GUVI ZEN FINAL PROJECT – 2

LOAN PREDICTION MODEL PERFORMANCE AND KEY INSIGHTS

NAME: **ASTER NATHAN**

BATCH: **DM3**

CONTENTS

1. **INTRODUCTION:**

   Loan status refers to the current state of a loan, including whether it is being repaid on time or if there are any issues with repayment. The status of a loan can be used to determine the likelihood of repayment, and to identify loans that are at risk of default or delinquency. Loan status is an important consideration for lenders and financial institutions, as it helps them assess the risk of lending money and manage their loan portfolio. By monitoring loan status and identifying loans that are at risk of default, lenders can take proactive steps to mitigate losses and minimize risk. They can also take proactive steps to manage risk, such as offering higher interest rates or requiring additional collateral for high-risk loans. Loan status is an important tool for lenders and financial institutions that want to manage risk, make informed decisions, and ensure regulatory compliance.

   Loan status prediction is the process of using predictive analytics to forecast the likelihood of loan defaults. It is an important tool for lenders and financial institutions, as it helps them identify high-risk loans and take proactive steps to mitigate potential losses. Loan status prediction uses historical data and statistical models to identify patterns and predict future outcomes. It takes into account a range of factors that can impact loan repayment, including credit history, income, employment status, debt-to-income ratio, and other financial variables. There are several methods for conducting loan status prediction, including logistic regression, decision trees, random forests, and neural networks. The chosen method will depend on the nature of the data, the level of accuracy required, and other factors.

2. **OBJECTIVE OF THE PROJECT:**

   The objective of a loan status prediction project is to use predictive analytics to forecast the likelihood of loan defaults or delinquencies. The goal of the project is to build a model that accurately predicts the likelihood of loan default based on historical data and a range of relevant variables, such as credit history, income, employment status, and debt-to-income ratio.

3. **DATA:**

   - SOURCE:

The Dataset was provided by the Guvi mentors

- SIZE:
  Two dataset was sent: train.csv and test.csv
  Train.csv contains 615 Rows and 13 Columns
  Test.csv contains 368 Rows and 12 Columns
- ATTRIBUTES:
  - Loan_ID:
    It gives us an information on the Applicants who have a unique Loan ID.
  - Gender:
    Gender displays the gender of the Loan Applicants
  - Married:
    Married shows whether the Loan Applicant is Married or Unmarried
  - Dependents:
    Dependents gives us an information on the number of people who rely on the borrower for financial support
  - Education:
    It gives us an insight that whether the applicant is a Graduate degree or not
  - Self_Employed:
    Self_Employed refers to individuals who work for themselves and operate their own business and are not dependent on an employer
  - Applicant Income:
    Applicant income in loan status prediction refers to the income earned by the person applying for a loan.
  - Co Applicant Income:
    Co-applicant income refers to the income earned by another person who is applying for the loan with the primary borrower, typically a spouse or family member.
  - Loan Amount:
    Loan amount refers to the total amount of money that a an applicant is requesting to borrow from the bank.
  - Loan_Amount_Term:
    Loan Amount Term refers to the duration of the loan and the applicant has to repay the loan during the time frame applied.

- Credit_History:
  Credit history refers to a record of an individual's past borrowing and repayment behavior.
- Property_Area:
  Property Area refers to the type of area that the Applicant resides
- Loan_Status(Available only on the train data):
  It gives us an information whether the loan was approved for the specific applicant or not

4. FEATURE ENGINEERING:

- Replaced the Null values from the train.csv data and test.csv data and replaced them with mode and median.
  - Train.csv

```
#FILLING THE NULL VALUES WITH 'mode()' and 'median()' in train dataset
train_data['Gender'].fillna(train_data['Gender'].mode()[0], inplace = True)
train_data['Married'].fillna(train_data['Married'].mode()[0], inplace = True)
train_data['Dependents'].fillna(train_data['Dependents'].mode()[0], inplace = True)
train_data['Education'].fillna(train_data['Education'].mode()[0], inplace = True)
train_data['Self_Employed'].fillna(train_data['Self_Employed'].mode()[0], inplace = True)
train_data['CoapplicantIncome'].fillna(train_data['CoapplicantIncome'].mean(), inplace = True)
train_data['ApplicantIncome'].fillna(train_data['ApplicantIncome'].mean(), inplace = True)
#FILLING THE 'Loan_Amount_Term' WITH 'mode()' AND 'LoanAmount' WITH 'median()'
train_data['Loan_Amount_Term'].fillna(train_data['Loan_Amount_Term'].mode()[0], inplace = True)
train_data['LoanAmount'].fillna(train_data['LoanAmount'].median(), inplace = True)
train_data['Loan_Status'].fillna(0, inplace = True)
train_data['Property_Area'].fillna(train_data['Property_Area'].mode()[0], inplace = True)
```

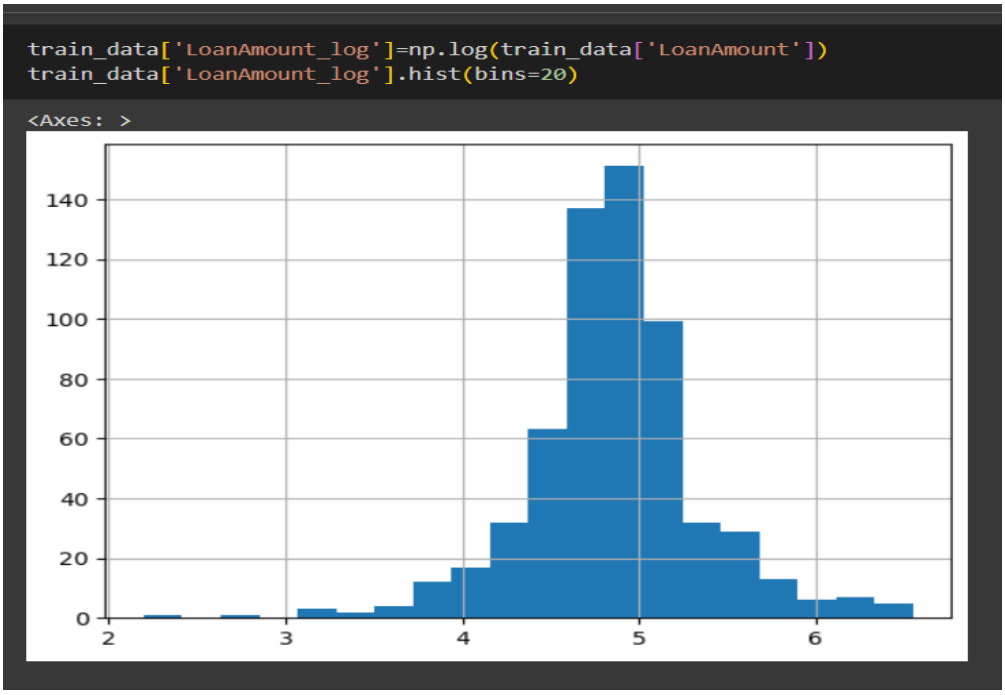
  - Test.csv

```
#FILLING THE NULL VALUES WITH 'mode()' and 'median()'
test_data['Gender'].fillna(test_data['Gender'].mode()[0], inplace = True)
test_data['Dependents'].fillna(test_data['Dependents'].mode()[0], inplace = True)
test_data['Self_Employed'].fillna(test_data['Self_Employed'].mode()[0], inplace = True)
test_data['Credit_History'].fillna(0, inplace = True)
test_data['Loan_Amount_Term'].fillna(test_data['Loan_Amount_Term'].mode()[0], inplace = True)
test_data['LoanAmount'].fillna(test_data['LoanAmount'].median(), inplace = True)
```

- Removed the outliers using np.log function to make sure that the data is properly distributed

```
train_data['LoanAmount_log']=np.log(train_data['LoanAmount'])
train_data['LoanAmount_log'].hist(bins=20)
```

```
<Axes: >
```



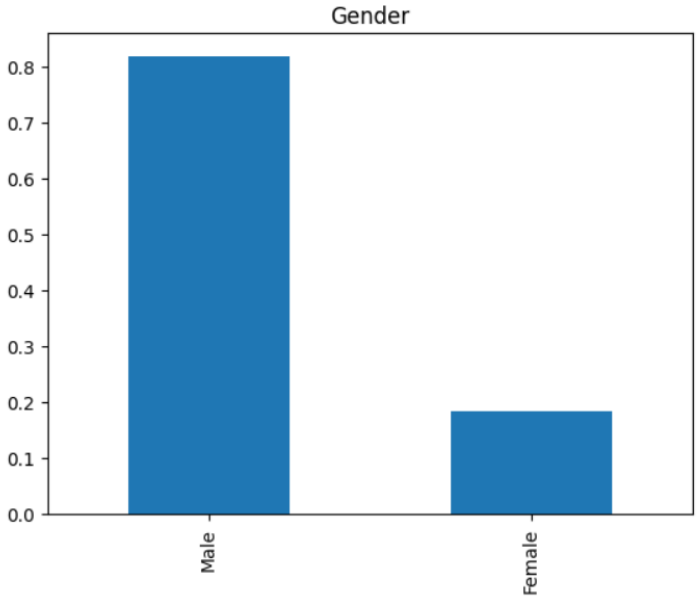
The graph above gives us an insight on the Normally distributed Loan Amount column.

5. **FEATURE SELECTION:**
   - Features Used:
     Loan_Amount_log, Loan_Status, Total_Income_log, LoanID

6. **MODELING:**
   - Logistic Regression
     - from sklearn.linear_model imported LogisticRegression
     - We have used Logistic Regression for the model training of train.csv dataset
     - It gave a score test of 77.4%
   - Random Forest
     - from sklearn.ensemble imported RandomForestClassifier
     - We also used Random Forest for the model training of train.csv dataset
     - It gave a score test of 74.1%
7. **EDA:**

```
train_data['Gender'].value_counts(normalize = True).plot.bar(title = 'Gender')
```

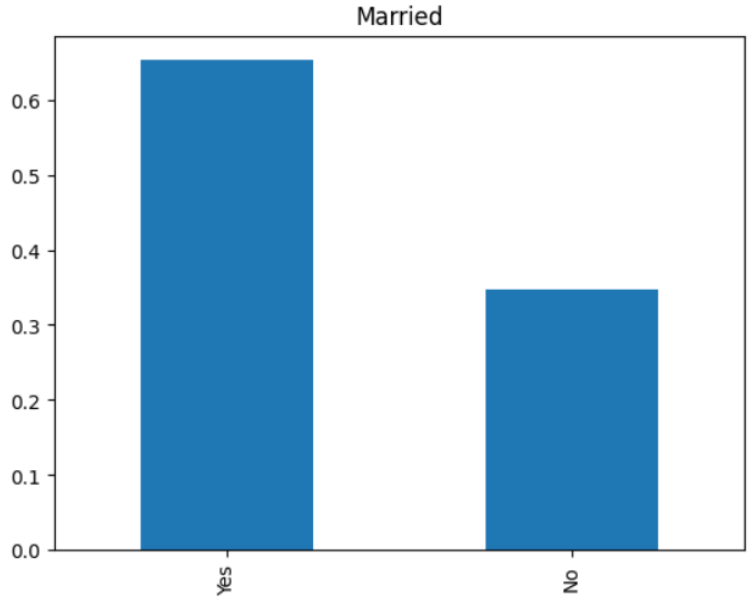
```
<Axes: title={'center': 'Gender'}>
```



The visualization gives us an insight that more than 80% of the Loan Applicants are Male and only 20% are Female

```
[881] train_data['Married'].value_counts(normalize = True).plot.bar(title='Married')
```

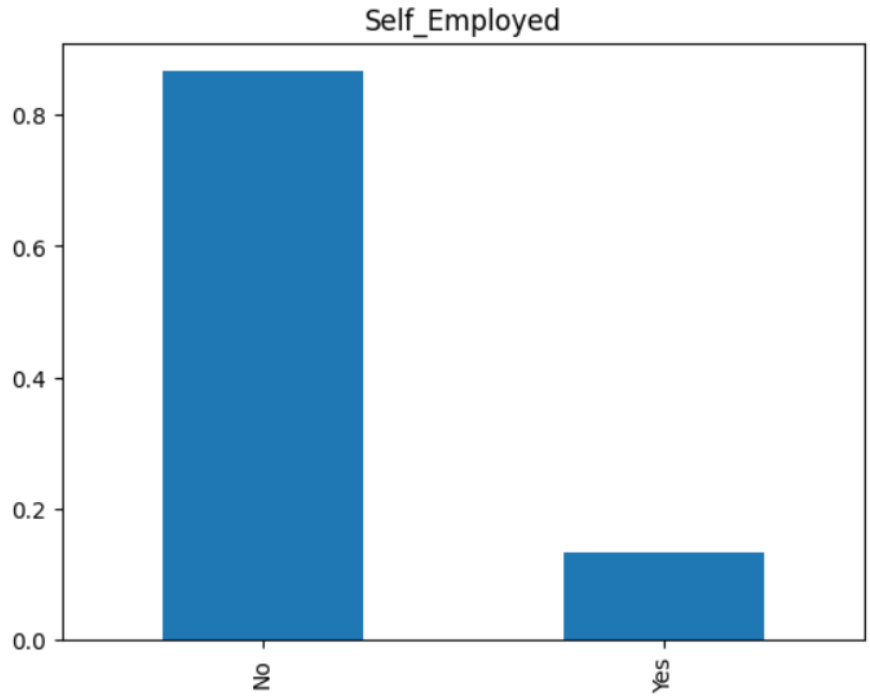
```
<Axes: title={'center': 'Married'}>
```



The visual gives us an insight that around 80% of the Loan Applicants are Married and 35% are Unmarried

```
train_data['Self_Employed'].value_counts(normalize=True).plot.bar(title = 'Self_Employed')
```

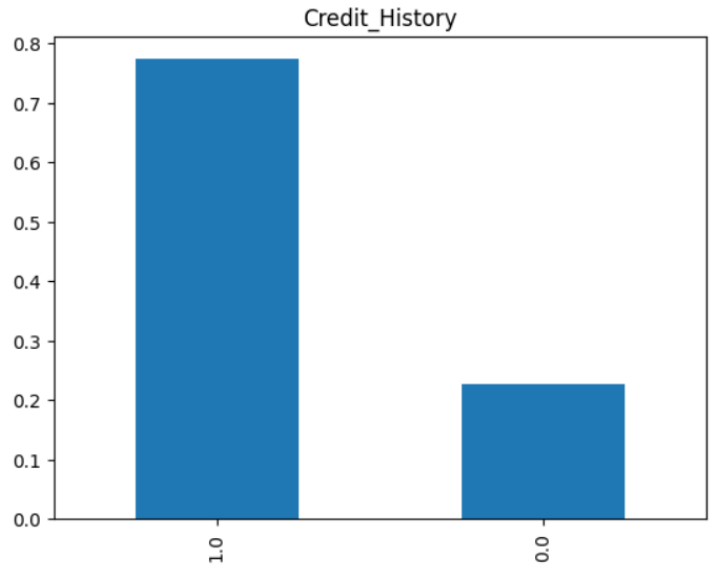
```
<Axes: title={'center': 'Self_Employed'}>
```



The bar graph gives us an insight that only 15% of the Loan Applicants are Self_Employed

```
[888] train_data['Credit_History'].value_counts(normalize = True).plot.bar(title = 'Credit_History')
```

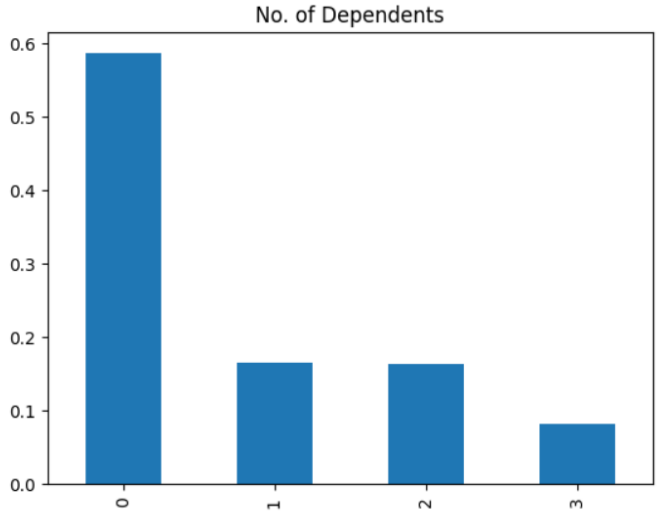
```
<Axes: title={'center': 'Credit_History'}>
```



The visualization shows that more than 80% of the Loan Applicants have paid their previous loans and 20% of the Applicants have either not paid their loans or do not have a Credit History

```
[891] train_data['Dependents'].value_counts(normalize=True).plot.bar(title = 'No. of Dependents')
```

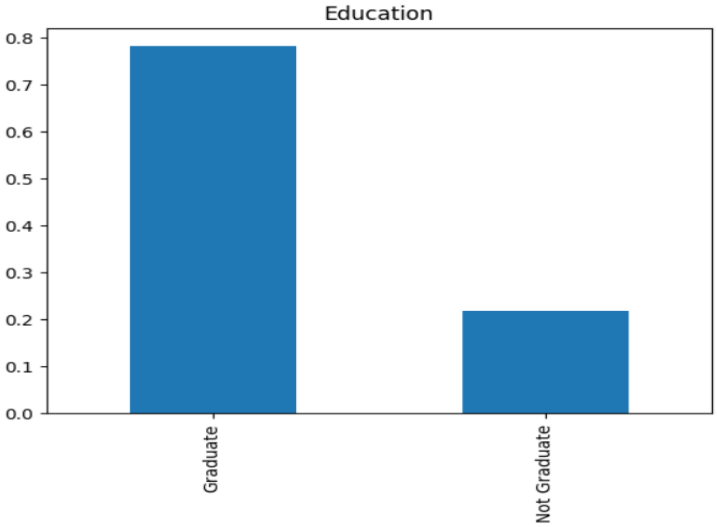
```
<Axes: title={'center': 'No. of Dependents'}>
```



The visual gives us an insight that approximately 60% of the Loan Applicants do not have any dependents, 15% have 1 Dependent, another 15% have 2 Dependents and 10% have 3 or more than 3 Dependents

```
train_data['Education'].value_counts(normalize=True).plot.bar(title = 'Education')
```

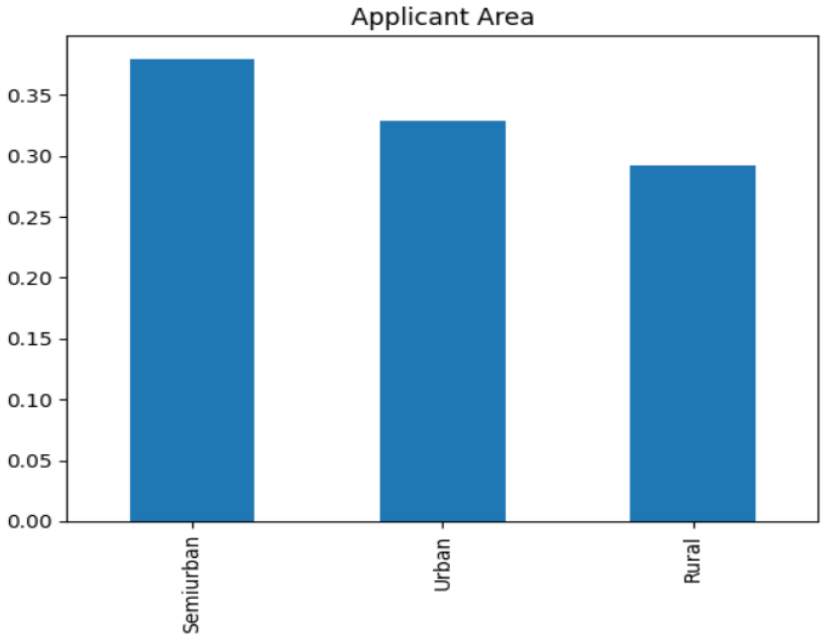
```
<Axes: title={'center': 'Education'}>
```



The visualization gives us an insight that approximately 80% of the Loan Applicants are Graduates and 20% are Non-Graduates

```
[895] train_data['Property_Area'].value_counts(normalize=True).plot.bar(title='Applicant Area')
```

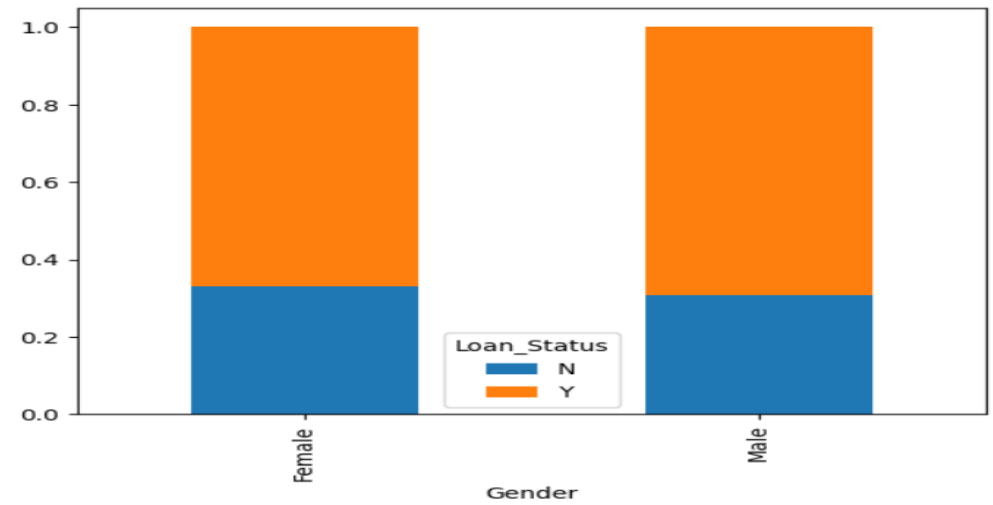
```
<Axes: title={'center': 'Applicant Area'}>
```



The visualization gives us an insight that most of the people applied for a Loan belong to the Semi-Urban area

```
Gender = pd.crosstab(train_data['Gender'],train_data['Loan_Status'])
Gender.div(Gender.sum(1).astype(float),axis=0).plot(
    kind='bar',stacked=True
)
```

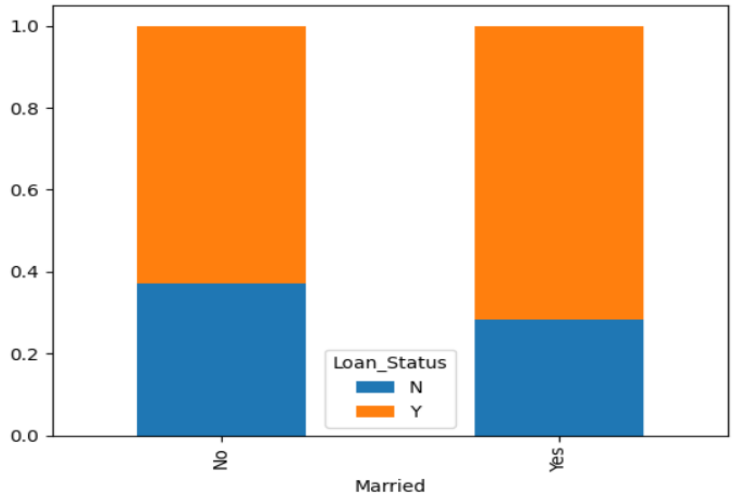
```
<Axes: xlabel='Gender'>
```



The proportions of Male and Female applicants are more or less the same for both Approved and Unapproved Loans

```
[769] Married = pd.crosstab(train_data['Married'],train_data['Loan_Status'])
      Married.div(Married.sum(1).astype(float),axis=0).plot(
          kind='bar',stacked=True
      )
```

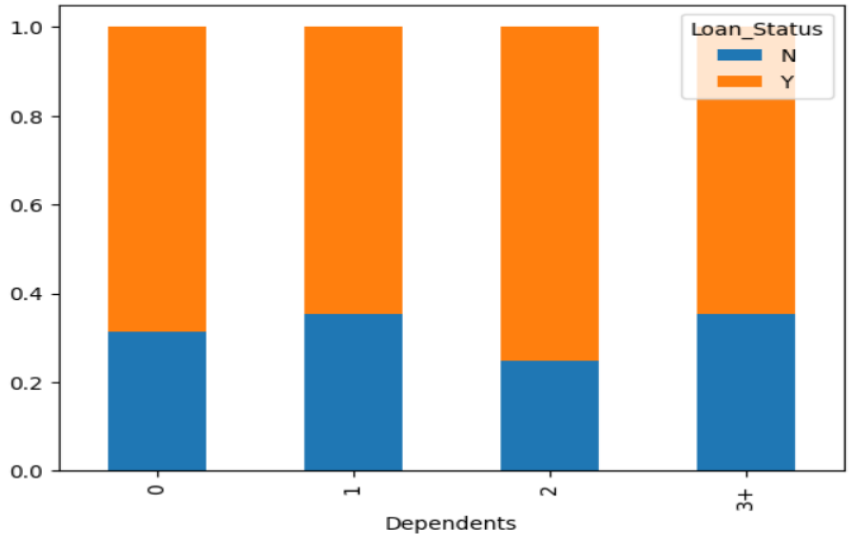
```
<Axes: xlabel='Married'>
```



The visualization gives us an insight that Married people are more capable for a loan approval than Unmarried people

```
[770] Dependents = pd.crosstab(train_data['Dependents'],train_data['Loan_Status'])
      Dependents.div(Dependents.sum(1).astype(float), axis=0).plot(
          kind='bar',stacked=True
      )
```

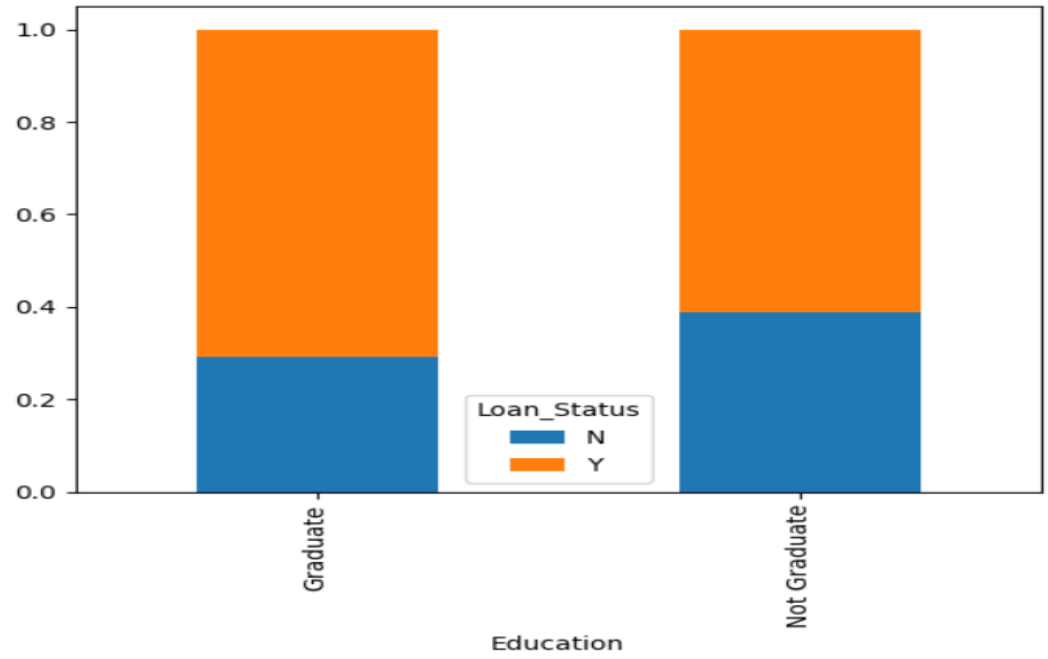
```
<Axes: xlabel='Dependents'>
```



People with 2 Dependents have a higher chance of a Loan Approval as compared to 1 and 3 Dependents

```
Education = pd.crosstab(train_data['Education'],train_data['Loan_Status'])
Education.div(Education.sum(1).astype(float),axis=0).plot(
    kind='bar',stacked=True
)
```

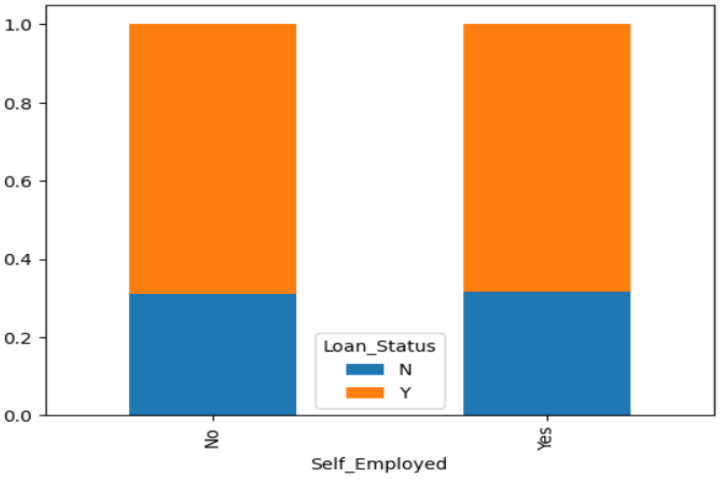
```
<Axes: xlabel='Education'>
```



Graduates have a higher chance of getting their Loan Approved than Non-Graduates

```
Self_Employed = pd.crosstab(train_data['Self_Employed'],train_data['Loan_Status'])
Self_Employed.div(Self_Employed.sum(1).astype(float),axis=0).plot(
    kind='bar',stacked=True
)
```

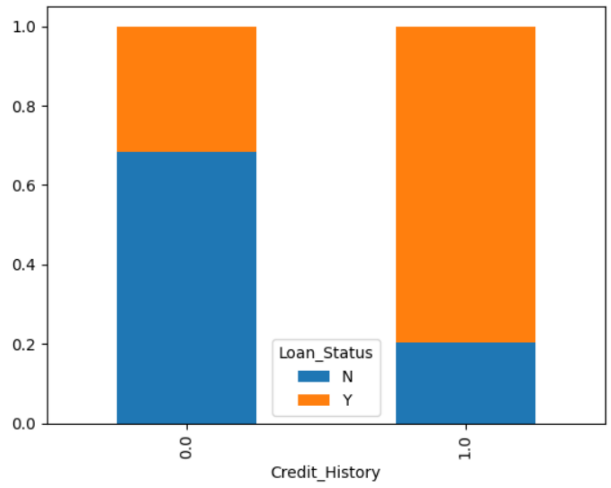
```
<Axes: xlabel='Self_Employed'>
```



There’s no pattern visible for Self_Employed and Loan_Status for Loan Approval

```
Credit_History=pd.crosstab(train_data['Credit_History'],train_data['Loan_Status'])
Credit_History.div(Credit_History.sum(1).astype(float),axis=0).plot(
    kind='bar',stacked=True
)
```

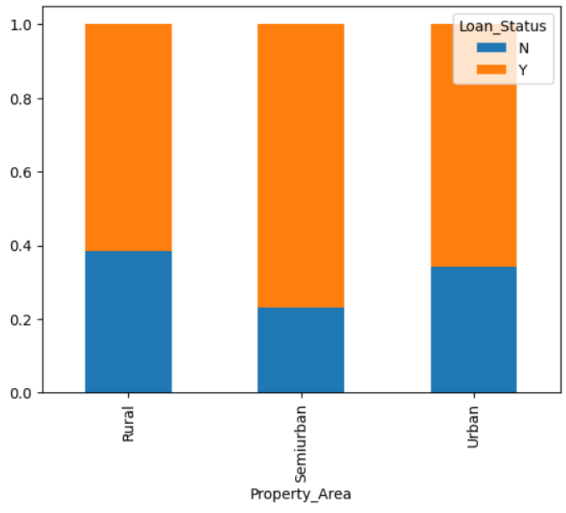
```
<Axes: xlabel='Credit_History'>
```



The Credit History against Loan status shows a much better pattern than any other parameter, inferring that people with a Credit History have a higher chance of getting their Loan Approved than the People with No Credit History

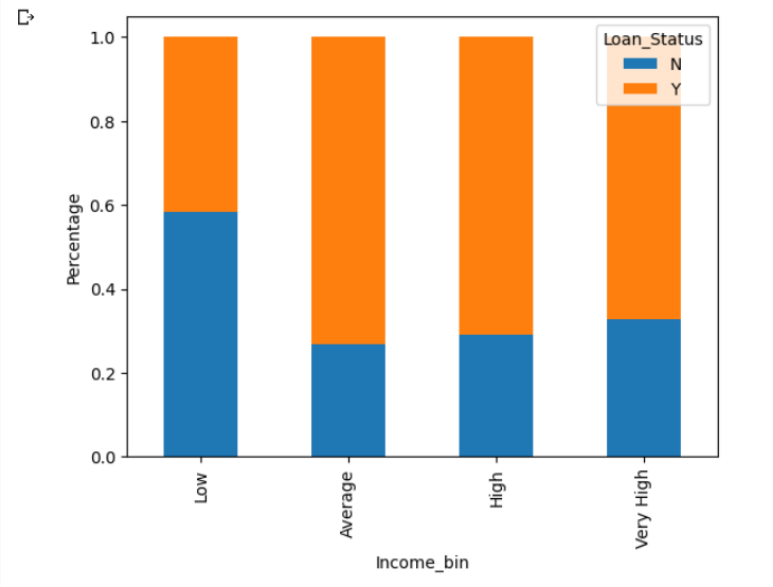
```
[ ] Property_Area = pd.crosstab(train_data['Property_Area'],train_data['Loan_Status'])
    Property_Area.div(Property_Area.sum(1).astype(float),axis=0).plot(
        kind='bar',stacked=True
    )
```

```
<Axes: xlabel='Property_Area'>
```



People residing in the Semi – Urban Area have a higher chance of getting their loan approved than the people living in either Rural or Urban area

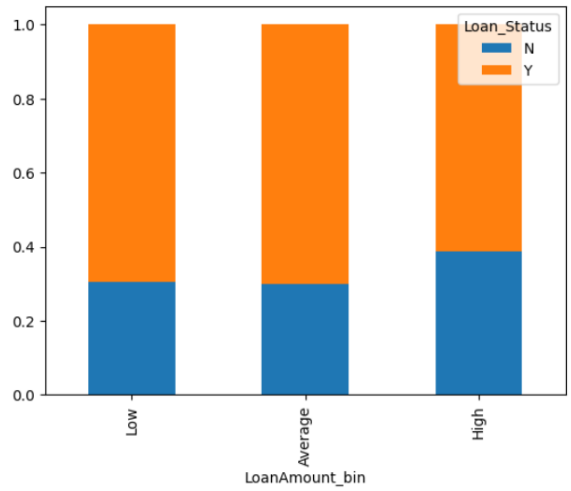
```
#Making bins based on Applicant's Income
bins=[0,2500,4000,6000,81000]
group=['Low','Average','High','Very High']
train_data['Income_bin']=pd.cut(train_data['TotalIncome'],bins,labels=group)
Income_bin=pd.crosstab(train_data['Income_bin'],train_data['Loan_Status'])
Income_bin.div(Income_bin.sum(1).astype(float),axis=0).plot(
    kind='bar',stacked=True
)
plt.xlabel='Applicants_Total_Income'
P=plt.ylabel('Percentage')
```



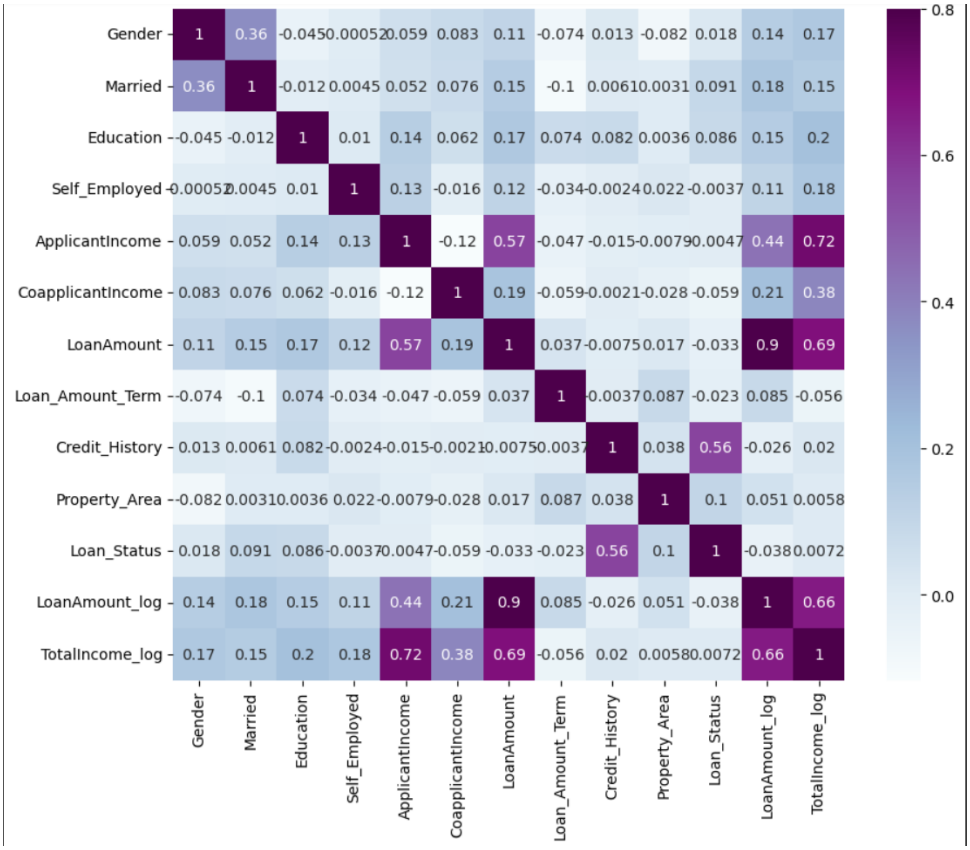
The Applicant's total income does not majorly affect the loan approval as the average and high income applicants tend to get their loan approved as compared to applicants with very high income. However, applicants with low income have a disadvantage as their income affects their Loan Approval. But the pattern contradicts our assumption that applicants with very high income level have a very high loan approval chance.

```
bins=[0,100,200,700]
group = ['Low','Average','High']
train_data['LoanAmount_bin']=pd.cut(train_data['LoanAmount'],bins,labels=group)
LoanAmount_bin = pd.crosstab(train_data['LoanAmount_bin'],train_data['Loan_Status'])
LoanAmount_bin.div(LoanAmount_bin.sum(1).astype(float),axis=0).plot(
    kind='bar',stacked='True'
)
```

```
<Axes: xlabel='LoanAmount_bin'>
```



The pattern above shows that the Loan will be approved if the Loan Amount is Low or Average as compared to the High Income Loan Amount. This proves that the chances of Loan Approval will be high if the Loan Amount is less



The correlation matrix gives us an insight that the most correlated parameter are TotalIncome_log and LoanAmount_log, Credit_History and Loan_Status

8. CONCLUSION:

Using the Random Forest Classifier, we predicted whether the Loan should be approved or not by inputting just the Loan ID