Amar Shah **180036**

Big Data & Data Analytics – II

W9 – Project Activity-2

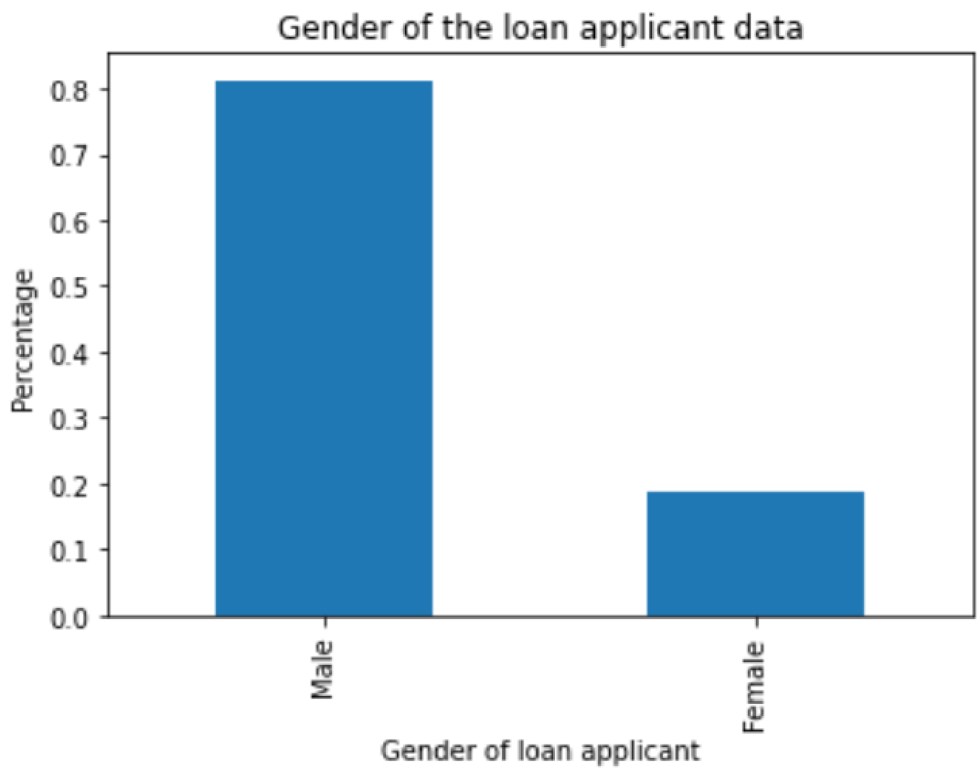
Loan Approval Prediction Project

https://github.com/SenseiAlerai/Loan-Approval-Prediction-Project-

Section: A (Exploratory Data Analysis)

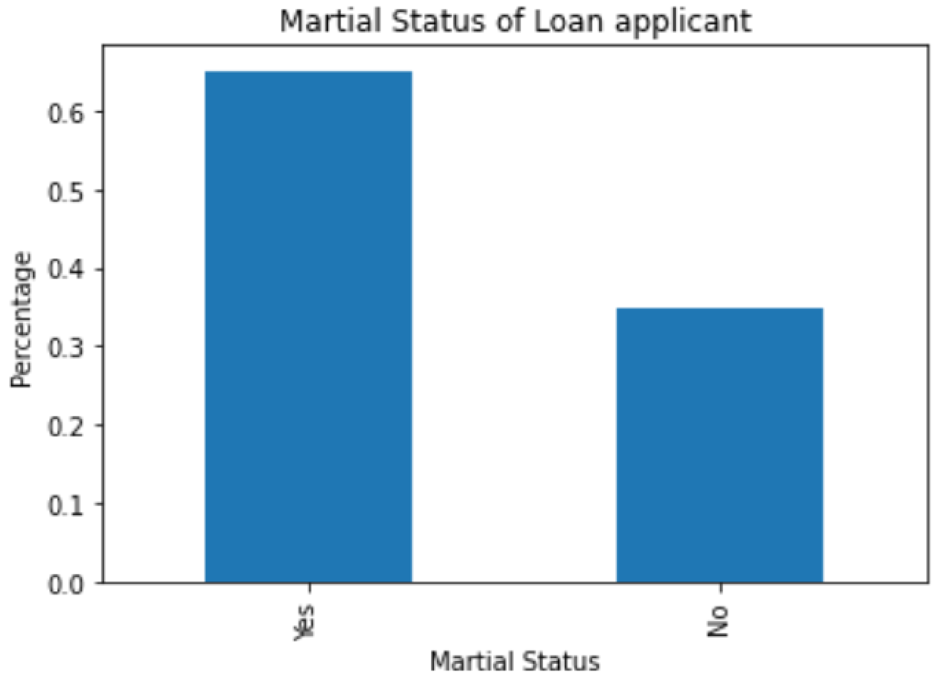
1. **Let us analyse and visualize the categorical attribute of the given train dataset using single variable.**

   i. **Find out the number of male and female in loan applicants' data.**



   There are 81% Male & 19% Female in loan application.

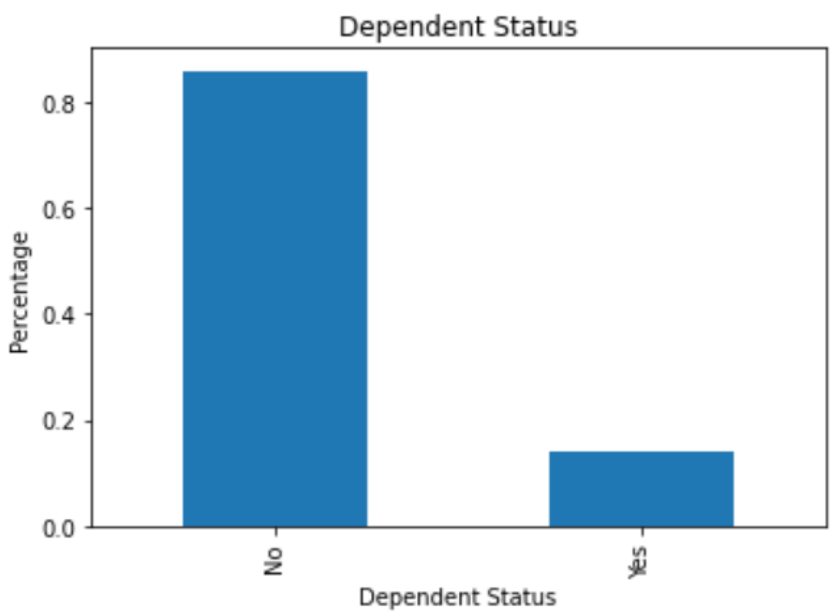
   ii. **Find out the number of married and unmarried loan applicants.**



Number of married people: 65%

Number of unmarried people: 35%

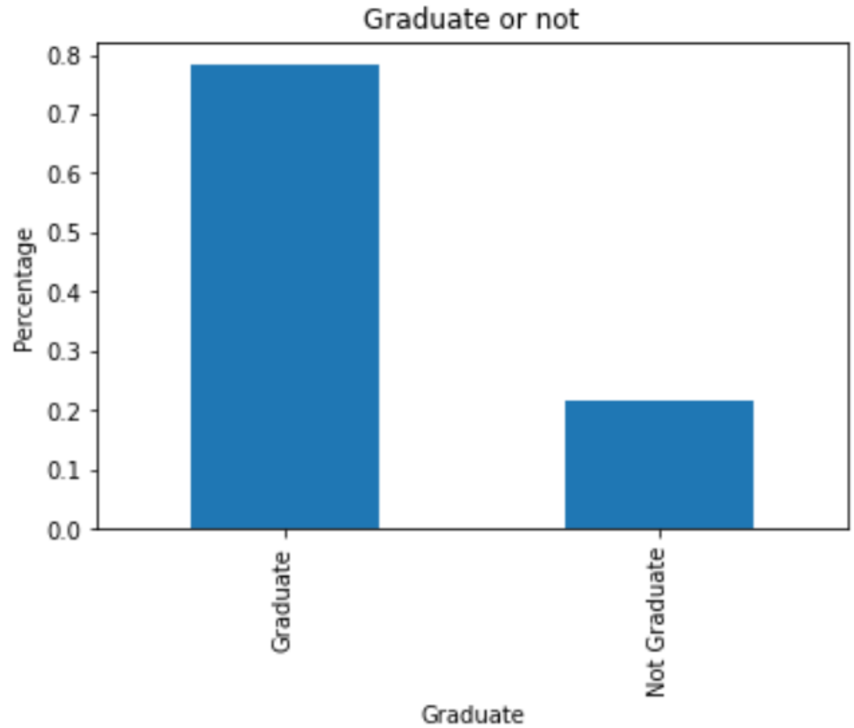
iii. Find out the overall dependent status in the dataset.



In a total of 582 people - 14% are Self-employed and - 86% are Not Self-employed

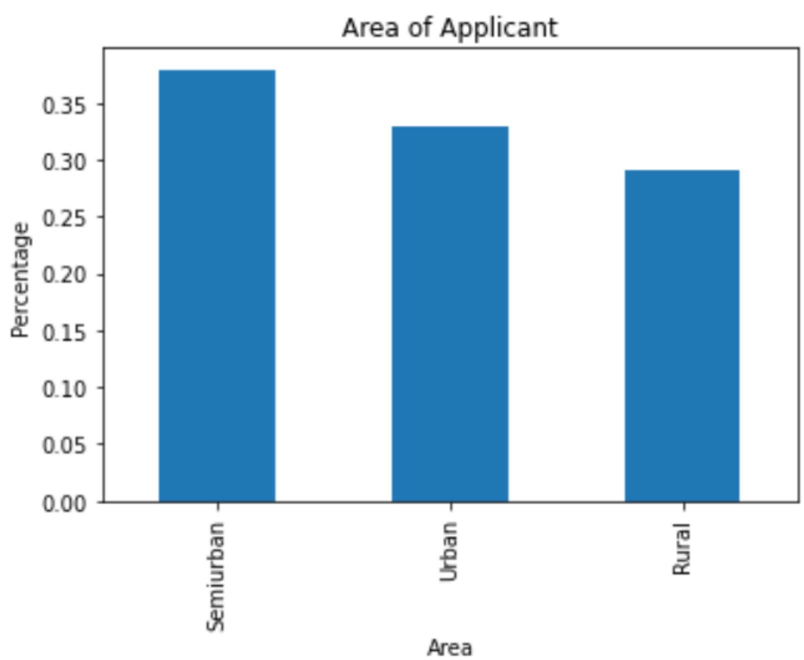
Amar Shah **180036**

iv. Find the count how many loan applicants are graduate and non-graduate.



78% are Graduated 22% are not Graduated

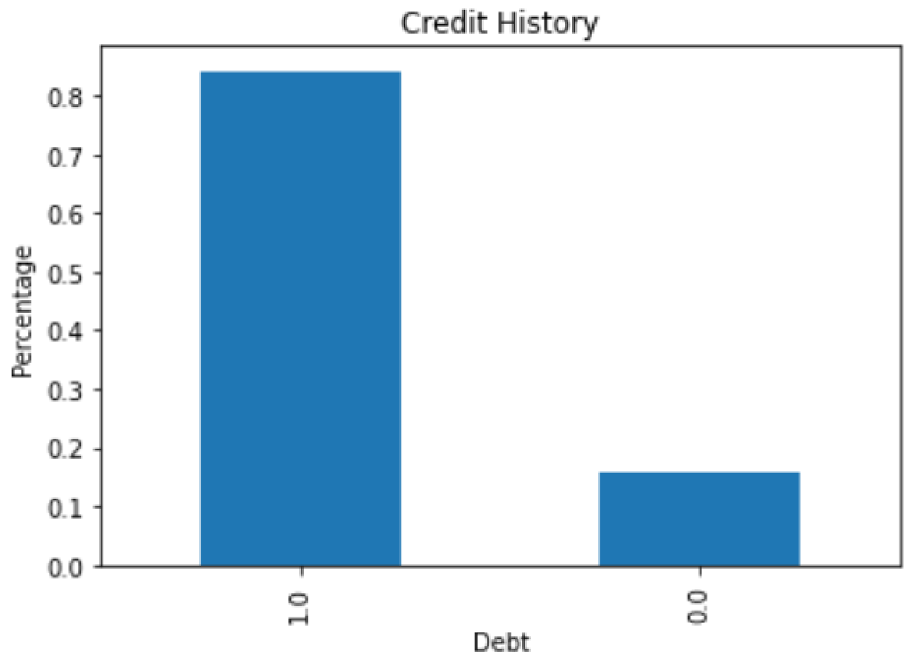
v. Find out the count how many loans applicants property lies in urban, rural, and semi-urban areas.



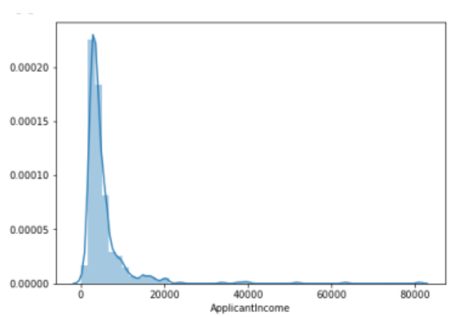
Applicants from Semiurban area = 38%, Applicants from Urban area = 33% & Applicants from Rural area = 29%

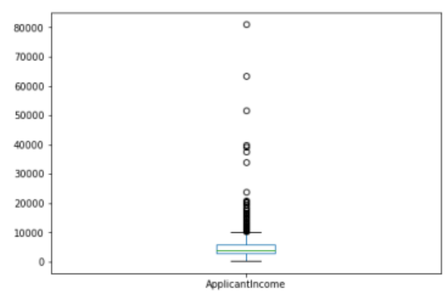
Amar Shah **180036**

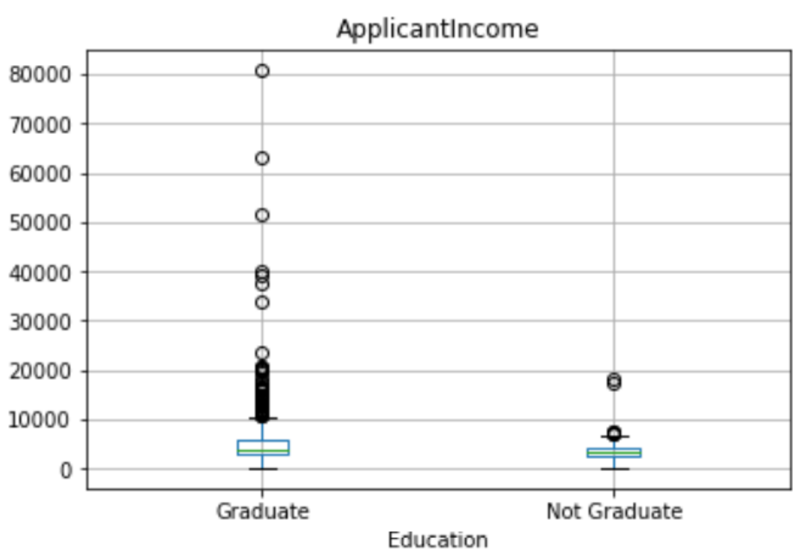
2. **What conclusions are you derived from the single variable analysis?**

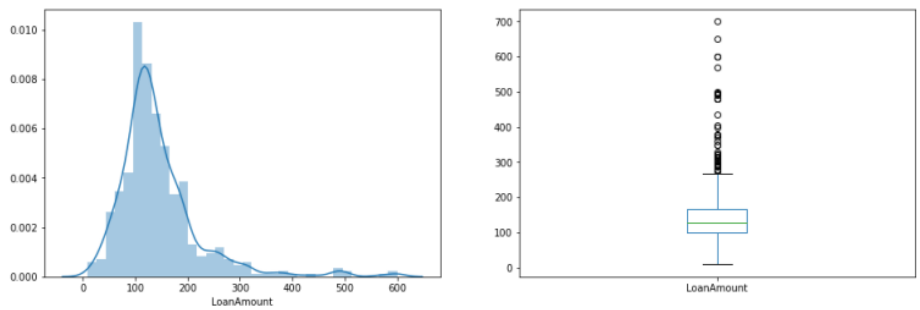


3. **Also visualize and plot the Question-1 based on Loan status of loan applicant (Multi variable analysis).**

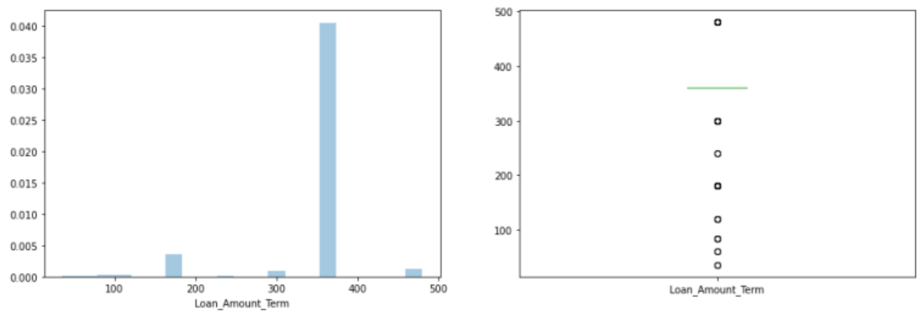








Loan Amount Distribution



4. What conclusions are you derived from the multi variable analysis?

Conclusion from Relation between Loan Status and Gender

Female whose Loan was approved = 75

Male whose Loan was approved = 339

Female whose Loan was not0 approved = 37

Female whose Loan was approved = 150

We can observe that the proportion of Male applicants is higher for the app roved loans.

Conclusion of relation between Loan_Status and Married status

Married people whose Loan was approved = 285

Married people whose Loan was not approved = 113

Unmarried people whose Loan was approed = 134

Unmarried people whose Loan was not approed = 79

We can observe that the proportion of Married applicants is higher for the approved loans.

Conclusion of relation between Loan_Status and Dependents

Number of dependents on the loan applicant

0 and Loan was approed : 238

0 and Loan was not approed : 107

1 and Loan was approed : 66

1 and Loan was not approed : 36

2 and Loan was approed : 76

2 and Loan was not approed : 25
3+ and Loan was approed : 33
3+ and Loan was not approed : 18
We can observe that the distribution of applicants with 1 or 3+ dependents is similar across both the categories of Loan Status.

Conclusion of relation between Loan Status and Education.
People who are Graduate and Loan was approved: 340
People who are Graduate and Loan was no approved: 140
people who are Not Graduate and Loan was approved: 82
People who are Not Graduate and Loan was not approved: 52
We can observe that the proportion of Graduate applicants is higher for the approved loans

Conclusion from Relation between Loan Status and Self-employed
People who are Self-employed and Loan was approved: 56
People who are Self-employed and Loan was not approved: 26
People who are not Self-employed and Loan was approved: 343
People who are not Self-employed and Loan was not approved: 157
There is nothing that we can signify and infer from Self-employed vs Loan_ Status plot.

Section: B (Decision Tree Classifier)

1. Building a Decision Tree Classifier in Python using Scikit-learn Library

We'll now predict if a consumer is likely to eligible for loan amount using the decision tree algorithm in Python. The data set contains a wide range of information for making this prediction, including the gender, married, dependents, education, self-employed, applicant_income, co-applicant_income, loan_amount, loan amount term, credit_history, property_area and whether the individual was eligible for loan amount (i.e. loan_status). The following steps should be followed during building a decision tree classifier:

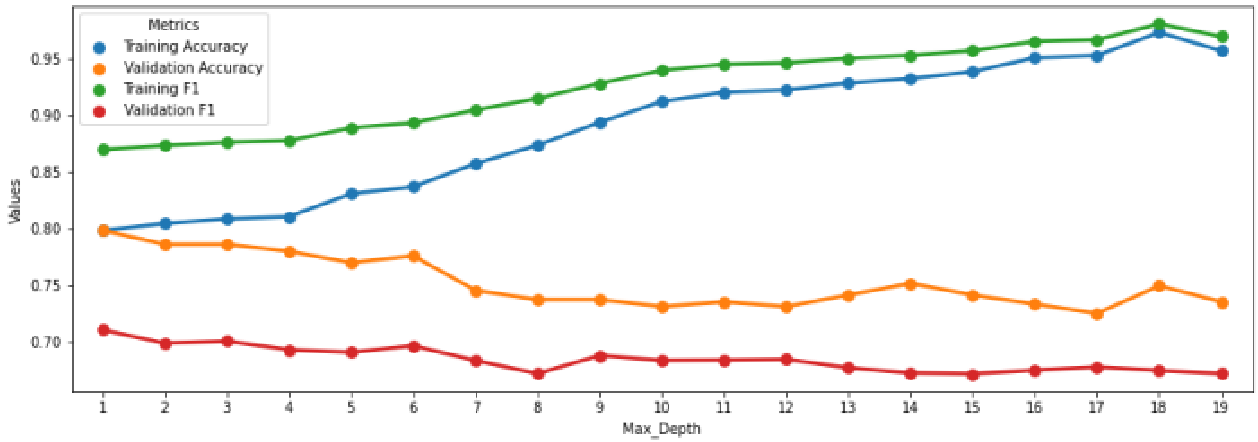
1. **Import the libraries required to build a decision tree in Python.**
2. **Load the train dataset and test dataset using the read_csv () function in pandas.**
3. **Data Cleaning: Preprecessing of both dataset.**
    a. **Missing Values: Check where there are missing values and fix them appropriately.**
4. **Feature Selection: Separate the independent and dependent variables using the slicing method.**

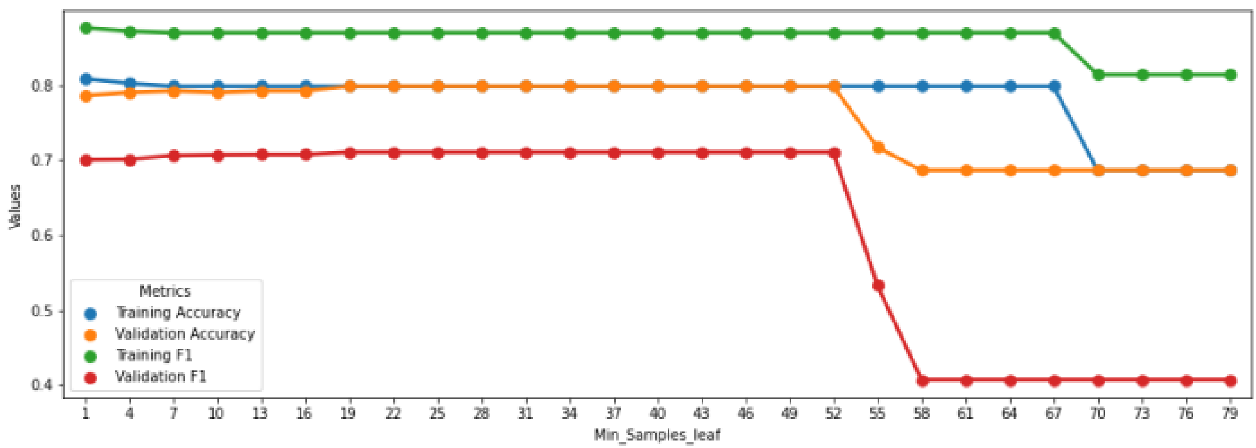
5. **Encoding to numeric data: Convert each of the categorical variables in to numeric data for modeling. For handling categorical variables, there are many methods like One Hot Encoding or Dummies.**
6. **Splitting Data: Split the data into training and testing sets.**
7. **Building Decision Tree Model: Train the model using the decision tree classifier.**
8. **Evaluating Model: Predict the test data set values using the model above.**
9. **Calculate the accuracy of the model using the accuracy score function.**
10. **Visualizing Decision Trees**

```
shape: Test dataset   (367, 12)
 shape: Train dataset  (614, 13)
Null values in Train dataset
Null values in Train data set
Null values in Test data set
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             614 non-null    object
 2   Married            614 non-null    object
 3   Dependents         614 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      614 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         614 non-null    float64
 9   Loan_Amount_Term   614 non-null    float64
 10  Credit_History     614 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
```

```
 0   Loan_ID            367 non-null    object
 1   Gender             367 non-null    object
 2   Married            367 non-null    object
 3   Dependents         367 non-null    object
 4   Education          367 non-null    object
 5   Self_Employed      367 non-null    object
 6   ApplicantIncome    367 non-null    int64
 7   CoapplicantIncome  367 non-null    int64
 8   LoanAmount         367 non-null    float64
 9   Loan_Amount_Term   367 non-null    float64
 10  Credit_History     367 non-null    float64
 11  Property_Area      367 non-null    object
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB
Encoding categrical variable
Split data Features and Target Varible
Splitting into train and test Data
handling Missing values
Training Data Set Accuracy:  1.0
Training Data F1 Score  1.0
Validation Mean F1 Score:  0.6742937089861218
Validation Mean Accuracy:  0.7393320964749537
Test Accuracy:  0.8536585365853658
Test F1 Score:  0.9032258064516
Confusion Matrix on Test Data
```

Predicted	0	1	All
True			
0	21	17	38
1	1	84	85
All	22	101	123


Metrics
Training Accuracy
Validation Accuracy
Training F1
Validation F1
Values
Max_Depth


Metrics
Training Accuracy
Validation Accuracy
Training F1
Validation F1
Values
Min_Samples_leaf