Loan Status Prediction Using Gradient Boost Classifier

Swapnil Gavit

Description:

1. Prediction of Loan Eligibility for Dream Housing Finance company is a Hackathon project on Datahack. (link)
2. This project is implemented using Gradient Boosting Classifier.

Problem:

Company wants to automate the loan eligibility process 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 provided a dataset to identify the customers segments that are eligible for loan amount so that they can specifically target these customers.

Data Source:

Datahack
Download Data Sets:

- Training Data Set
- Testing Data Set

Data Dictionary:

- Training Data:

Train file CSV containing the customers for whom loan eligibility is known as 'Loan_Status'.

Variable	Description
Loan_ID	Unique Loan ID
Gender	Male/ Female
Married	Applicant married (Y/N)
Dependents	Number of dependents
Education	Applicant Education (Graduate/ Under Graduate)
Self_Employed	Self employed (Y/N)
ApplicantIncome	Applicant income
CoapplicantIncome	Coapplicant income
LoanAmount	Loan amount in thousands
Loan_Amount_Term	Term of loan in months
Credit_History	credit history meets guidelines
Property_Area	Urban/ Semi Urban/ Rural
Loan_Status	(Target) Loan approved (Y/N)

- Testing Data:

Test file**:** CSV containing the customer information for whom loan eligibility is to be predicted.

Variable	Description
Loan_ID	Unique Loan ID
Gender	Male/ Female
Married	Applicant married (Y/N)
Dependents	Number of dependents
Education	Applicant Education (Graduate/ Under Graduate)
Self_Employed	Self employed (Y/N)
ApplicantIncome	Applicant income
CoapplicantIncome	Coapplicant income
LoanAmount	Loan amount in thousands
Loan_Amount_Term	Term of loan in months
Credit_History	credit history meets guidelines
Property_Area	Urban/ Semi Urban/ Rural

- Final Result Format:

The final submission file format should be in following manner. The Loan_ID column contains unique loan IDs and the Loan_Status column contain predicted result of loan status.
The final submission file should be in CSV format

Variable	Description
Loan_ID	Unique Loan ID
Loan_Status	(Target) Loan approved (Y/N)

Why Gradient Boosting Classifier?

- It provides predictive scores which is better than other classifiers.
- It often provides predictive accuracy that cannot be trumped.
- Optimize on differentiable loss function.
- Provides several Hyper Parameter Tuning options that make the function fit very flexible.

The Project is divided into Two parts:

1. **Building Machine Learning Model.**
2. **Predicting the Outcomes of Test Dataset**.

Part I: Building ML Model

Step 1st: Importing Important Libraries

```
# Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Step 2nd: Reading Train & Test Data Set

```
# Reading Training Dataset
Data = pd.read_csv(r'Data\TrainData.csv')

# Reading Testing Dataset
Test_Data = pd.read_csv(r'Data\TestData.csv')

# Check Shape of datasets
print('Size of Training Data',Data.shape)
print('Size of Testing Data',Test_Data.shape)
```

Step 3rd: Descriptive Statistics

Descriptive statistic of Quantitative Data

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

Descriptive statistic of Qualitative Data

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Property_Area	Loan_Status
count	614	601	611	599	614	582	614	614
unique	614	2	2	4	2	2	3	2
top	LP001194	Male	Yes	0	Graduate	No	Semiurban	Y
freq	1	489	398	345	480	500	233	422

Step 4th: Data Cleaning

The Missing Values present in data set are:

```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

Imputing Missing Values

Here we created the function for imputing Null values So we can reuse the function for Testing data set as well. In that function we use the pandas fillna method to fill null values. The Gender, Married , Self Employed, and Dependents features are fill with their respective modes and Loan Amount, Loan Amount History, Credit History are fill with medians.

```
# Function for Imputing Missing Values

def Impute_Missing_Val(DataFrame):
    # Imputing Missing values with Mode
    DataFrame['Gender'].fillna(DataFrame['Gender'].mode()[0],inplace=True)
    DataFrame['Married'].fillna(DataFrame['Married'].mode()[0],inplace=True)
    DataFrame['Self_Employed'].fillna(DataFrame['Self_Employed'].mode()[0],inplace=True)
    DataFrame['Dependents'].fillna(DataFrame['Dependents'].mode()[0],inplace=True)

    # Imputing Missing values with Median
    DataFrame['LoanAmount'].fillna(DataFrame['LoanAmount'].median(),inplace=True)
    DataFrame['Loan_Amount_Term'].fillna(DataFrame['Loan_Amount_Term'].median(),inplace=True)
    DataFrame['Credit_History'].fillna(DataFrame['Credit_History'].median(),inplace=True)
    return Data
```

Imputing the Null values using function Impute_Missing_Val and check if there are null values present in data after calling a function

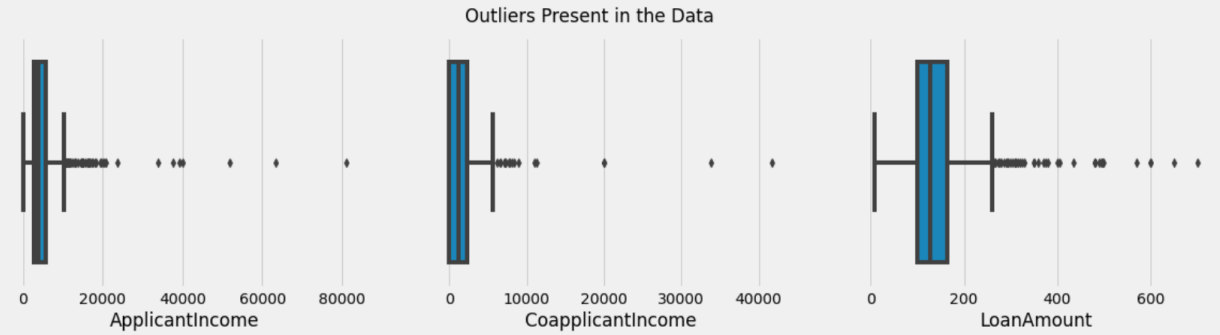
```
# Lets Impute Missing values in Training Data using Function
Impute_Missing_Val(Data)

# Check if Missing values are present in Data
Data.isnull().sum()
```

Total Null values are zero after calling a function

Visualize and Remove Outliers

Visualize the Applicant Income, Co-applicant Income and Loan Amount feature of Data using seaborn boxplot method.



Remove Outliers

As we see in above plot too many outliers present in the data and we removed them as follows

1. Removed customers having Applicant Income more than 25000.
2. Removed customers having Co-Applicant Income more than 10000.
3. Removed customers having Loan Amount more than 400.

```
# Removing Outliers

# Check Shape of data before removing outliers
print('Shape of Data before removing outliers:',Data.shape)

# Remove the customers having more than 25000 income
Data=Data[Data['ApplicantIncome']<25000]

# Remove the customers having more than 10000 coapplicant income
Data=Data[Data['CoapplicantIncome']<10000]

# Remove the customers having more than 400 loan amount
Data=Data[Data['LoanAmount']<400]

# check the shape of data after removing outliers
print('Shape of Data after removing outliers:',Data.shape)
```

We check the shape of the data before and after removing the data

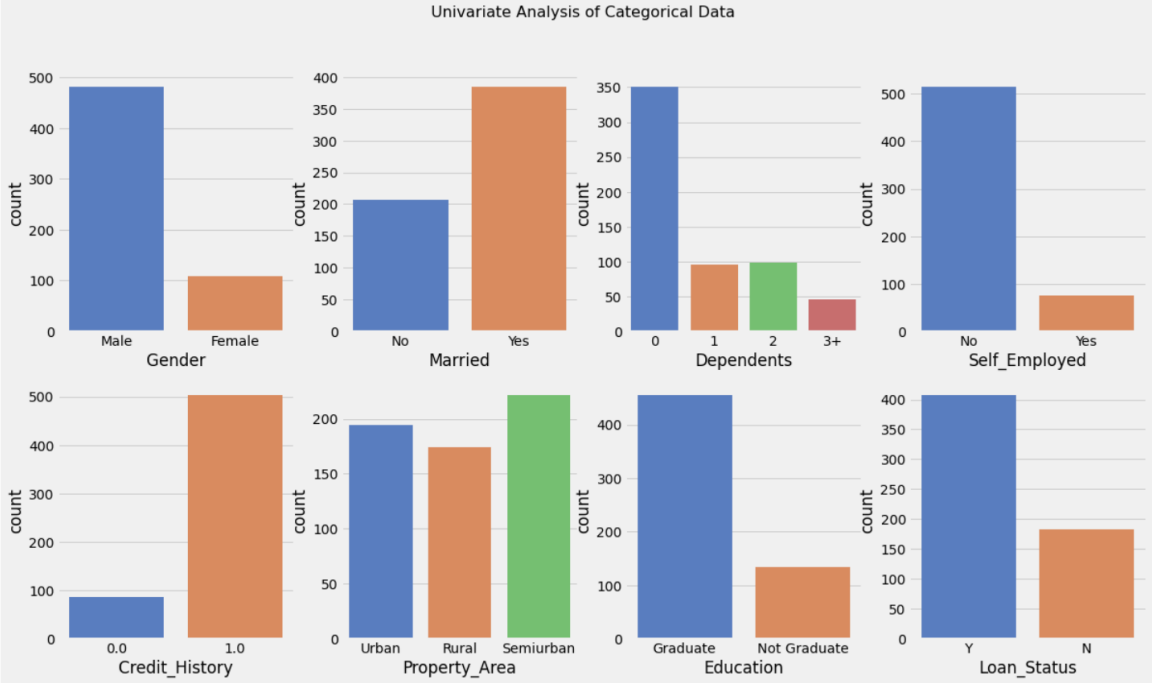
Shape of data before and after removing outliers

```
Shape of Data before removing outliers: (614, 13)
Shape of Data after removing outliers: (590, 13)
```

Step 5th: Univariate Data Analysis

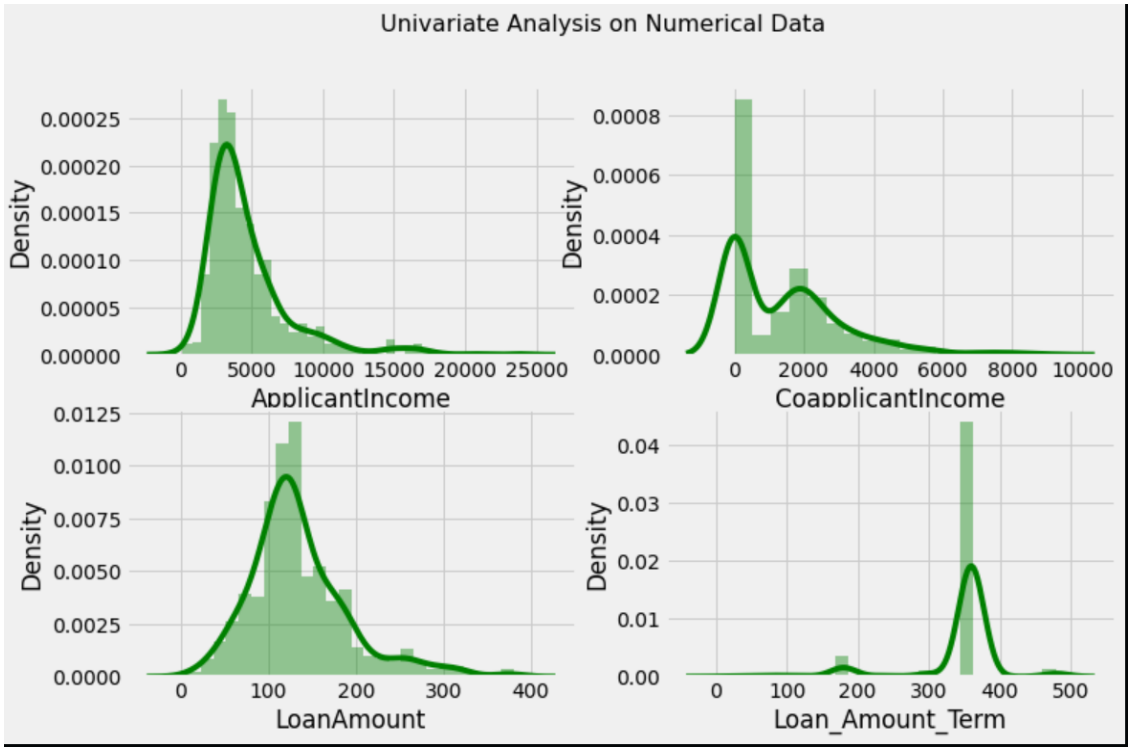
1. On Qualitative Features of Data

Visualize the different categorical features of data using coutnplot



2. On Quantitative Features of Data

Visualize the quantitative features of the data using distplot



As we see in above plot the data is highly skewed.

Check the skewness of data

```
# Lets measure the skewness of Data
Data.skew()
```

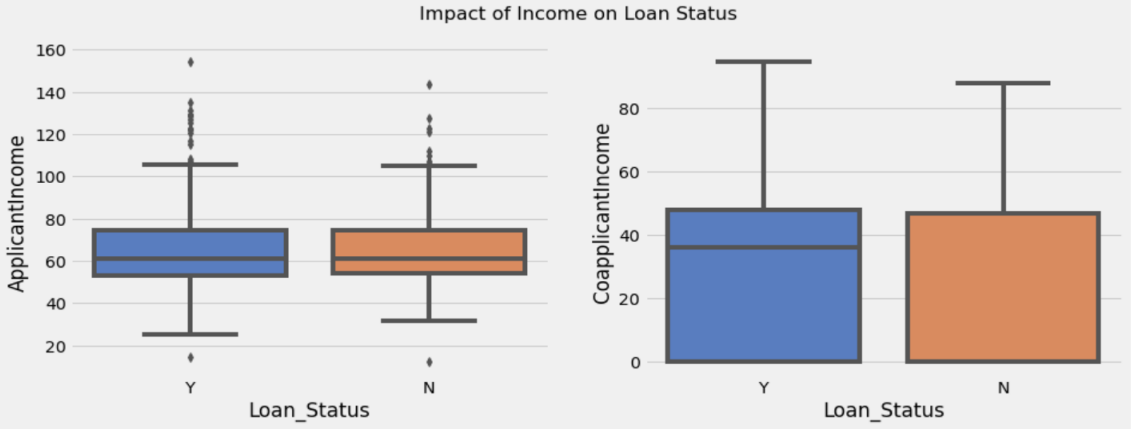
```
ApplicantIncome      2.344116
CoapplicantIncome    1.355108
LoanAmount           1.165654
Loan_Amount_Term    -2.448984
Credit_History      -2.012881
dtype: float64
```

The skewness of data is high so we removed in the data preparation

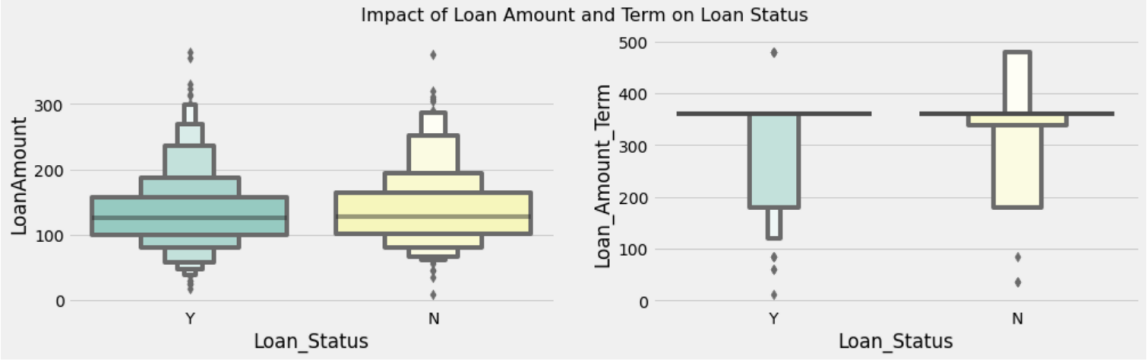
Step 6th: Bivariate Data Analysis

Bivariate Analysis on Quantitative Features

Visualize the impact of Applicant Income and Co-applicant Income of the Loan Status

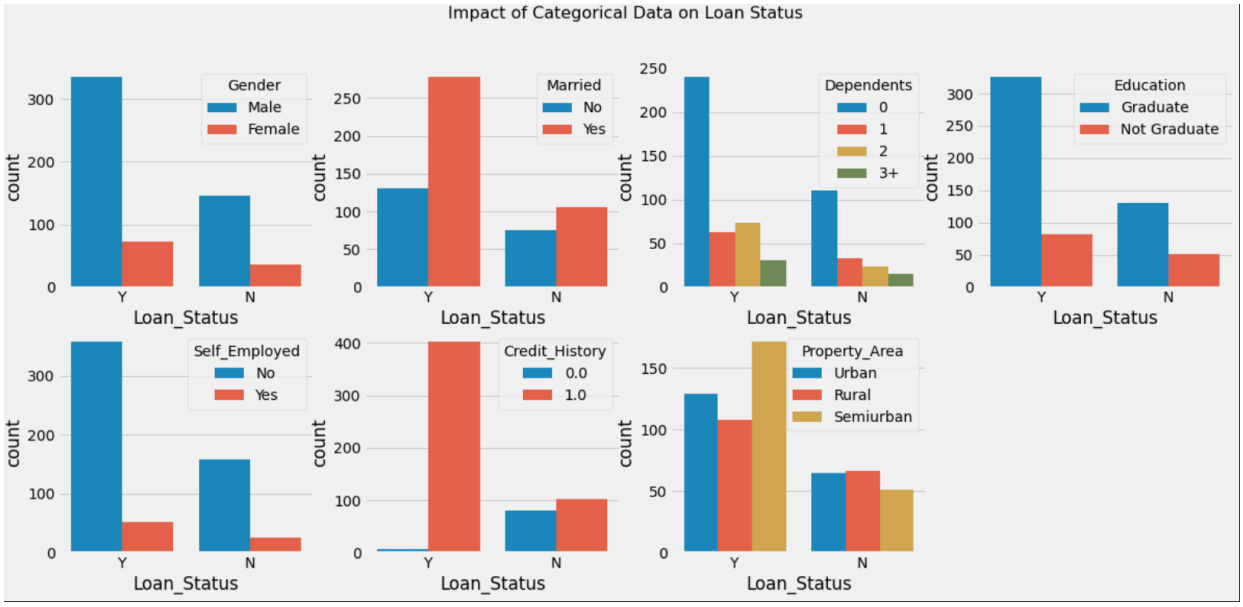


Visualize the Impact of Loan Amount and Loan Amount Term on Status of Loan



Bivariate Analysis on Qualitative Features

Visualize the relation between different categories of Data with Target variable (Loan_Status)



As we see in above chart the Credit history is high relation with Loan Status

Step 7th: Data Preparation

Data Encoding

We check the all unique values of the categorical columns of data so we can replace them using pandas replace method. Here we created the function to encode the data so we can use it later for Testing Data.

```
# Function for Numeric Conversiton

def Numeric_conversion(DataFrame):
    DataFrame['Gender'].replace(('Male','Female'),(1,0),inplace=True)
    DataFrame['Married'].replace(('Yes','No'),(1,0),inplace=True)
    DataFrame['Education'].replace(('Graduate','Not Graduate'),(1,0),inplace=True)
    DataFrame['Self_Employed'].replace(('Yes','No'),(1,0),inplace=True)

    # Urban and Semi Urban Property have very similar Impact on Loan Status, so, we will merge them together
    DataFrame['Property_Area'].replace(('Urban','Semiurban','Rural'),(1,1,0),inplace=True)

    # As seen above that apart from 0 dependents, all are similar hence, we merge them to avoid any confusion
    DataFrame['Dependents'].replace(('0','1','2','3+'),(0,1,1,1),inplace=True)
    return DataFrame
```

```
# Test Data isn't contain Loan_Status column So we put it outside of function
Data['Loan_Status'].replace(('Y','N'),(1,0),inplace=True)
```

Set Target Column

We know that Loan Status is our Target column so set Y (dependent variable) and X (independent variable)

```
# Split the Target column from the Data
X = Data.drop(['Loan_Status'],axis=1)
Y = Data['Loan_Status']

# Check the Shape of X and Y
print('Shape of X:', X.shape)
print('Shape of Y:', Y.shape)
```

Resembling Data

It is very important to resample the data, as the Target class is Highly imbalanced. Here We are going to use Over Sampling Technique to resample the data. Use SMOTE algorithm to do the same.

```
# Import the SMOTE algorithm to do the same.
from imblearn.over_sampling import SMOTE
x_res, y_res = SMOTE().fit_resample(X, Y)

# Check Shape of X and Y after resampling it
print('Shape of X:',x_res.shape)
print('Shape of X:',y_res.shape)
print('\n')
# Check the value counts of target variable
print("Before Resambling :")
print(Y.value_counts())
print("\n After Resambling :")
print(y_res.value_counts())
```

Check the data begore sampling and after sampling

```
Before Resambling :
1.0    408
0.0    182
Name: Loan_Status, dtype: int64

 After Resambling :
0.0    408
1.0    408
Name: Loan_Status, dtype: int64
```

Step 8th: Data Splitting

Split the data to Training set and Testing set for building a model and to check the accuracy of model

```
# Split the Data for Training Model and Testing Model
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x_res,y_res, test_size=0.20,
                                                    random_state=0,shuffle= True, stratify=y_res)
```

Step 9th: Applying ML Model

Creating a Gradient Boosting Classifier model and fit the data to training of model

Predict the output of test data set and store it to a variable

```
# Apply Gradient Boosting Classifier
# Import Grdient Boosting Classifier from sklearn
from sklearn.ensemble import GradientBoostingClassifier

# Create  ML Model and fit the training data
GBC = GradientBoostingClassifier()
GBC.fit(x_train, y_train)

# Predict Output and Store it
y_pred = GBC.predict(x_test)
```

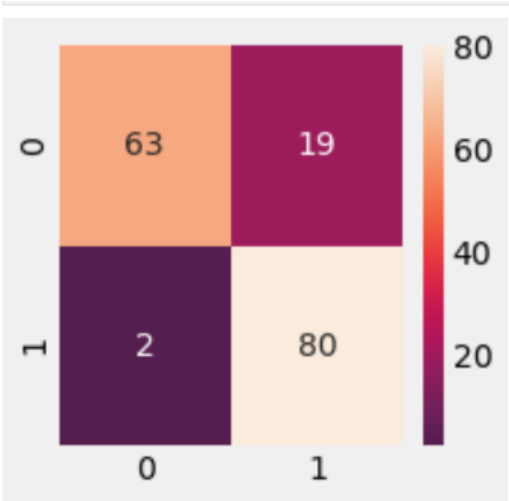
Step 10th: Check the Accuracy of Model

Accuracy: 87.20%

```
# Accuracy
from sklearn.metrics import accuracy_score
print('Accuracy : %s '%'{0:.2%}'.format(accuracy_score(y_test, y_pred)))
```

```
Accuracy : 87.20%
```

Confusion-Matrix:



Classification Report:

	precision	recall	f1-score	support
0.0	0.97	0.77	0.86	82
1.0	0.81	0.98	0.88	82
accuracy			0.87	164
macro avg	0.89	0.87	0.87	164
weighted avg	0.89	0.87	0.87	164

Cross-Validation score: 82.52%

```
[0.78787879 0.81818182 0.84615385 0.84615385 0.84615385 0.8
 0.87692308 0.78461538 0.8        0.84615385]

 Cross-Validation Score :82.52%
```

Part II: Predict the Outcomes of Test Data

Step 1st: Check Testing Data Set

Input Data:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	360.0	1.0	Urban
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	1.0	Urban
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	360.0	1.0	Urban
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	360.0	NaN	Urban
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	360.0	1.0	Urban

Shape: (367,12)

Total Null Values present in Testing data: 84

Step 2nd: Transform Data

Total Null Values are 84

Impute missing values using function the function Imput_Missing_Val which was created earlier.

Convert object type features to numeric type features using the function Numeric_Conversion which was created earlier.

Perform sqare root transformation to transform the data

```
# Impute Missing Values using function which was created earlier
Impute_Missing_Val(Test_Data)
print('Null Values are:',Test_Data.isnull().sum().sum())

# Convert To Numeric Values using function which was created earlier
Numeric_conversion(Test_Data)
print('Object Data Type:',Data.select_dtypes('object').columns)

# Store Loan ID
loan_id=Test_Data['Loan_ID']

# Test Data Transformation
Test_Data = np.sqrt(Test_Data.drop('Loan_ID',axis=1))
```

```
Null Values are: 0
Object Data Type: Index([], dtype='object')
```

Step 3rd: Apply ML Model

Apply ML model and fit the data to predict the outcomes

The output is in numpy array format so we convert it in pandas DataFrame

```
# Predict Target Variable and Store it
result = GBC.predict(Test_Data)

# Convert to pandas DataFrame
result=pd.DataFrame(result,columns={'Loan_Status'})
```

The output is in numeric type so we convert it back to normal Y and N type and set index as Loan ID

```
# Replace the Values of 1 and 0 by Y and N
result['Loan_Status'].replace((1, 0),('Y', 'N'),inplace=True)

# Set Index as Loan_ID which was stored in loan_id earlier
result.set_index(loan_id, inplace=True)

# Check the Head of Data
result.head()
```

Step 4th: Save the Result

The result is saved in CSV format using pandas**.**

```
# Store the Final result
result.to_csv(r'Output\Result.csv')
```

Result:

Loan_ID	Loan_Status
LP001015	Y
LP001022	Y
LP001031	Y
LP001035	Y
LP001051	Y