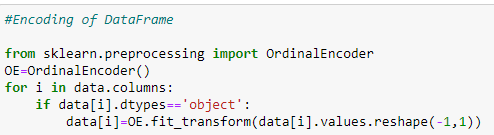
From distribution plot for Loan\_Amount & Loan\_Amount\_Term it can be seen that both the attributes are skewed. Hence replacing null values with median.

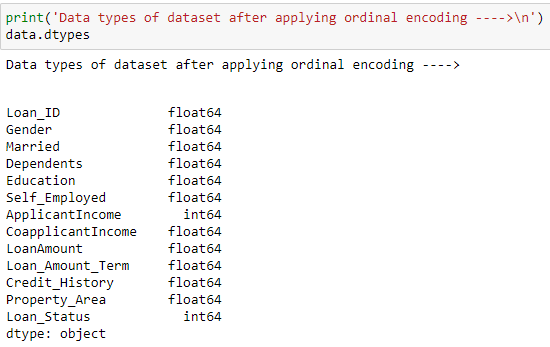
Secondly, Replacing Y and N of loan\_status to 1 and 0 to convert it to numeric value in order to find percentage.



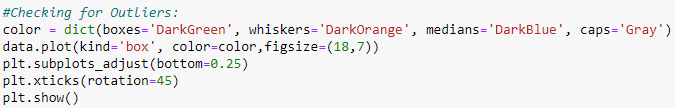


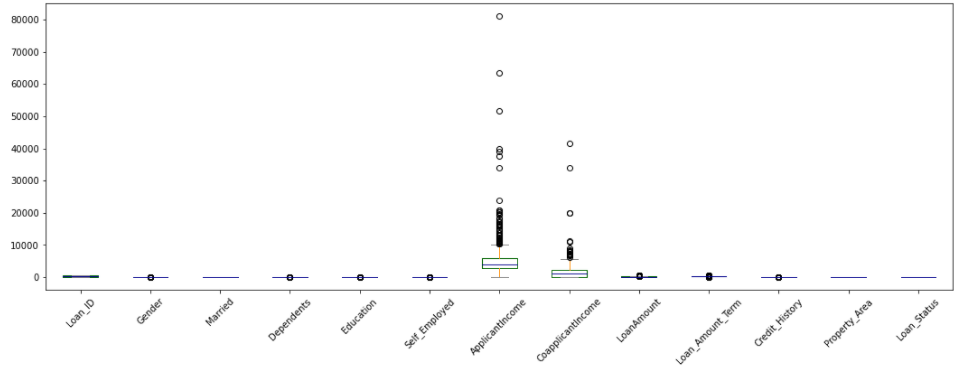
Thirdly, encoding categorical(object type data type) to numeric form before fitting and evaluating the model. Encoding is a required Pre-Processing step when working with categorical data for machine learning algorithms. Machine learning models require all input and output variables to be numeric. Using Ordinal encoding, each unique categorical value will be assigned a integer value. Every object type data is converted into float type value.





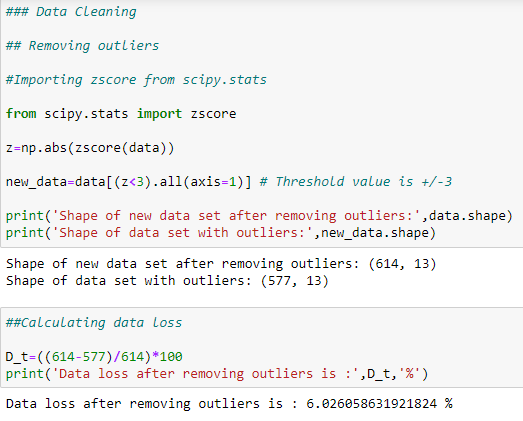
Fourthly, Checking for outliers and removing the same.



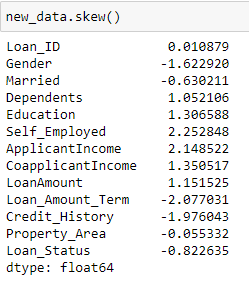


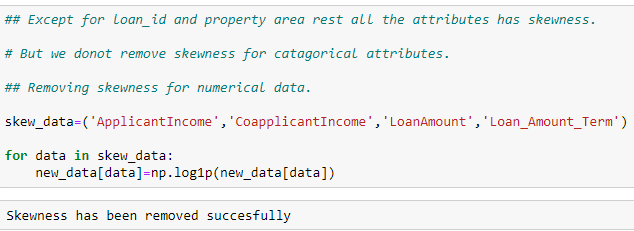
It is observed that Data-set has many outliers for attribute applicants income, co applicants income

Hence removing outliers using zscore method to improve performance of model.

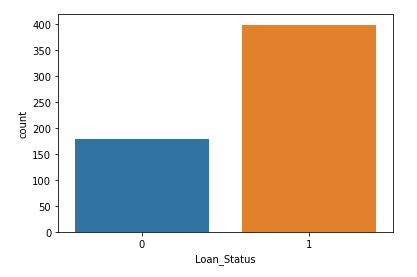


Fifth, Checking for skewness using skew() function and removing the same. As skewness can affect the performance of the model.

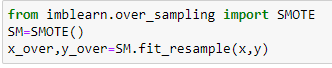




Sixth, on observing count plot for target variable, it was observed that data-set is imbalanced with more number of ‘1’s than ‘0’s as shown below.

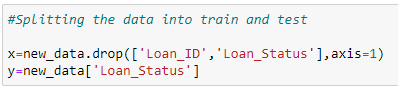


Imbalance of data affect performance of the model. Hence balancing the data-set using imblearn.SMOTE().

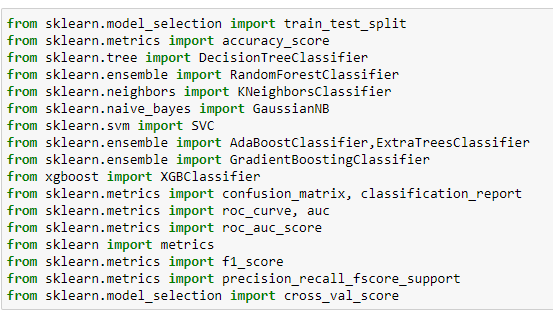


**Building Machine Learning Models:**

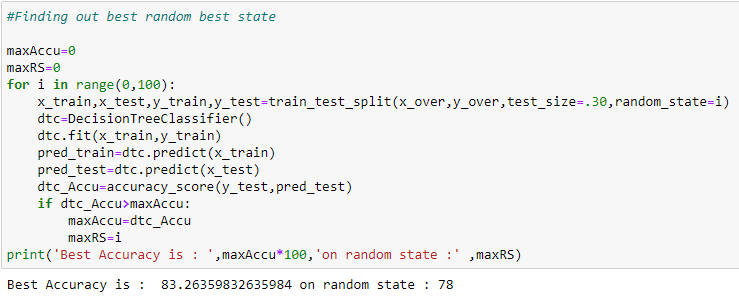
Splitting the data-set into training and testing set.



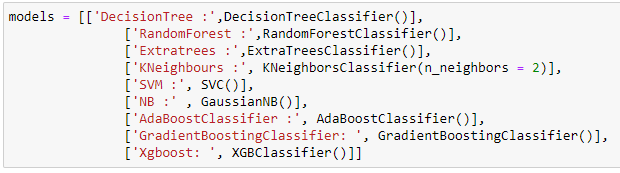
Importing libraries:



In order to find out best random state we apply for loop on Decision Tree Classifier Model by splitting 30% of data for test case. Firstly we fit the train data into the model and then predict for test data. On checking for best random state we get 78 as best random state and 83.26 as best accuracy.



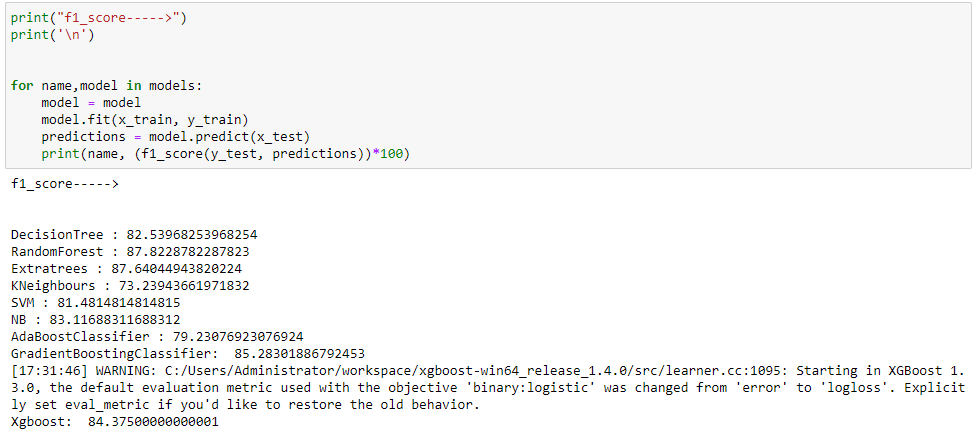
Defining machine learning models:



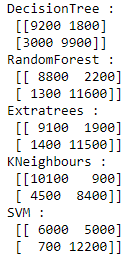
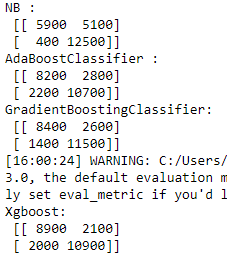
Checking for Accuracy score for each model:



Checking for f1 score for each model defined:



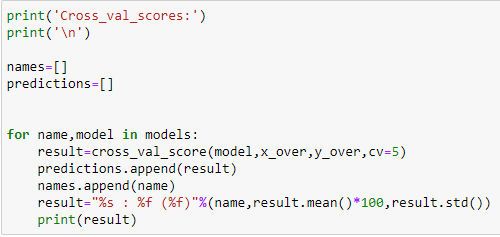
Checking for confusion matrix for each model defined:

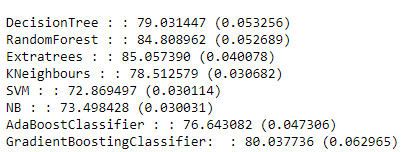
 

Among-st all the models, Random forest & Extra trees classifier has the highest accuracy and f1 score.

But first, let us check, how models performs, when we use cross validation. Cross validation on model eliminates problem of over fitting / under fitting of the data in data set.

Checking for cross validation score of each model:





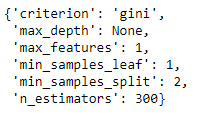


On observing accuracy score and cross\_val-score, Extra trees Classifier comes out to be best result with average accuracy of 85.05% with a standard deviation of 4%.

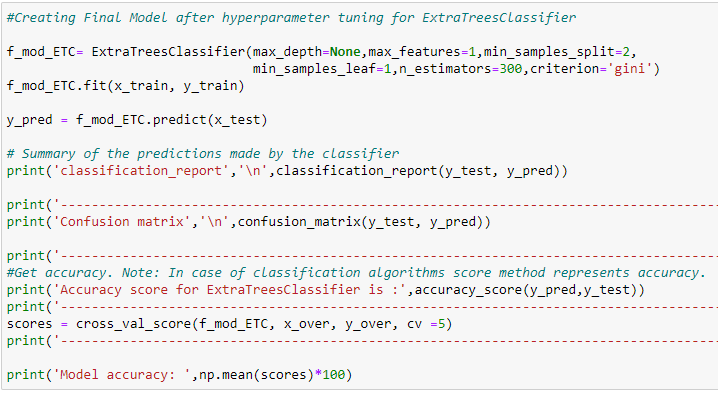
This means in our case that the accuracy of our model can differ + — 4%.

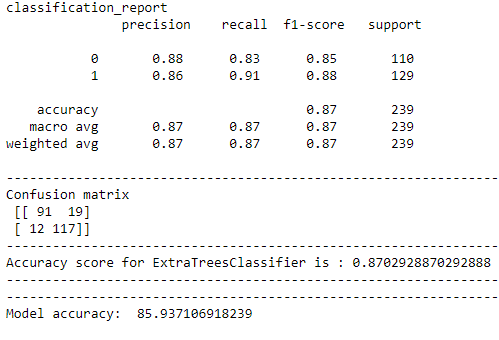
In order to get better accuracy and to improve performance of model hyper parameter tuning is done.

**Hyper parameters** are crucial as they control the overall behaviour of a machine learning model. The ultimate goal is to find an optimal combination of hyper-parameters that minimizes a predefined loss function to give better results. **Tuning** is the process of maximizing a model's performance without over-fitting or creating too high of a variance.



Testing new parameters:

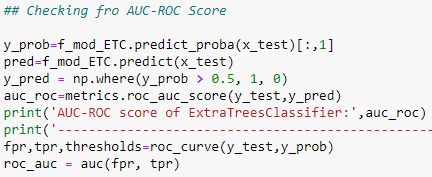


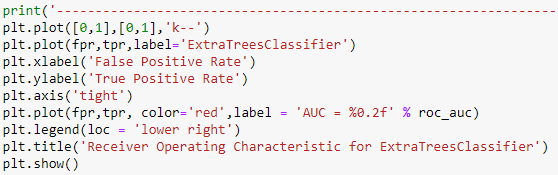


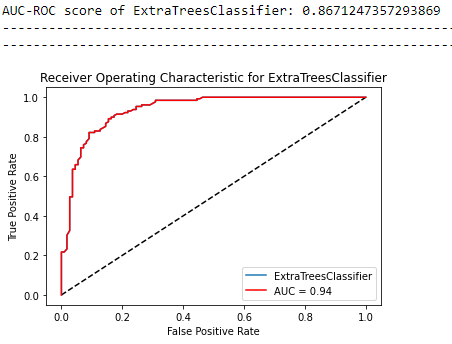
### **After hyper parameter tuning, model accuracy with high performance is 85.93%.**

### ROC AUC Curve:

Another way to evaluate and compare your binary classifier is provided by the ROC AUC Curve. This curve plots the true positive rate (also called recall) against the false positive rate (ratio of incorrectly classified negative instances), instead of plotting the precision versus the recall.







The black dotted line in the middle represents a purely extra tree classifier and therefore your classifier should be as far away from it as possible. Our Extra trees model seems to do a good job with a AUC score of 94%.In Machine Learning, performance measurement is an essential task. So when it comes to a classification problem, we can count on an AUC - ROC Curve. When we need to check or visualize the performance of the multi-class classification problem, we use the AUC (Area Under The Curve) ROC (Receiver Operating Characteristics) curve. It is one of the most important evaluation metrics for checking any classification model’s performance. It is also written as AUROC (Area Under the Receiver Operating Characteristics)

Of course we also have a trade-off here, because the classifier produces more false positives, the higher the true positive rate is.

**Concluding Remarks:**

We started with the data exploration where we checked on information about the data set, its data types, shape of data, checked about missing data and learned which features are important. During this process we used seaborn and matplotlib to do the visualizations. During the data pre processing part, we computed missing values with mean/median/mode of the data by checking distribution plot, converted features into numeric ones using encoding, grouped values into categories. Next we trained 9 different machine learning models which included Decision Tree Classifier, random Forest Classifier, Extra trees classifier, support vector, machine, k nearest neighbor, naive Bayes, Ada Boost classifier, Gradient Boost Classifier and XG Boost Classifier, checked for its accuracy score, f1 score and confusion matrix and picked one of them (Extra Trees ) and applied cross validation on it to fix problem of under fitting/ over-fitting. After that we tuned it’s performance through optimizing it’s hyper-parameter values.After hyper parameter tuning, model accuracy with high performance is 85.93%. Lastly, we checked for AUC-ROC score and plotted the graph for same. AUC score for Extra Trees classifier came out to be 94%.At the end we are ready with the model for deployment to the client.

Of course there is still room for improvement, like doing a more extensive feature engineering, by comparing and plotting the features against each other and identifying and removing the noisy features. Another thing that can improve the overall result would be a more extensive hyper-parameter tuning on several machine learning models.