**Loan Status Prediction**

The aim of this project is to build a machine learning model that can predict whether the loan of an applicant will be approved or not on the basis of the details provided in the dataset.

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

**Dataset Description**:

Independent Variables:

- Loan\_ID – id of Loan Application

- Gender – Gender of the applicant

- Married – whether the applicant is married or not

- Dependents – number of dependents of the applicant

- Education - indicates the applicant’s education status - Graduate or Not graduate

- Self\_Employed – whether the applicant is self employed or not

- ApplicantIncome – applicant’s income

- CoapplicantIncome – coapplicant’s income

- Loan\_Amount – loan amount applied by the applicant

- Loan\_Amount\_Term -- term of the loan in months

- Credit History – whether the loan applicant has any credit history in the Past

- Property\_Area - location of the property – urban, rural or semi-urban

Dependent Variable (Target Variable):

- Loan\_Status – loan Approval status. (approved =Y, not approved=N)

The dataset can be downloaded from

<https://github.com/dsrscientist/DSData/blob/master/loan_prediction.csv>

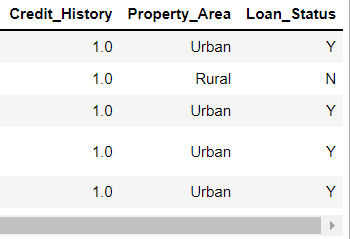
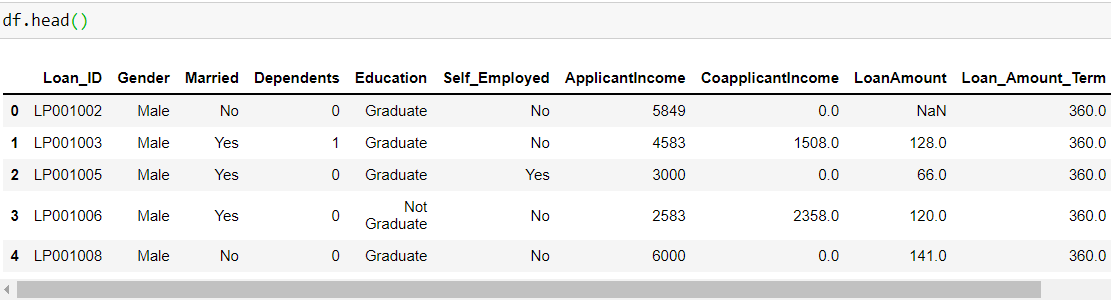
The link to the project can be found at https://github.com/sk1930/Projects/blob/main/Loan%20Application%20Status%20Prediction.ipynb

**Exploratory Data Analysis**

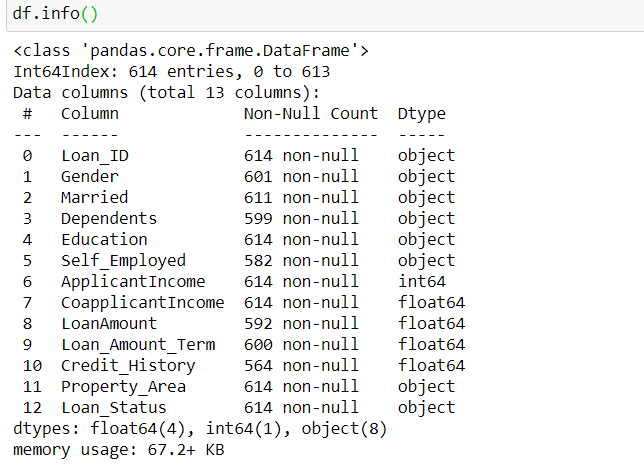
The size of the dataset is 614 rows with 13 columns.

The target feature “Loan Status” is categorical. Hence it is classification problem.

To get the overview of the data, looking at first 5 rows.



Checking the data types of each features using info

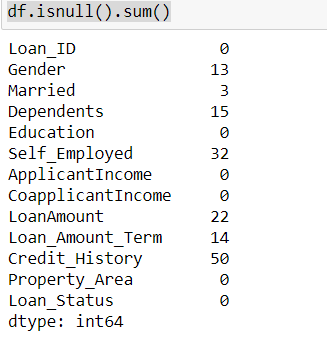


Loan\_ID, Gender, Married, Education, Self\_Employed, Property\_Area, Loan\_Status are categorical columns

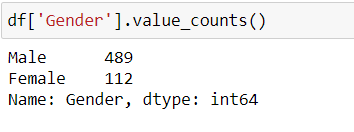
Dependents should be a numeric column, due to missing values it is being shown as object

ApplicantIncome, CoapplicantIncome, LoanAmount, Loan\_Amount\_Term, Credit\_History are numeric columns.

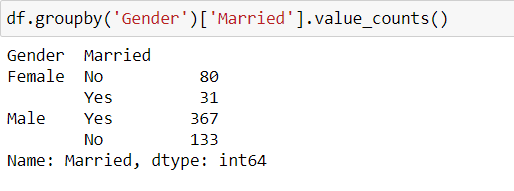
**Checking for missing values:**



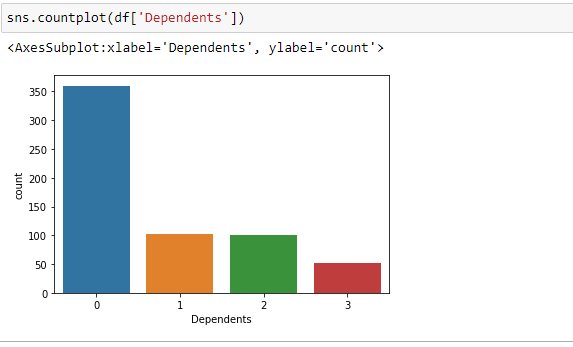
There are 13 missing values in Gender, 3 in Married , 15 for Dependents, 32 for self\_employed, 22 for LoanAmount, 14 for Loan\_Amount\_Term, and 50 missing values in Credit\_History.



As it can be seen that most of the values in Gender are Male, filling missing values in Gender feature with Male.

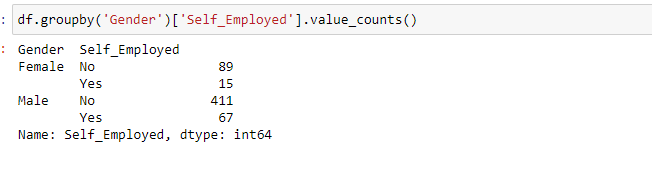


There are 3 missing values in Married feature. Most of the females are not married and most of the males are married so filling married feature with respect to Gender, ie. For Female filling missing married values with No and for Males filling missing married values with Yes.



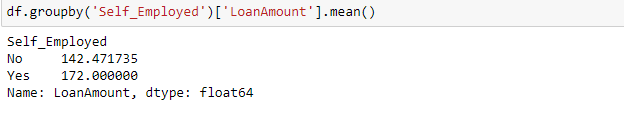
There are 15 Missing values in Dependents feature. 0 is occurring 350 times. So missing values in Dependents feature is filled with the mode (mostly repeated value). And also the value 3+ is replaced with 3 for number of dependents.

Loan applicants with 0 number of dependents is around 350, and for 1,2 number of dependents there are around 100 number of applicants each, and loan applicants with 3 number of dependents are around 50.



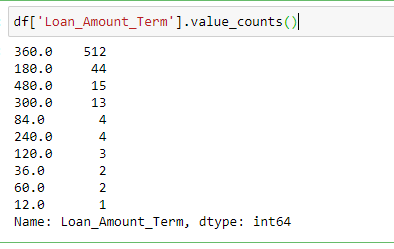
There are 32 missing values for self employed.

Filling missing values in Self\_ Employed with respect to Gender. With respect to Gender, in females and males both majority are not Self Employed. So filling missing Self\_ Employed with No.

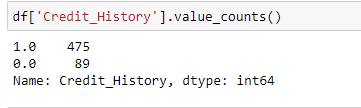


There are 22 missing values in Loan Amount and filling missing values based on Self employed. Grouping the data based on Self\_ employed feature and taking mean of loanAmount and filling the missing values with that mean values.

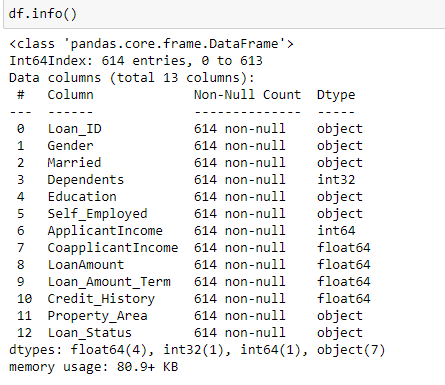
df ['LoanAmount'] = df.groupby ('Self\_Employed')['LoanAmount'].transform(lambda val: val.fillna(val.mean()))



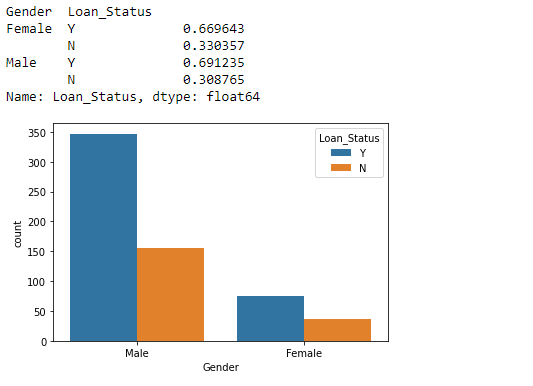
There are 14 missing values in Loan Amount Term. Mostly repeated value us 360 and it is repeating 512 times. So filling missing values with the mode.



In Credit History there are 50 missing values. And credit history 1 is repeating 475 times. So filling missing values with the mode.

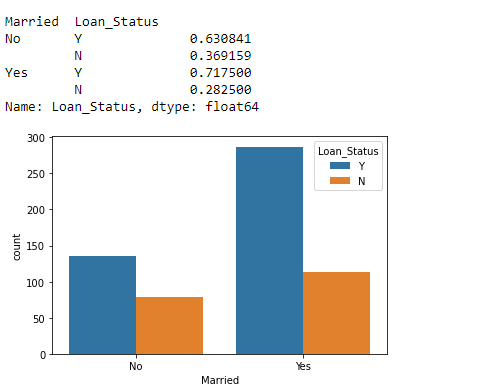


After treating missing values in each feature, there are no more missing values in any feature now.



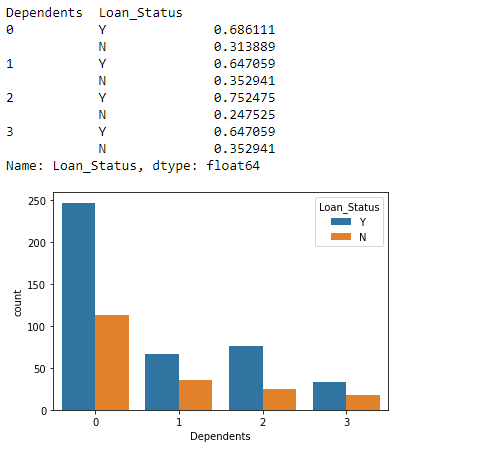
Gender Vs Loan Status

Gender captures two unique values Male and Female. Large number of loan applications are applied by Male compared to Females. There is high rate of loan approval in males with 69% and in females there is a bit less approval rate with 66 percent.



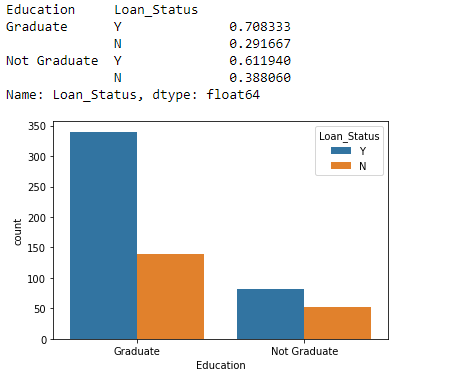
Married vs Loan Status

Married feature has two unique values – Yes and No. More Married people apply for loans than unmarried people. There is a high approval rate for Married people with 71% and in case of unmarried people there is somewhat less approval rate with 63%.



No. Of Dependents vs Loan Status

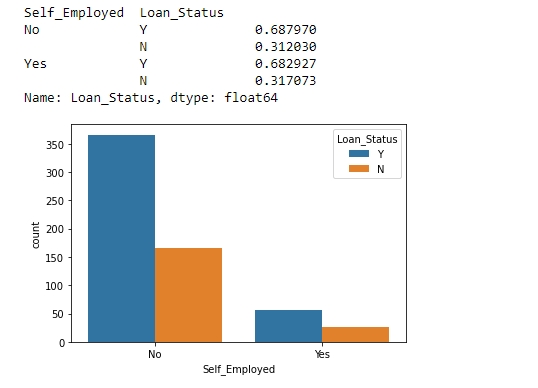
The dependents feature indicates the number of features of the loan applicant. It has 4 unique values- 0, 1, 2 and 3. Majority of the population have zero dependents. The highest approval rate is for people having 2 dependents with 75%. The least approval rate is for people having 1 and more than or equal to 2 dependents with 64%. In people with only 0 dependents approval rate is 68 percent.



Education vs Loan Status

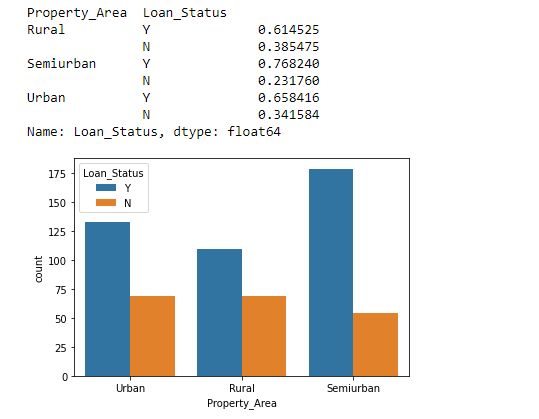
Education feature captures two values. One is Graduate, other is Not Graduate. More Graduates apply for loans.

As expected there is high approval rate for applicats who are graduates. The approval rate for graduates is 70% while for applicants who are not graduates the approval rate is 61 percent.



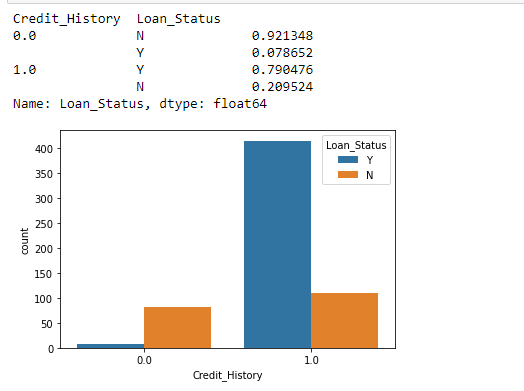
Self Employed Vs Loan Status

Self employed feature captures whether the Loan Applicant is Self Employed or not. There are less self employed loan applicants. There is almost same approval rate whether the loan applicant is self employed or not.



Property area vs Loan status

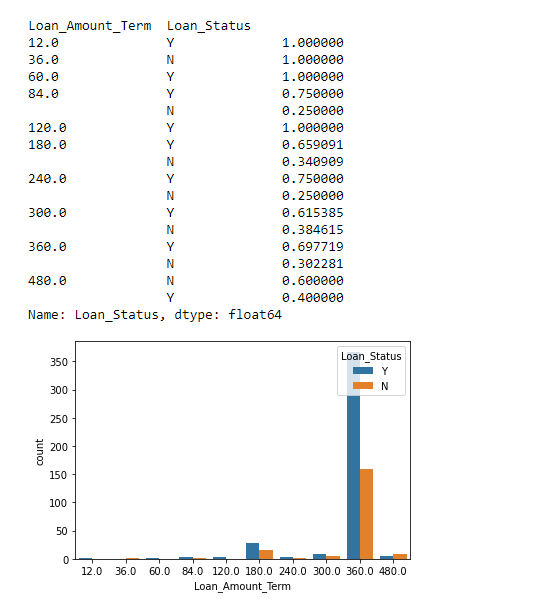
Property area captures the location of the property. It has 3 unique values – Urban, Rural, Semi urban. Most of the property areas are in Semi Urban area. Highest approval rate is for loan applicants with property area is Semi uban location with 76 percent approval rate, while the least approval rate is for loan applicants with property area in Rural location with 61 percent loan approval rate. And Loan applicants with property area in Urban location have approval rate of 65 percent.



Credit History Vs Loan Status

Credit History feature indicates whether the person applying for a loan has any credit history in the past. It has 2 unique values 1 indicating that the person has a credit history and 0 indicating that the person does not have any credit history in the past.

There are large number of loan applications having credit history in the past. The highest approval rate is in applicants not having a credit history in the past with 92 percent, while in case of applicants with credit history in the past the approval rate is some what less with 79 percent.

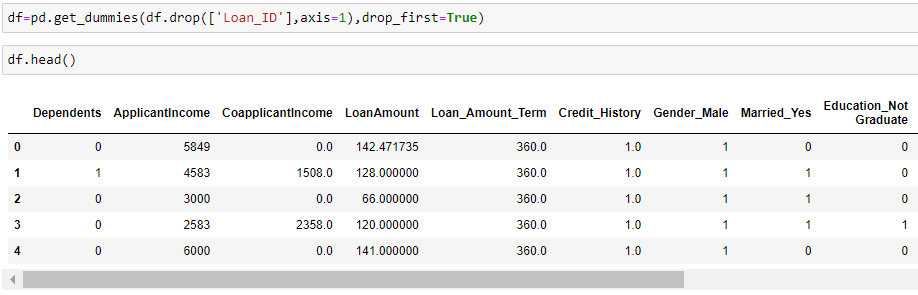


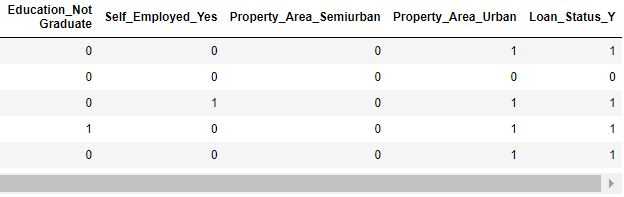
Loan Amount Term vs Loan status.

Loan Amount term indicates the tenure during which the loan is repaid in months. There are some constant values for loan amount term like 12, 36, 60, 84, 120, 180, 240, 300, 360, 480.

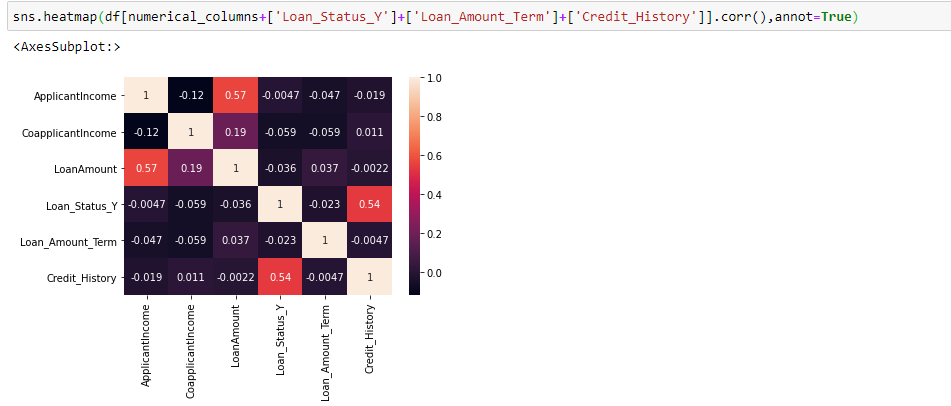
Most of the loan applications are with 360 months that is 6 years of repayment term.

Highest approval rate is 100 percent for 12, 60, 120 months and least approval rate is for 36 months with 100 percent again.





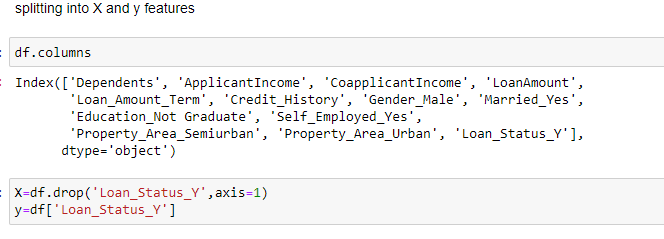
Calling df = pd . get\_dummies ( df.drop ( [ ‘Loan\_ID'] , axis = 1 ) , drop\_first = True ) to transform all categorical features in multiple features with Yes = 1 or No = 0 values. And also passing drop\_first as True so that it avoids multicollinearity problem.



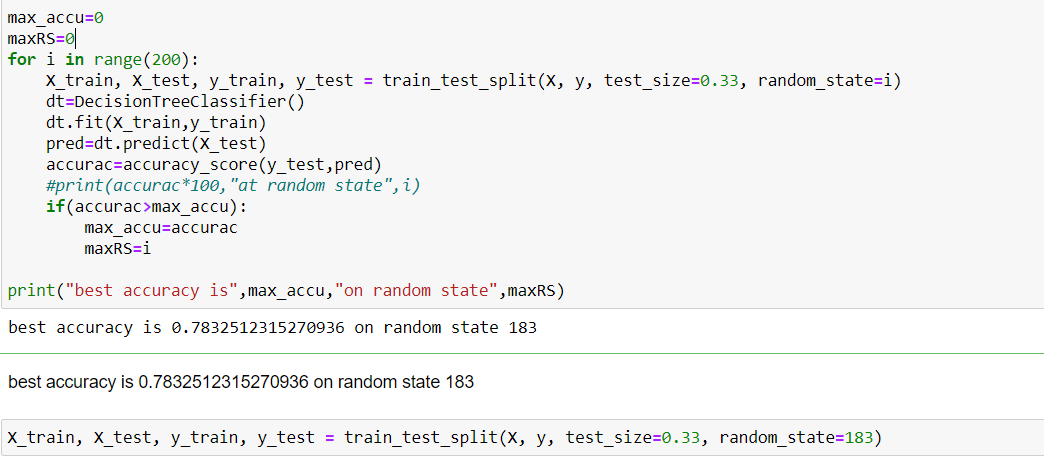
Checking the corelation of the dataset.

Credit History is positively corelated with the Loan Status, while others are negatively corelated.

**Splitting the data in to X and Y features:**

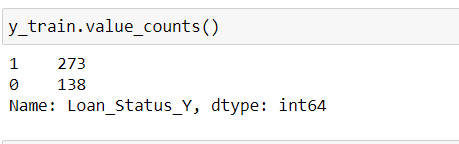
****

Splitting the data in to X (independent features) and y (dependent feature).

****

Identifying the best random state by trying 200 values ranging from 0 to 199 for random state.

The best random state is 183 with an accuracy of 0.783



Checking the value counts of target variable, to identify any class imbalance issue.

Here, the count of 0 – loan not granted is 138 and count of 1 – loan approved is 273. So clearly there is class imbalance issue.

To overcome the class imbalance issue there are two approaches – Upsampling and Downsampling.

**Upsampling**:

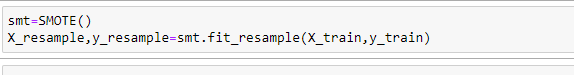
It is a procedure where synthetically generated data points (corresponding to minority class) are injected into the dataset. After this process, the counts of both labels are almost the same. This equalization procedure prevents the model from inclining towards the majority class. Furthermore, the interaction (boundary line) between the target classes remains unaltered. And also, the Upsampling mechanism introduces bias into the system because of the additional information.

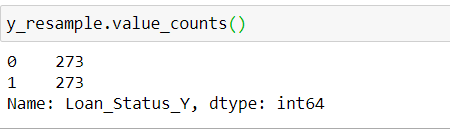
Downsampling is a mechanism that reduces the count of training samples falling under the majority class. As it helps to even up the counts of target categories. By removing the collected data, we tend to lose so much valuable information.

So it is better to do Upsampling and here we are applying SMOTE an implementation for Upsampling.

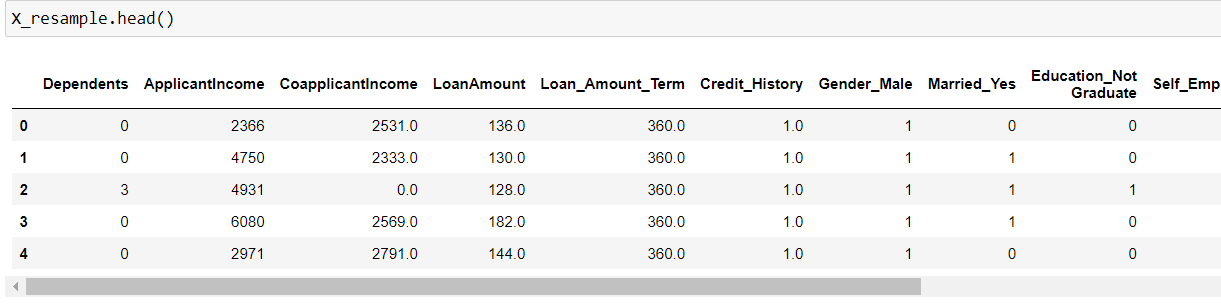
**SMOTE (Synthetic Minority Oversampling Technique) — Upsampling :-**

It works based on the K Nearest Neighbours algorithm, synthetically generating data points that fall in the proximity of the already existing outnumbered group. The input records should not contain any null values when applying this approach.

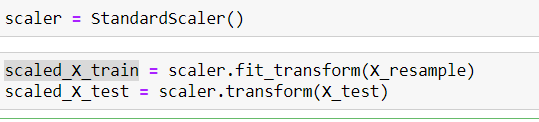




After applying SMOTE the training data is balanced with equal number of classes with each having 289 samples.

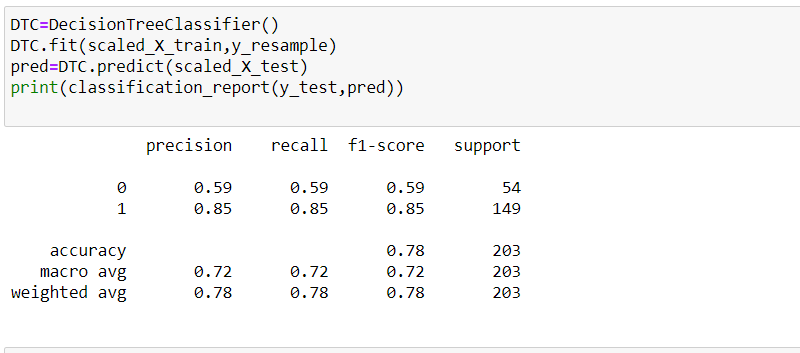


All the features’ values in the training data are not in the same range. So fitting the



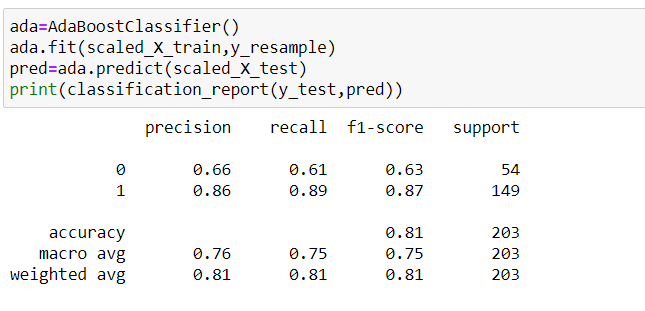
Standard Scaler on the training data and transforming the training and testing data. Once the standard scaler is applied on the training and testing set all the values will be in the same range.

**Model Training**:

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

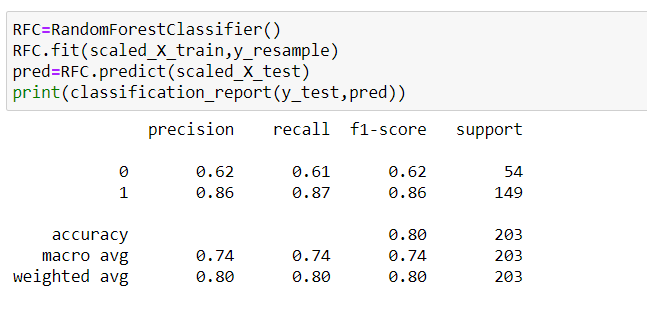
The accuracy on test set with Decision Tree classifier is 78% and F1 scores are 59 and 85 for label 0 and label 1 respectively.

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

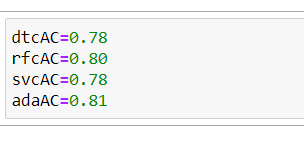


The accuracy achieved on test set with AdaBoost classifier is 81 percent and f1 scores are 63 and 87 on label 0 and 1 respectively.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max\_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

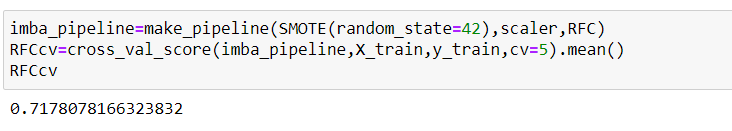
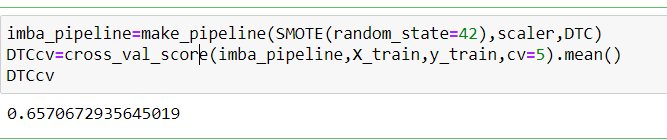


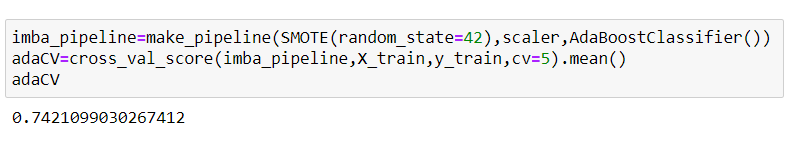
The accuracy achieved on test set with Random Forest classifier is 80 percent and f1 scores are 62 and 86 on label 0 and 1 respectively.

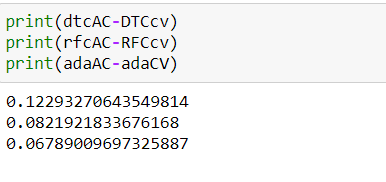


Checking the cross validation scores:

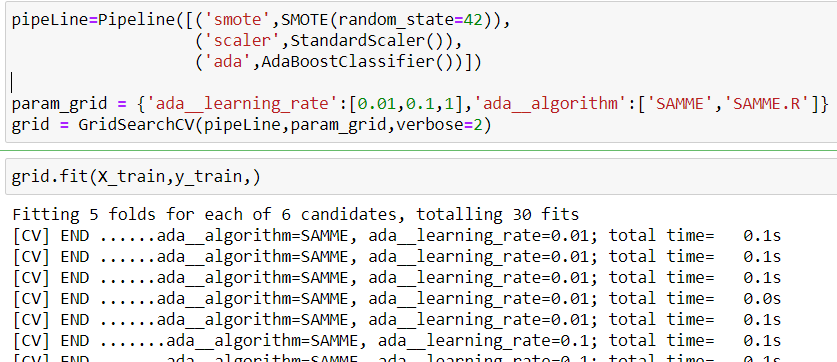
Sequentially apply a list of transforms and a final estimator. Intermediate steps of the pipeline must be ‘transforms’, that is, they must implement fit and transform methods. The final estimator only needs to implement fit. The transformers in the pipeline can be cached using memory argument.







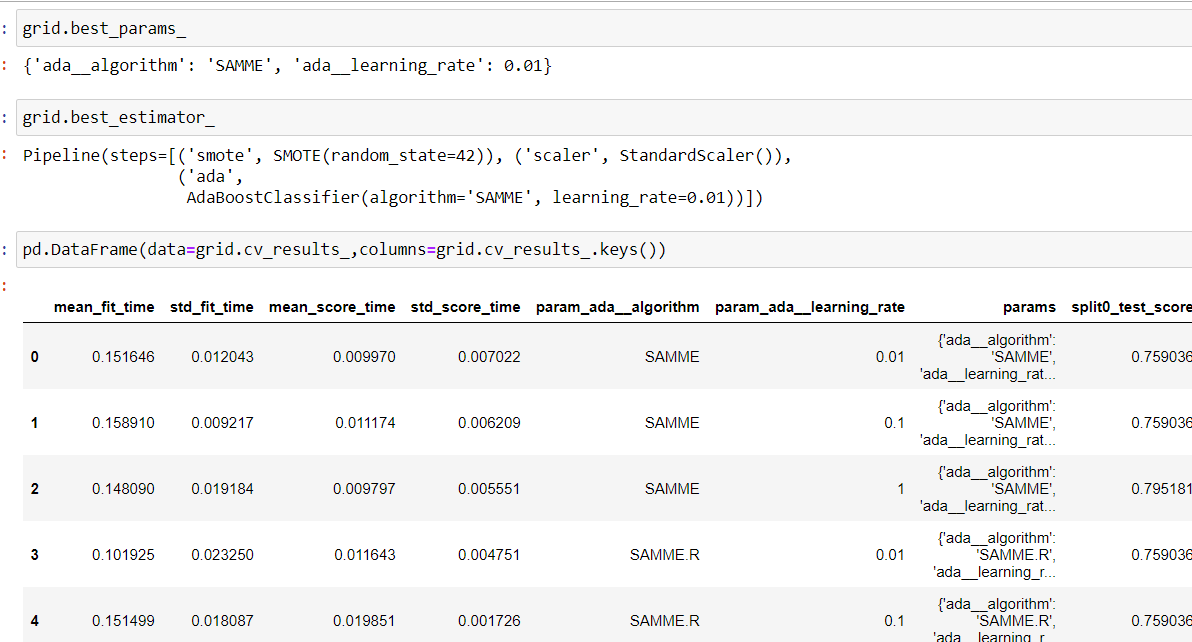
AdaBoost Classifier is the best model as it has the highest precision and recall and accuracy as well. Ada Boost classifier has the less difference between the accuracy scores( on test set) and Cross Val score(on validation), so it is the best model.

**Hyper Parameter Tuning - AdaBoost Classifier:**

Hyperparameters are the variables that the user specify usually while building the Machine Learning model. Thus, hyperparameters are specified before specifying the parameters or we can say that hyperparameters are used to evaluate optimal parameters of the model. the best part about hyperparameters is that their values are decided by the user who is building the model. For example, max\_depth in Random Forest Algorithms, k in KNN Classifier.

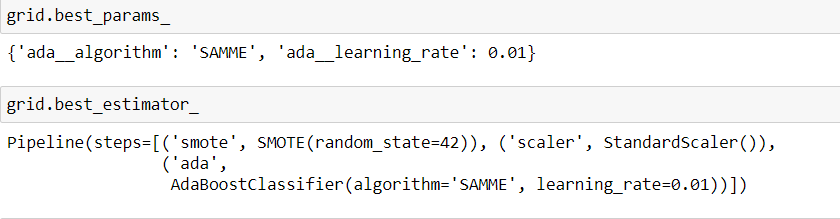
Grid Search uses a different combination of all the specified hyperparameters and their values and calculates the performance for each combination and selects the best value for the hyperparameters. This makes the processing time-consuming and expensive based on the number of hyperparameters involved.

In GridSearchCV, along with Grid Search, cross-validation is also performed. Cross-Validation is used while training the model. As we know that before training the model with data, we divide the data into two parts – train data and test data. In cross-validation, the process divides the train data further into two parts – the train data and the validation data.

****

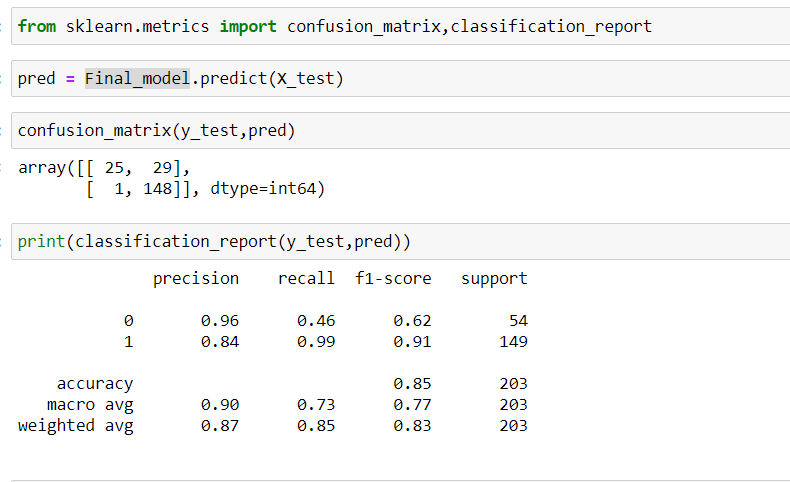
**Training the final model with the best parameters based on GridSearchCV:**

Creating a pipeline with Smote for Upsampling the training data, scaler – fitting the scaler on training data, transforming the training and testing data, AdaBoostClassifier with learning rate of 0.01 and algorithm as SAMME.

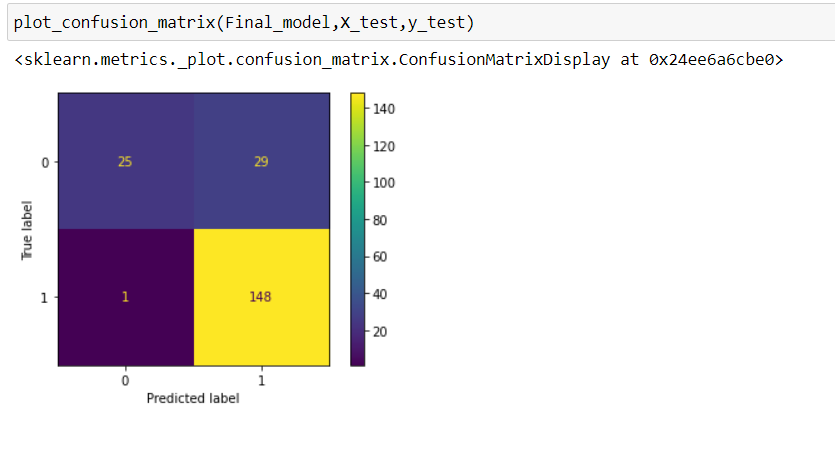




After training the model on the training data with the best parameters , accuracy of 85% is achieved on the testing data.

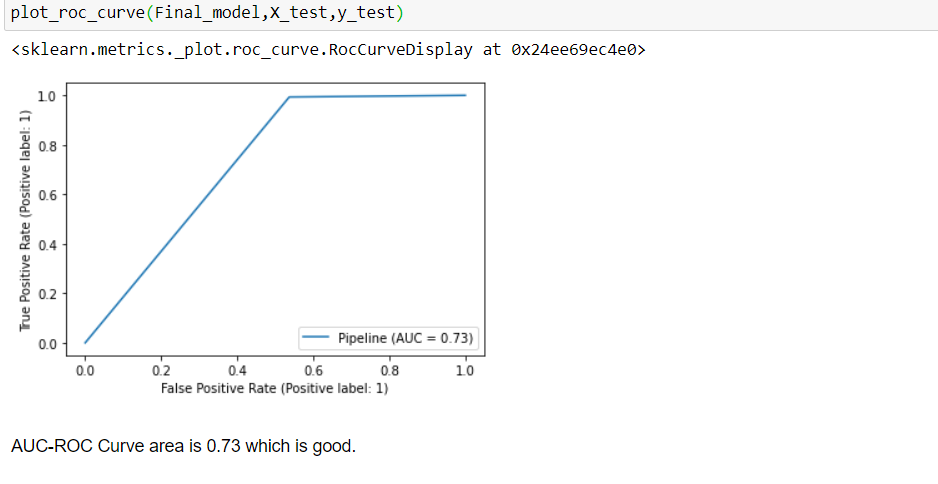


Classification report shows the f1 scores 0.62 and 0.91 on label 0 and label 1 respectively.



Confusion matrix reveals that out of 149 records from class1, 148 are correctly predicted as class 1 and on record is falsely predicted as class 0. Out of 54 records in class 0, 25 are correctly predicted as class 0 and 29 records are incorrectly classified as class 1.

A receiver operating characteristic (ROC), or simply ROC curve, is a graphical plot which illustrates the performance of a binary classifier system as its discrimination threshold is varied. It is created by plotting the fraction of true positives out of the positives (TPR = true positive rate) vs. the fraction of false positives out of the negatives (FPR = false positive rate), at various threshold settings. TPR is also known as sensitivity, and FPR is one minus the specificity or true negative rate.

The roc\_auc\_score function computes the area under the receiver operating characteristic (ROC) curve, which is also denoted by AUC or AUROC. By computing the area under the roc curve, the curve information is summarized in one number.

In Conclusion, the AdaBoost classifier is chosen as the best model and the accuracy is 85%.

The project can be found at https://github.com/sk1930/Projects/blob/main/Loan%20Application%20Status%20Prediction.ipynb