**Predicting Loan Application Status Using Machine Learning Model**

****

**Introduction:**

**Loan!!!** A common reason that people take out a loan is to consolidate their debt. Most people do not ask for money help from family and friends. Instead, they apply for loan. After loan application the bank checks various factors to decide whether to approve or reject the application. Let’s build the machine-learning models to classify the loan application status as yes or no!

**Problem Statement:**

In this project we will examine the data and build a machine learning model that can predict whether the loan of the applicant will be approved or not on the basis of the details provided in the dataset.

**Dataset:**

The dataset used in this project is **loan.csv**. This dataset includes details of applicants who have applied for loan. The dataset includes details like credit history, loan amount, their income, etc. The dependent or target variable is “Loan\_Status”. The description of each feature is as follows:

1. Loan\_ID => Unique Loan ID for the Applicant applying for Loan.
2. Gender => Gender of the Applicant
3. Married => Marital status of the Applicant
4. Dependents => Number of dependents of the Applicants
5. Education => Applicant’s education status(Graduate/Not Graduate)
6. Self\_Employed => Applicant is self-employed or not
7. ApplicantIncome => Applicant’s income
8. CoapplicantIncome => Co-applicant’s income
9. LoanAmount => Loan amount in thousands

10. Loan\_Amount\_Term => Term of the loan in months

11. Credit\_History => Applicant’s previous credit history meeting guidelines

12. Property\_Area => Urban, Semi-Urban, or Rural Areas

13. Loan\_Status => Loan Approval status (Target Variable)

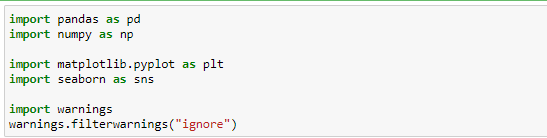
The python libraries and packages we’ll use in this project are namely:

* Numpy
* Pandas
* Seaborn
* Matplotlib
* Scikit-learn

**Let’s get started!**

**Importing the python libraries and packages:**

The first and foremost step involves importing necessary libraries and packages and loading the dataset as a pandas dataframe.

****

**Reading the CSV file of the dataset:**

Pandas read\_csv() function imports a CSV file (in our case, loan.csv) to DataFrame format.

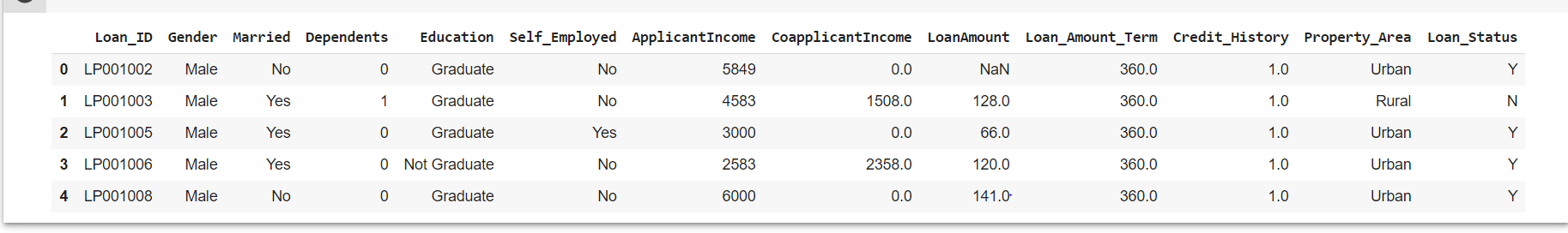


**Examining the Data:**

After importing the data, to learn more about the dataset, we’ll use **.head() .info()** and **.describe()** methods.



The **.head()** method will give you the **first 5 rows** of the dataset. Here is the output:

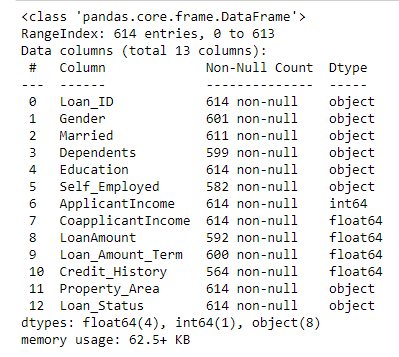


Result of df.head()

Loan\_Status which is a binary variable that takes the values Yes(Y)/No(No) serves as the Target Variable which needs to be predicted.



The **.info()** method will you a **concise summary** of the DataFrame. This method will print the information about the DataFrame including the index dtype and column dtypes, non-null, and memory usage. Here is the output:

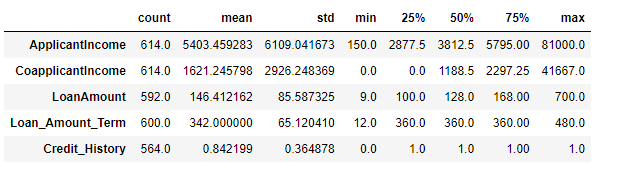


Result of df.info()

## 

The **.describe()** is used to view some basic **statistical details** like percentile, mean, std, etc. of a DataFrame or a Series of numeric values.

Here is the output:



Result of df.describe()

## The shape of the dataset:

## 

Here is the output:



Result of df.shape

This shows that our dataset contains 614 rows i.e. instances of loan applications and 13 columns i.e. the details of applicants like gender, married, dependents, education, income, loan amount, credit history, etc.

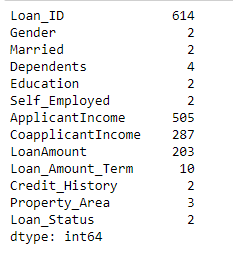
**Unique value counts for each column:**

****

Result of df.nunique()

The **.nunique()** return columns with number of distinct elements. NaN values are not counted.

Here is the output:



Result of df.nunique()

This shows that Loan\_ID has 614 unique values. It’s just the unique loan application ID. Target variable “Loan\_Status” in the dataset has two unique values.

**Count of the unique occurrences of Loan\_Status column:**

****

This shows that Loan\_ID has 614 The .value\_counts() method will give you the count of the unique occurrences. Here is the output:



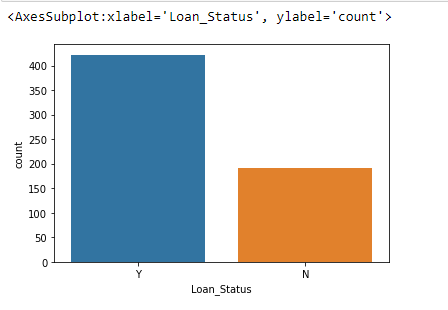
Result of .value\_counts

As we can see, there are **422 occurrences of approved loan applications** and **192 occurrences of rejected loan applications** in the dataset.

**Now let’s visualize the count of approved and rejected loan applications using Seaborn:**

****

The .countplot() method is used to show the counts of observations in each categorical bin using bars. Here is the output:



Count plot to visualize the count of approved and rejected applications

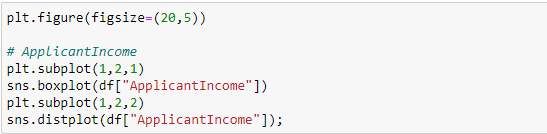
From the count plot, we see that **the dataset is imbalanced.**

**Exploratory Data Analysis:**

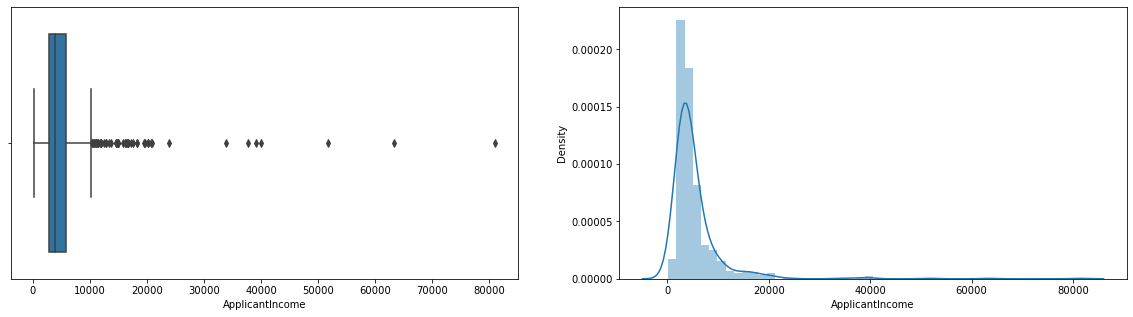
EDA is one of the crucial step in data science that allow us to achieve certain insights and statistical measures. The main purpose of EDA is to help look at data before making any assumptions. It can help identify obvious errors, as well as better understand patterns within the data, detect outliers, find interesting relations among variables.

* **Univariate Analysis:** Univariate analysis is the simplest form of analyzing data. “Uni” means “one”, so in other words your data has only one variable.

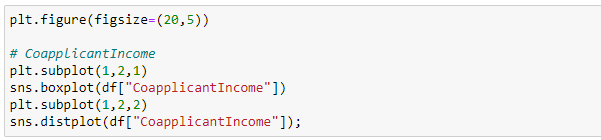
1. **Applicant Income:**

****

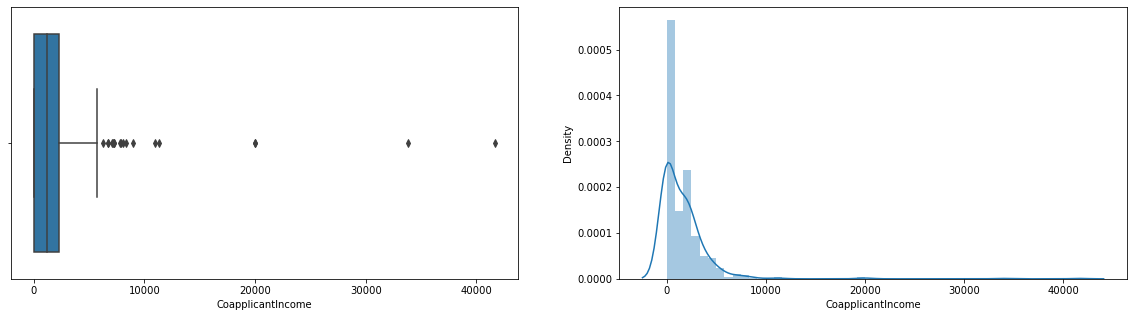
Here is the output:

****

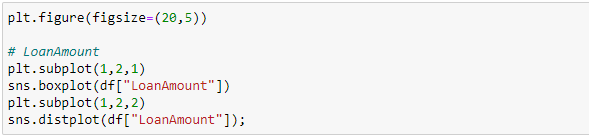
1. **CoapplicantIncome**

****

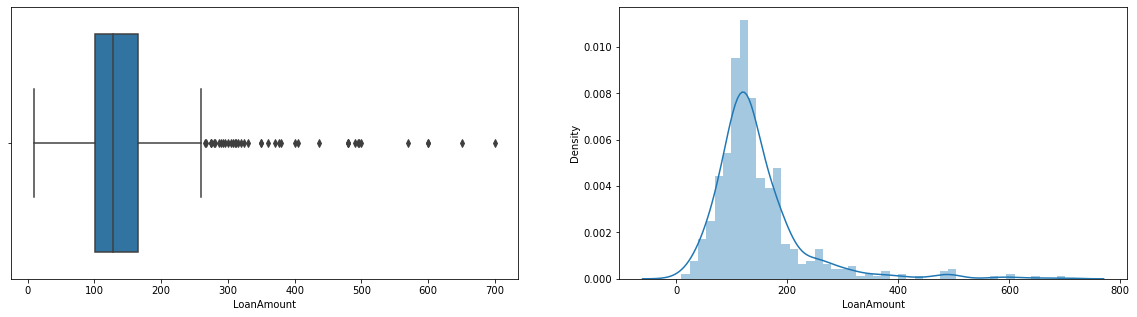
Here is the output:

****

1. **LoanAmount:**

****

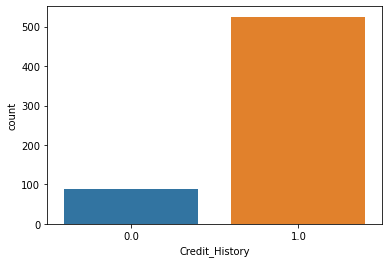
Here is the output:

****

1. **Credit\_History:**

****

Here is the output:

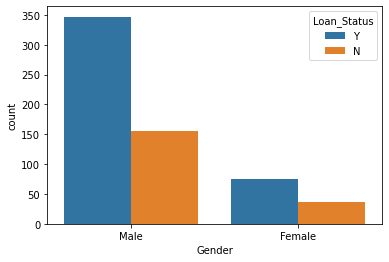
****

* **Bivariate Analysis:** Bivariate analysis means the analysis of bivariate data. It is used to find out if there is a relationship between two sets of values.

1. **Gender vs Loan\_Status:**

****

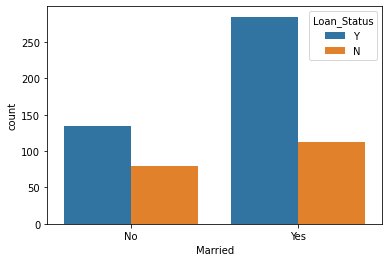
Here is the output:

****

1. **Married vs Loan\_Status:**

****

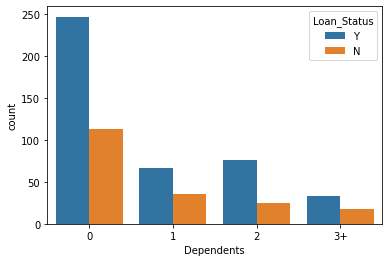
Here is the output:

****

1. **Dependents vs Loan\_Status:**

****

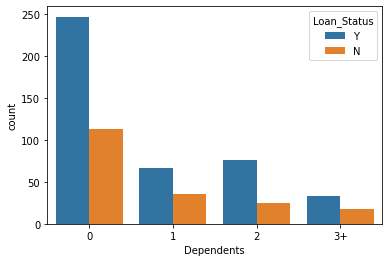
Here is the output:

****

1. **Education vs Loan\_Status:**

****

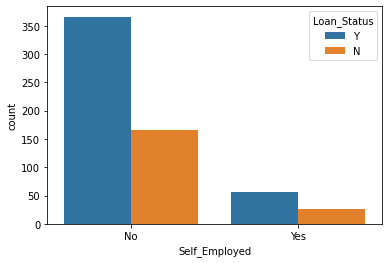
Here is the output:

****

1. **Self\_Employed vs Loan\_Status:**

****

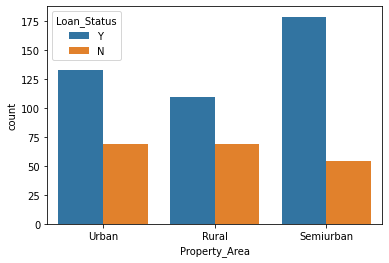
Here is the output:

****

1. **Property\_Area vs Loan\_Status:**

****

Here is the output:

****

**Inferences drawn from the above carried analysis are:**

1. It is clearly visible from the figure that Applicant’s Income is not normally distributed with a fair amount of anomalies and outliers present.
2. Co-applicant’s Income too is not normally distributed and there are many extreme outliers present.
3. However, the Loan Amount figure depicts that the density of Loan Amount is somehow normally distributed, though, it still carries outliers that needs attention.
4. The majority of the loan applicants have a good previous credit history with the bank.
5. Loan application approval ratio is almost same for Male and Female. That means there exists no such major bias by the bank in approving loans on Gender grounds.
6. Married people and people with two dependents have a great chance of their loans get approved.
7. Quite obvious, graduates are preferred by the bank in providing loans, however, no such preference exists for self\_employed applicants.
8. People residing in Semi-Urban localities tend to lure the bank more as compared to urban and rural residing people in providing loans.

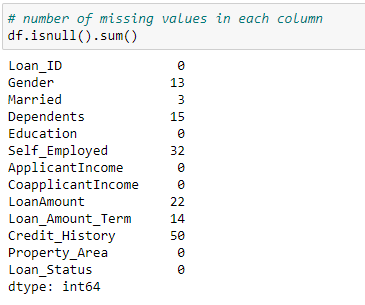
**Data Preprocessing and Feature Engineering:**

Data preprocessing is an integral steps in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn, therefore, it is extremely important that we preprocess our data before feeding it into our model.

Feature engineering is the process of creating new input features from raw data for machine learning. Not only does feature engineering prepare the dataset to be compatible with the algorithm, but it also improves the performance of the machine learning models.

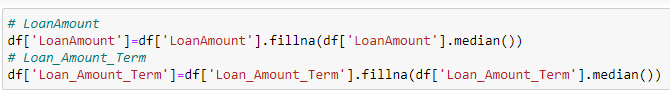
* **Missing Values:**

Initially, we check for null or NaN occurrences in our dataset. It is a good practice to identify and replace missing values for each column in your input data prior to modelling the prediction task.



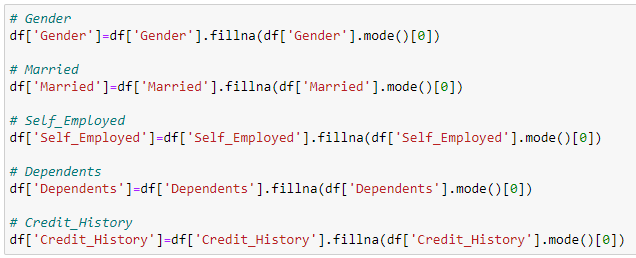
We can see that in our dataset many columns has null values, some are categorical and some are numerical. Let’s fill these null values column wise.

* *LoanAmount, Loan\_Amount\_Term:* We have observed that the distribution of LoanAmount and Loan\_Amount\_Term is not normal. Hence, imputing the features’ null values with their respective median value.

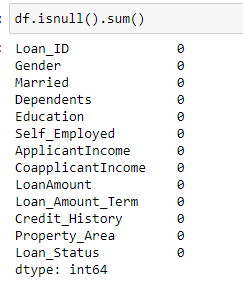


Rest other columns having NaN values are categorical. So let’s impute them with their respective modes.

* *Gender, Married, Self\_Employed, Dependents, Credit\_History:*

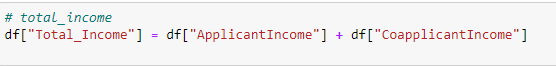


All the NaN values are filled. Now, let’s confirm this so that we can proceed ahead:



* **Creation of New Attribute from the Existing one:**

Here the ApplicantIncome and CoapplicantIncome can be added together to create a new attribute as they both are from the same family. Introducing a new feature as Total\_Income = ApplicantIncome + CoapplicantIncome as a feature engineering measure to optimize feature importance.

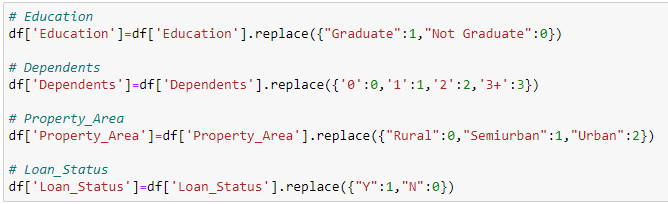


* **Encoding:**

Machine learning models require all input and output variables to be numeric. We need to convert our categorical variables to numbers such that the model is able to understand and extract valuable information. There are two kinds of categorical data:

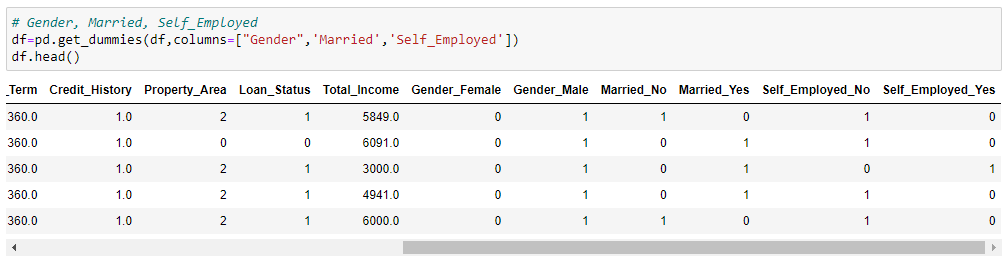
1. ***Ordinal Data:*** The categories have an inherent order. We use **Label Encoding technique** when the categorical feature is ordinal. In this case, retaining the order is important. Hence, the sequence should reflect.

In our dataset, we have ordinal data for the columns, *Education, Dependents, Property\_Area and Loan\_Status.* So let’s perform Label Encoding maintaining a sequence for their values:



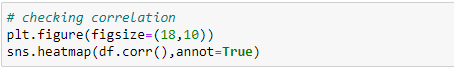
1. ***Nominal Data:*** The categories do not have an inherent order. We use **One Hot Encoding technique** when the features are nominal (do not have any order). In one hot encoding, for each level of a categorical feature, we create a new variable. Each category is mapped with a binary variable containing either 0 or 1.

We have nominal data for the columns Married, Gender, Self\_Employed. Let’s perform One Hot Encoding for our nominal data:



**Correlation Matrix**:

Correlation is an indication about the changes between two variables. Correlation explains how one or more variables are related to each other. These variable can be input features which have been used to forecast our target variable.

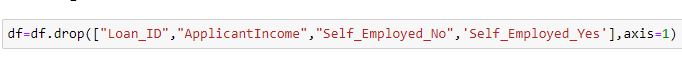


Here is the output:



**Inferences drawn from the above correlation matrix:**

* The correlation matrix states a quite evident fact that Loan Approval Status is strongly correlated with the credit history of the people in the past.
* Loan Approval Status is least correlated with Self\_Employed\_Yes, Self\_Employed\_No and ApplicantIncome.
* Total\_Income is strongly correlated with the other independent variables ApplicantIncome, CoapplicantIncome and LoanAmount.
* **Deleting Unwanted Columns not required during Modelling:**

****

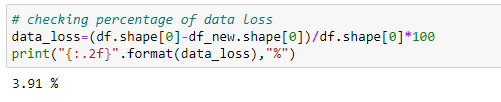
* **Outliers:**

Machine learning algorithms are sensitive to the range and distribution of attribute values. Data outliers can spoil and mislead the training process resulting in longer training times, less accurate models and ultimately poorer results.

As observed from inferences drawn from visualizations, for outliers treatment of LoanAmount, Total\_Amount and CoapplicantIncome we use the zscore method.



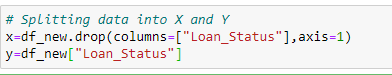
**Percentage of Data Loss after removing outliers**:



We see that there is a data loss of 3.91% after removing outliers.

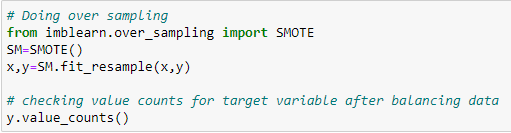
**Creating X and Y Split:**

Splitting the data into x and y. Since we want to predict the status of the loan application, we will drop the “Loan\_Status” column.

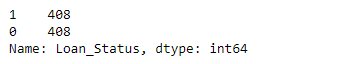
****

**Treating Imbalanced Data:**

As we have observed from visualizations, we have imbalanced dataset as we saw that instances of approved loan applications is higher than the rejected one. An imbalanced dataset will bias the prediction model towards the more common class!



Here is the output:



Great! Now the dataset is balanced and there will not be any biasedness in the prediction model.

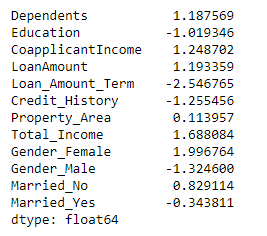
**Skewness:**

Skewness is the measure of the symmetry of a distribution. Skewness degrades the model’s ability as it has to deal with rare cases on extreme values. To ensure that the machine learning model capabilities is not affected, skewed data has to be transformed to approximate to a normal distribution.

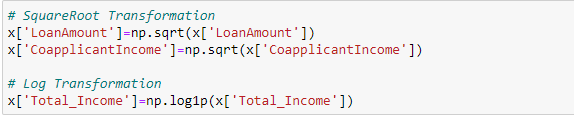
As we have already removed the outliers so with that skewness would also have reduced. So let’s check the skewness first.



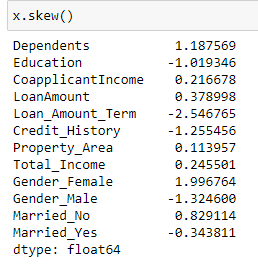
Here is the output:



Ignoring the skewness of the categorical features, we have skewness for *CoapplicantIncome, LoanAmount and Total\_Income*. So let’s treat the skewness using **Log and SquareRoot transformation**.

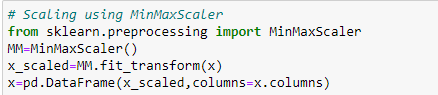


Let’s check the skewness after removal:

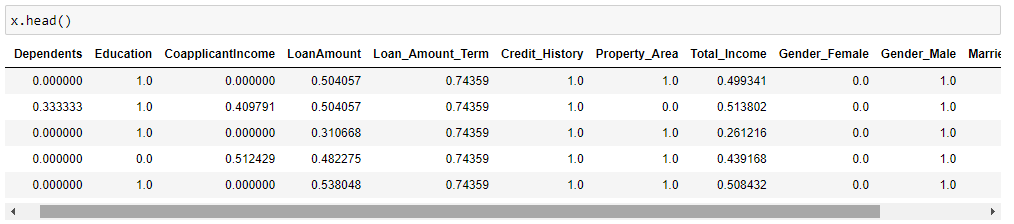


**Scaling**:

To bring all features in the same standing, we need to do scaling so that one significant number doesn’t impact the model just because of their magnitude. If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values.

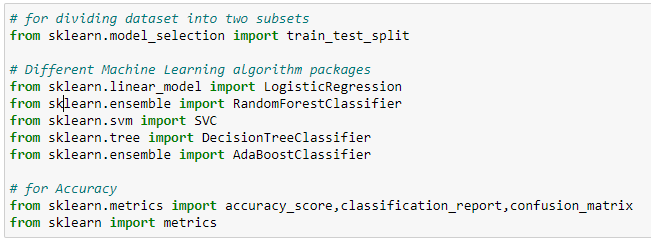


Here is the output:



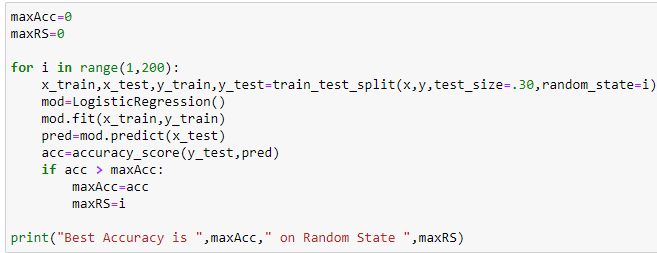
**Building the Model:**

Let’s begin with importing all the required packages essential for our model building. All these packages belongs to our Scikit-learn library.



**Finding Best Random\_State:**

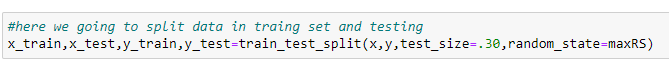
random\_state as the name suggests, is used for initializing the internal random number generator, which will decide the splitting of data into train and test data. If you do not use a random\_state in train\_test\_split, every time you make the split you might get a different set of train and test data points. Let’s find the best random\_state at which we get highest accuracy:



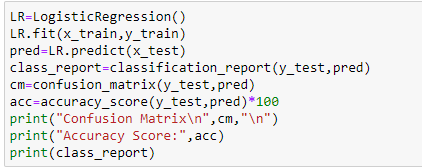
Here is the output:



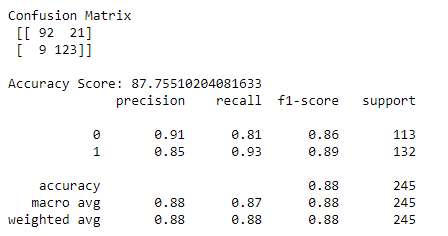
Now, splitting the data into train and test respectively using the best random\_state found.



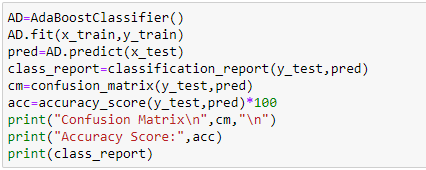
1. **Logistic Regression Classification:**

****

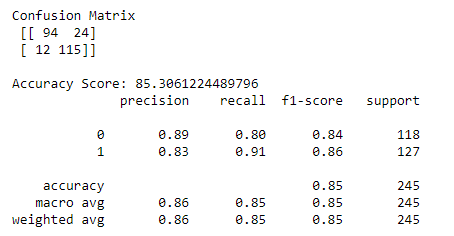
Here is the output:

****

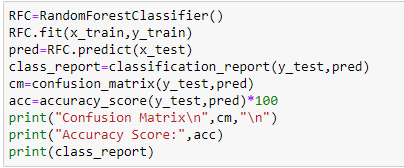
1. **Ada Boost Classifier:**

****

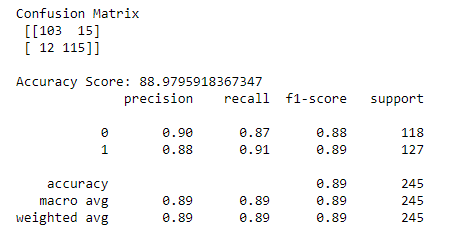
Here is the output:

****

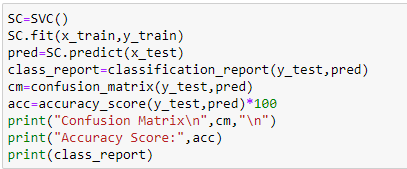
1. **Random Forest Classifier:**

****

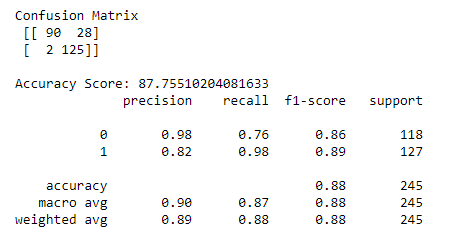
Here is the output:

****

1. **SVC Classifier:**

****

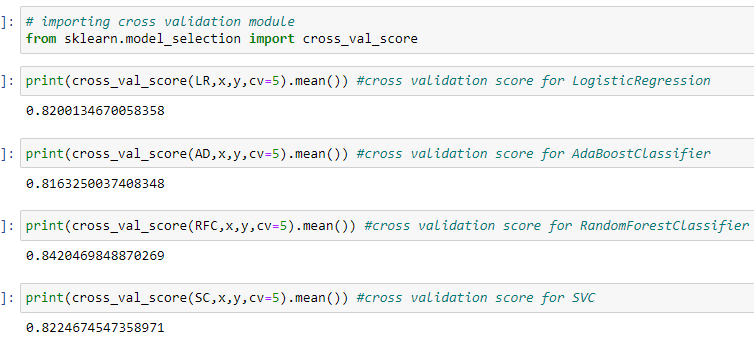
Here is the output:

****

From above all the algorithms, RandomForestClassifier is giving the highest accuracy score of 89%. But it may or may not be due to overfitting, not sure about this. **Overfitting** occurs when the machine learning algorithm captures the noise of the data. **Cross-validation** is a powerful technique to avoid overfitting.

**Cross-Validation:**

Cross-validation is a very useful technique for accessing the effectiveness of machine learning models. It is used to protect against overfitting in a predictive model, particularly in a case where the amount of data is limited.

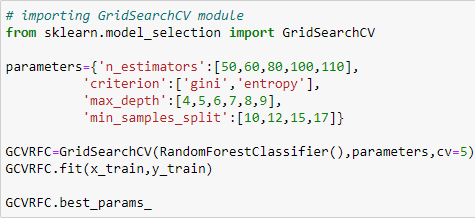


After doing Cross-validation RandomForestClassifier is giving the least difference. So concluding it as the best model.

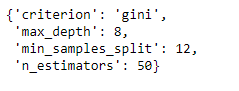
**Hyperparameter Tuning:**

Hyperparameter Tuning finds the best version of a model by running many training jobs on the dataset using the algorithm and ranges of hyperparameters that we specify.

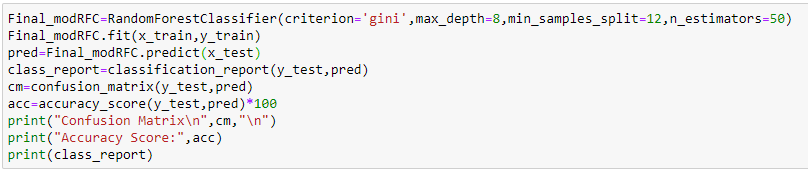
It then chooses the hyperparameter values that result in a model that performs the best.



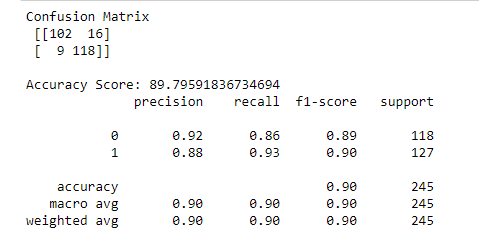
Here are our best hyperparameters:



Now, let’s build our model with the best hyperparameters found.



Here is the output:



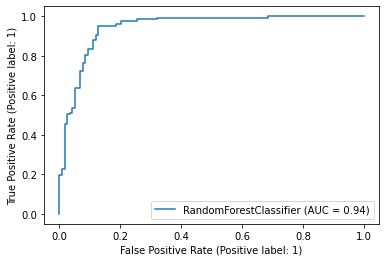
The final model on applying the best hyperparameters are giving a decent accuracy of 89.8%.

**AUC\_ROC Curve:**

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

****

Here is the output of the plot:



The AUC score is 0.94

**Conclusion:**

This was the basic binary classification problem. Here we used regression, tree-based and svm algorithms and out of these RandomForestClassifier stood out the best model for this problem. There may be many more algorithms that would have provided more efficient results. With the hyperparameter tuning we have definitely improved the model performance. This same problem can be solved by many permutations and combinations of different approaches in data science and machine learning. Machine learning and Data Science are a huge ocean of opportunities to try and gamble out various techniques and processes.

**Thank you!!!**