Predicting Loan Application Status

The article is about predicting whether the applicant is eligible for the loan or not by using various machine learning algorithms

Problem Definition

Approving the loan is a very important process for banks. Banks should be careful in approving the loan or rejecting the loan. It is very difficult to predict which applicant will repay the loan in correct intervals. With the help of the given data set as a data scientist, we should predict which applicant is eligible to get the loan and which applicant is not.

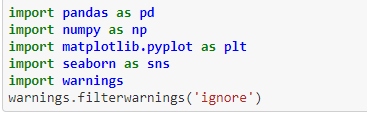


Data Description

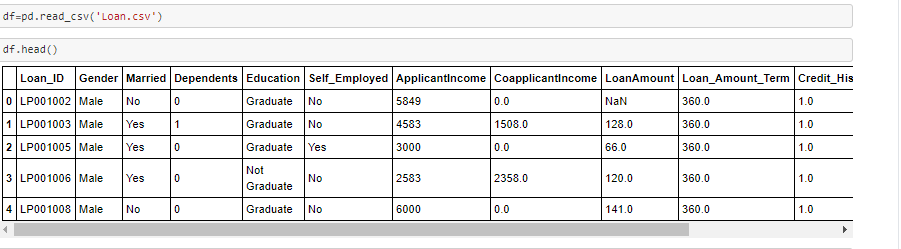
The data set contains details of the applicants who applied for the loan. The data set contains details like applicant gender, education, marital status, employment, the income of the applicant and income of the co-applicant, loan amount, credit history, property area, etc...

The coding used for this problem is Python

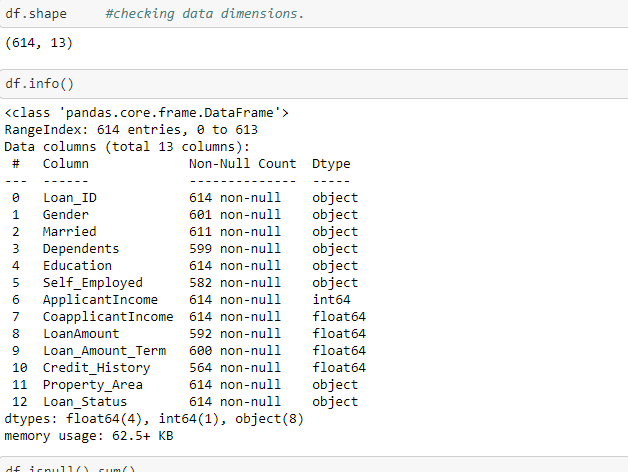
Import required libraries



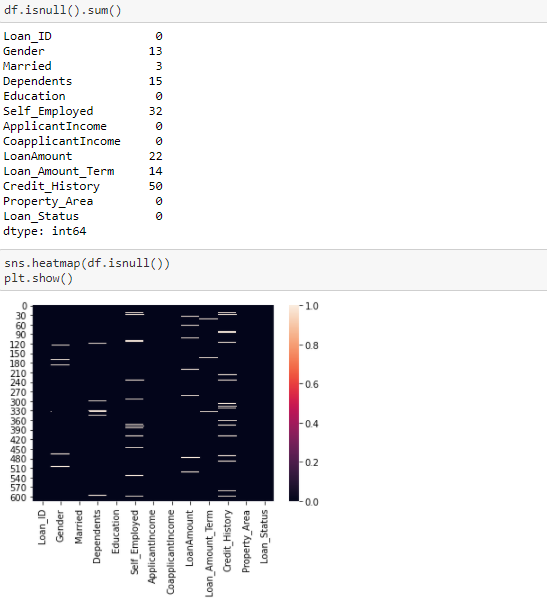
Loading the data



Checking data type and missing values

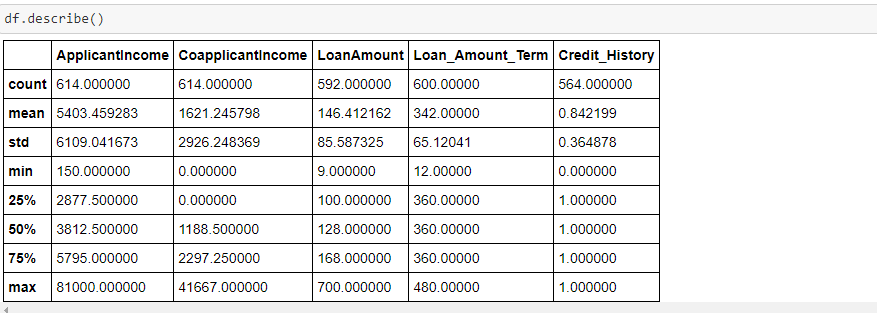


Missing Values



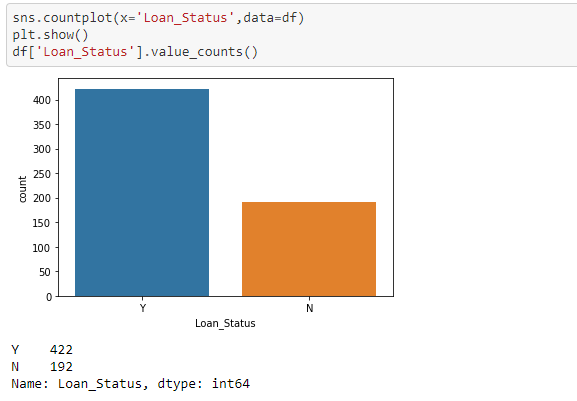
We can see that there are missing values (Nan) values in the given data set. We will treat the missing values in the future process.

Descriptive Statistics.



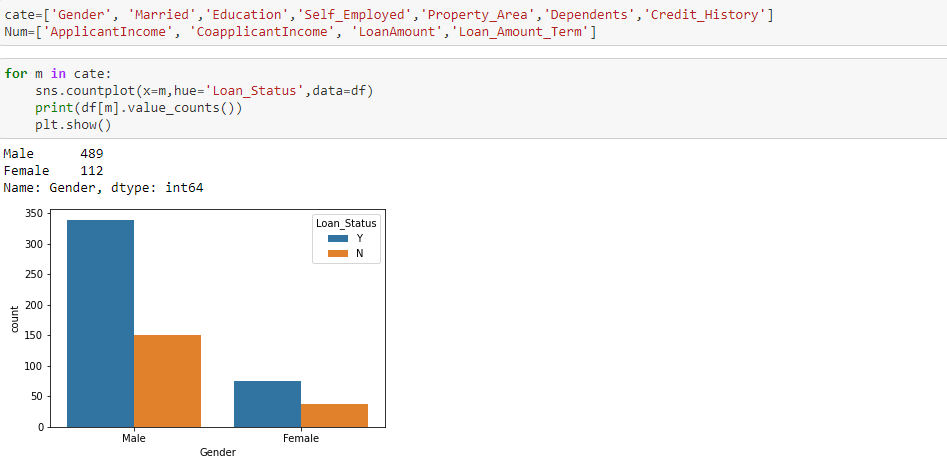
Though the column “Dependents” is in integer format it is not showing in the statistical summary. So let's check the values in this column. Also, the min and max values have a high difference. Also, the Standard deviation is high. So, there will be skewness in the data.

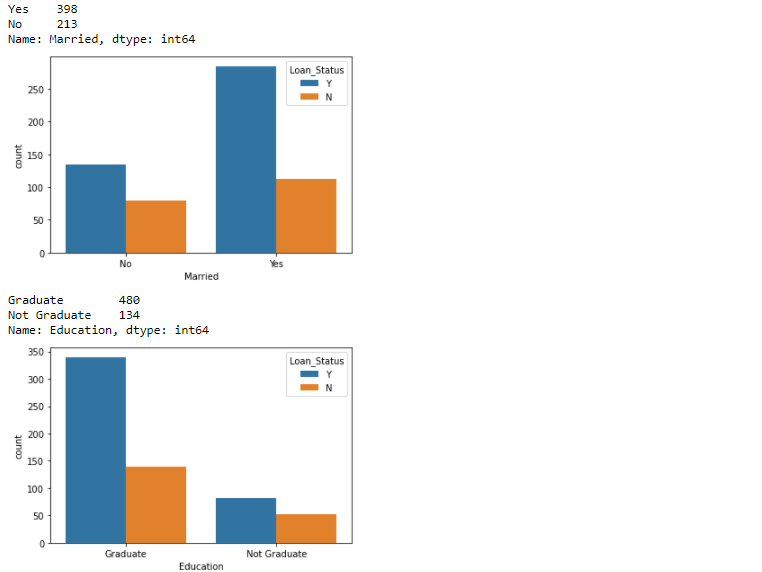
EDA

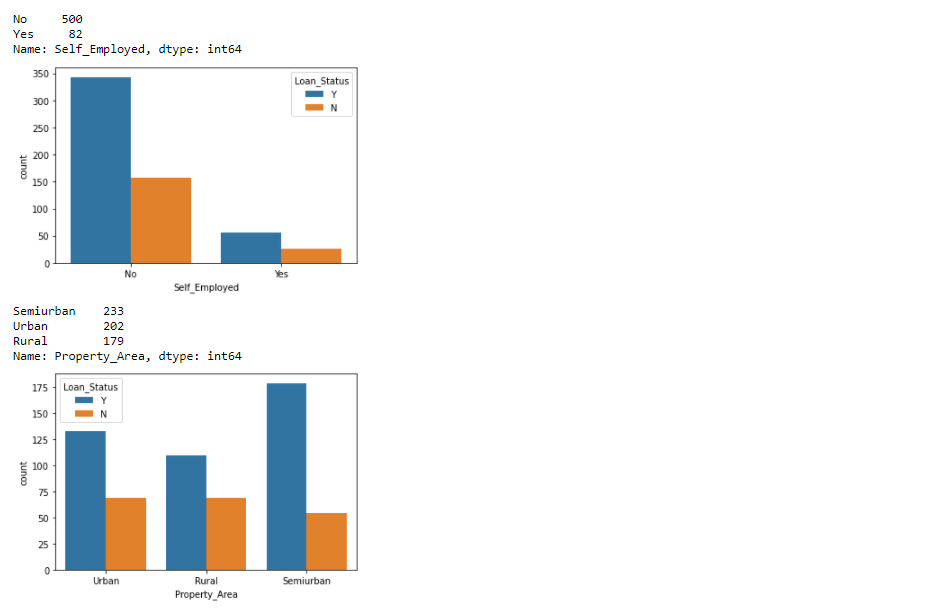


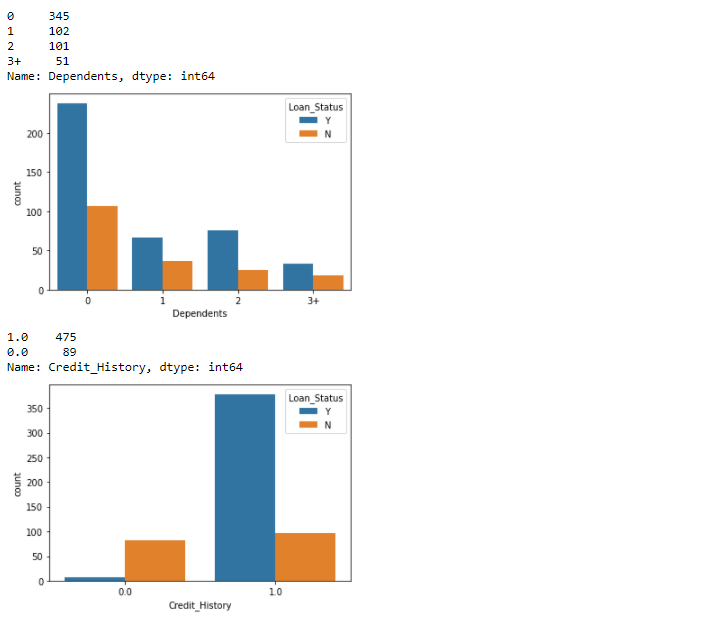
We can see that the data set contains 422 approved loan applicants' details and 192 loans rejected information.

Separating categorical data and Numerical data for analysis purposes. Here I took integer format columns Dependents and Credit History into the categorical side.





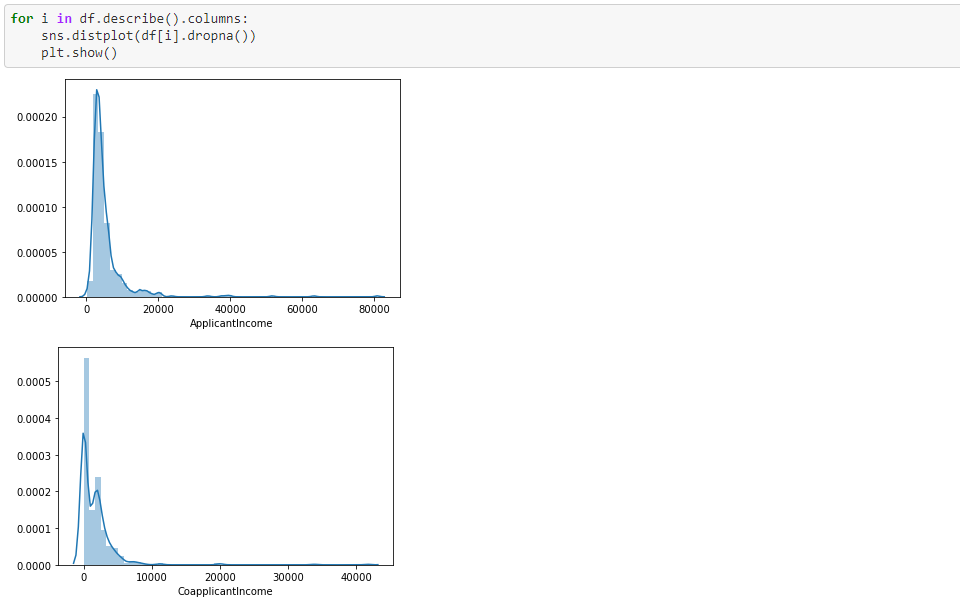


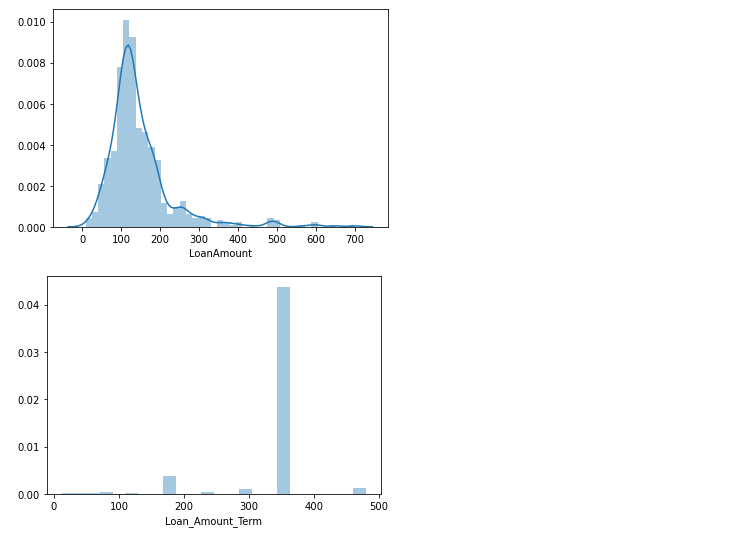


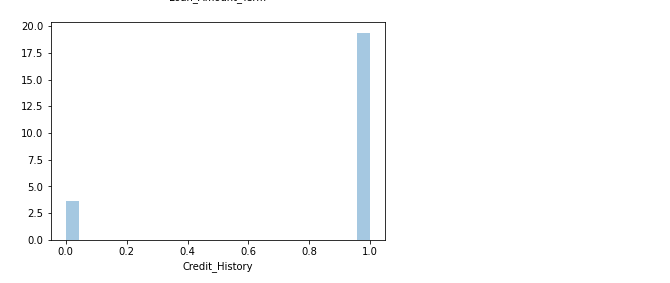
Observations from the categorical data:

* Gender - Male applicants approved more than Female applicants.
* Education – Higher loans approved for the graduates when compared to undergraduates.
* Self \_ Employeed – Most loans approved for the people who work for an organization with fixed salaries when compared to self employeed people.
* Property\_Area – Where the property rates high (urban,semi- urban) those property owners loans are approved when compared to the owners from rural areas.
* Dependents – Zero dependents applicants loan approved when compared to 1,2,3+ dependents. Also in this column due to the “3+” value, the data type is showing as an object instead of an integer. So we will change the “3+” as “3” in the future process to train the model
* Credit\_History – Credit History is the first and main thing, that should be checked by the bank to approve the loan. The people who have 0 credit history will not receive loans. So the applicants should maintain their credit score(1.0).

Numerical data:







Observations:

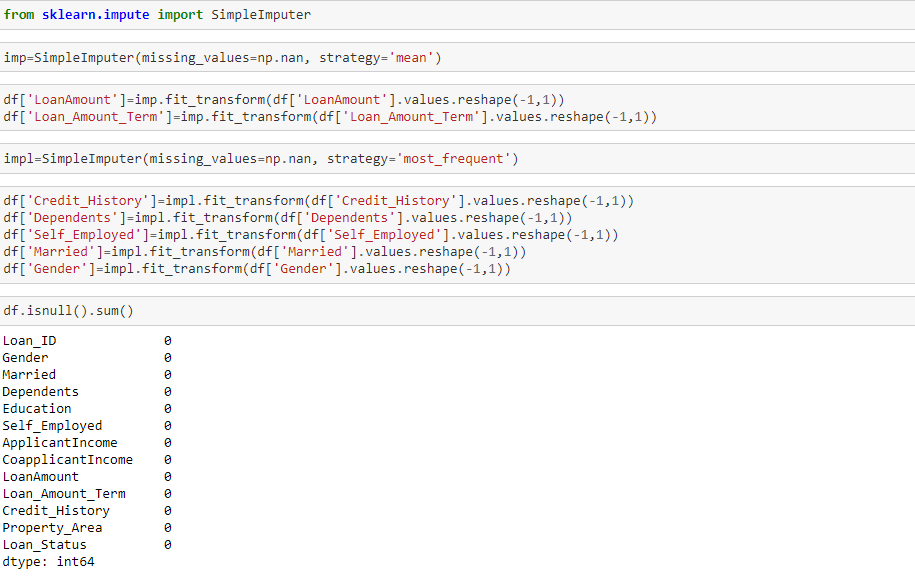
In the columns Applicant\_income, Coapplicant\_income, and Loan\_amount we can see that there is skewness in the data.

We can check the skewness with df.skew()

We also see that there are missing values in the given data set. First, we will treat the missing values, and later we will treat the skewness in the data.

Treating Missing Values:

Here I used scikit-learn ​simple imputer, to treat the missing values.



We replaced the null values with mean and most\_frequent values.

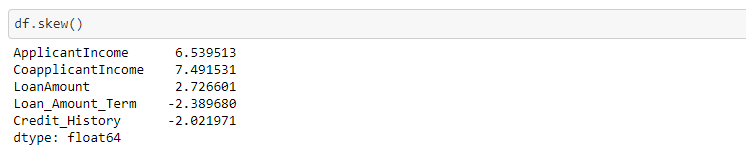
Treating Skewness:

What is Skewness and why it is bad?

skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. The skewness value can be positive, zero, negative, or undefined.

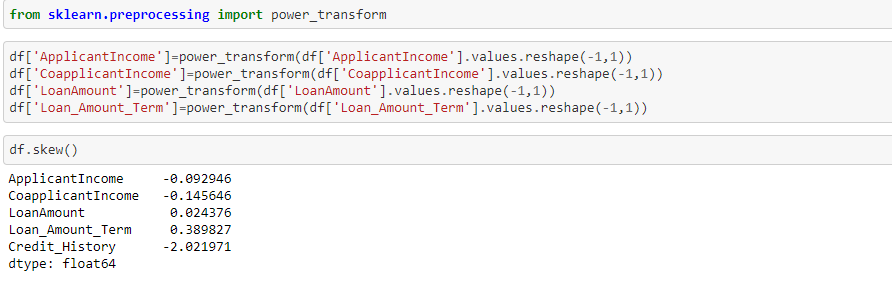
If there is too much skewness in the data, then many statistical models don't work but why. So in skewed data, the tail region may act as an outlier for the statistical model and we know that outliers adversely affect the model's performance. So, it is very important to remove the skewness before you transfer the data to train the model

We can check the skewsness with distplot or else with the code df.skew().



Take the threshold values as +/- 0.5 we can see that there is skewness in all the numerical data. However, the Credit\_History column has 2 values either 1 or 0. So, leaving the credit\_history we need to remove skewness from ApplicantIncome, Co-applicant income, LoanAmount, Loan\_Amount\_term.

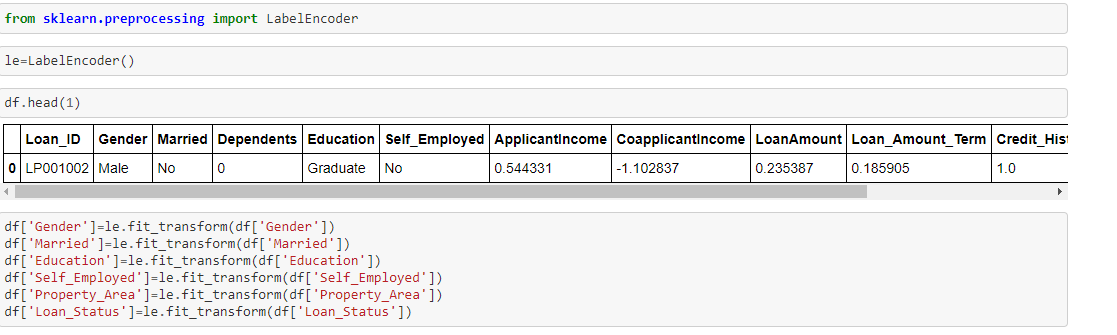
To remove the skewness we can use boxplot, log transformation, Squareroot transformation or power\_transformation(sklearn). In all these power\_trasnformation is the best which is adopted from scikit-learn. So I used the power\_transform method to remove the skewness.



We removed the skewness from the data, now the numerical variables are normally distributed.

**Prepare categorical variables for model using label encoder**

We need numerical data to train the model. So we will do an encoding process to change the categorical data into numeric data.

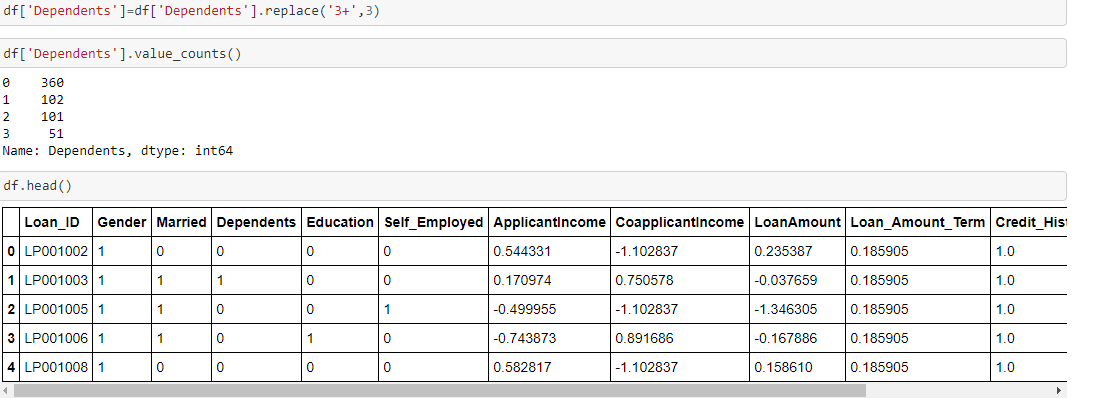


Feature Engineering:

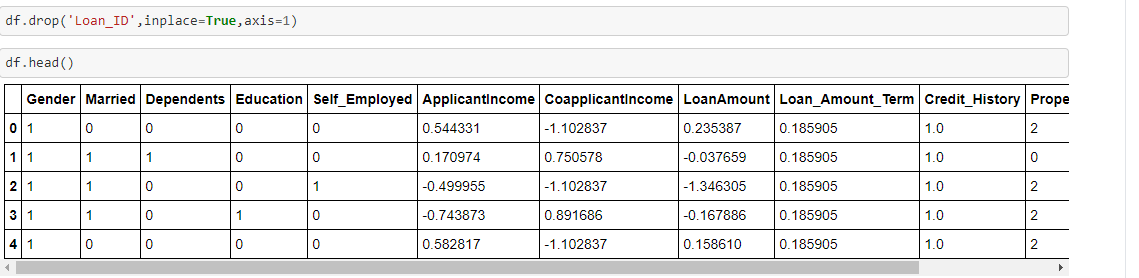
Feature engineering is a very important process to prepare the proper dataset to transfer the data to train the model for various machine learning algorithms. It will help to improve the model accuracy on unseen data.

Here in our datset:

We see that there is 3+ in the Dependent column that's the reason we are not seeing that column in the statistical summary. So we will replace this 3+ with 3 which will help us.



Loan\_ID is unique Id. So, no use of this column and we will remove it.



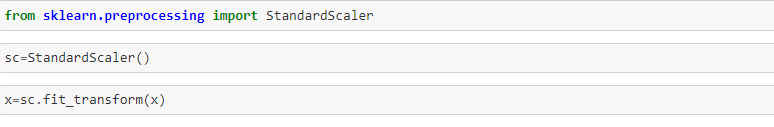
Splitting the dependent and Independent Variables:



Scaling:

In the descriptive statistics, we saw there is a huge difference in minimum and maximum values. It looks the data is not normalized. So we will do scaling. Here I used the Standard scaling method.

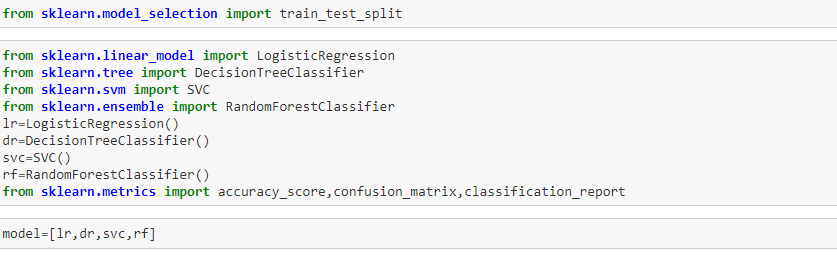
The idea behind standard scaling is that it will transform our given dataset such that its distribution will have a mean value 0 and standard deviation of 1.



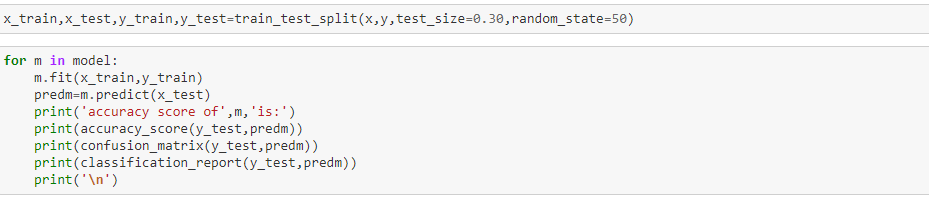
 Building Machine Learning Models.

The goal in this step is to train different machine learning algorithms and check which algorithm is performing good. Since the dependent variable is yes or no we will use classification models. The final goal is to get good accuracy and good f1 score.

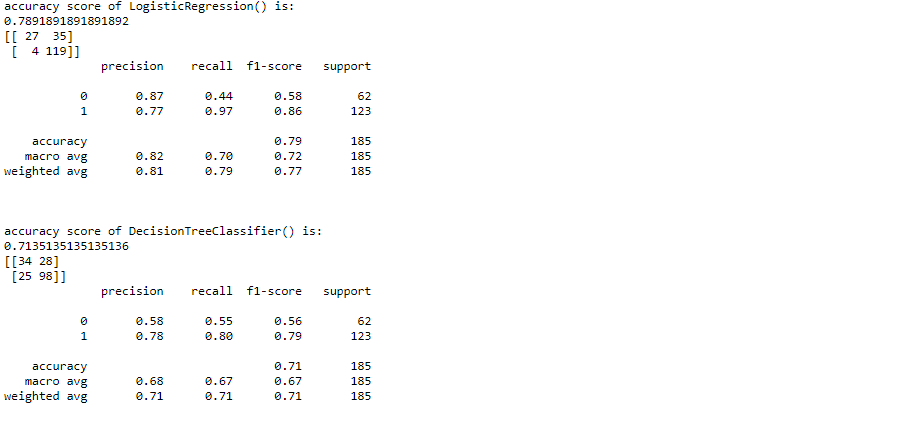
Required Machine Learning Libraries

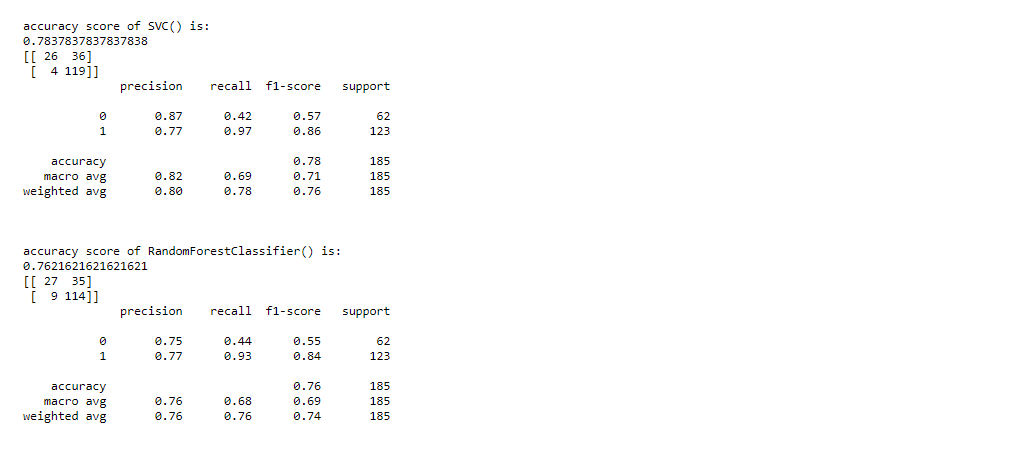


Now we will split the data for train and test and we will pass that data into the above algorithms at one go by using for loops and we will get required matrics scores.



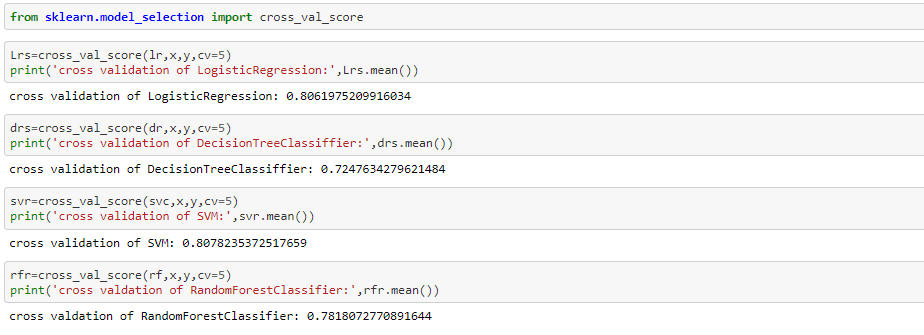
Now we will get accuracy score, confusion matrics and classification report of Logistic Regression, SVM, Decision Tree classifier and Randome forest classifier





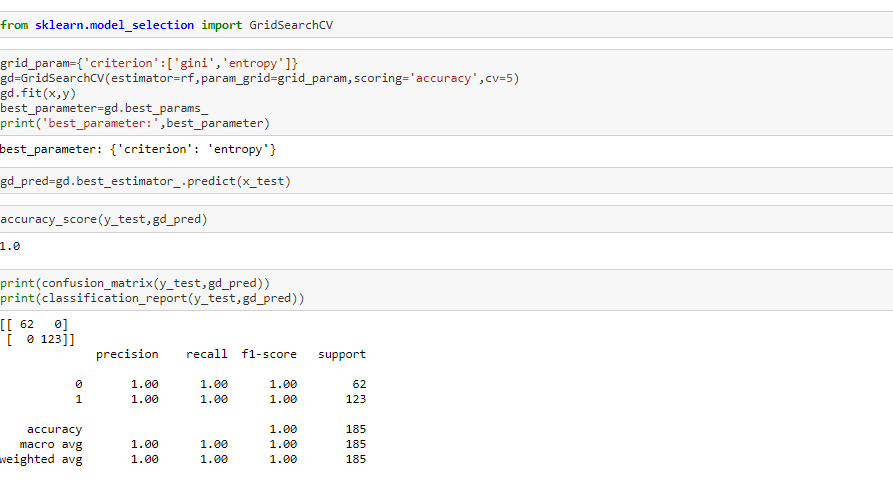
We are getting the accuracy with SVC (78%) and Random Forest classifier (76%), Logistic Regression is providing 78% and Decision tree classifier is providing 68%. But a f1 score is not good. Let's do cross-validation and Hyperparameter tunning to get better accuracy and fewer errors.

Cross Validation:



We can see that RandomForest Classifier is working best with 78% accuracy while doing cross-validation. Let’s do Hyperparameter tuning for Random forest classifier

Hyperparameter Tuning



With the best parameter “Entropy” after doing hyperparameter testing it shows 100% accuracy, zero errors in confusion matrics, and a 1.0 f1 score.

Conclusion:

In this type of problem Feature Engineering is the most crucial thing . You can see how we have handled the categorical data by doing encoding process and numerical data by removing skewness and scaling, also replacing missing values with simple imputer and also how we build different ML model on the same dataset, doing cross validation and hyperparameter testing.

Final Words:

Any reviews or feedback regarding article and way of problem solving is greatly appreciated, as a data science student it will help me to learn more. You can send your feedback at manjunathreddy.in@gmail.com.