Project Report

Project: Loan Prediction System

Prepared By: Sanif Ali Momin

Data Science Process:

Setting the Research Goal:

The research goal was to make a system that is capable of predicting the loan status i.e. that the person should be getting the loan or not depending on following information about him.

- Gender (Male/Female)
- Married (Y/N)
- Dependents (no. of persons depending on his income)
- Education (Graduate/Under Graduate)
- Self-Employment (Y/N)
- Applicant Income (The one who is applying for loan)
- Co-applicant Income
- Loan Amount (Loan that he had requested)
- Loan Amount Term (Duration in months in which he would pay his loan)
- Credit History (Credit History meet guideline)
- Property Area (Urban/Semi-Urban/Rural)

In the end it would get the result in (Y/N) that person loan is being approved and stored in Loan Status.

The Project would be delieverd in 6 month period.

Retrieving Data:

The data used in this project include the variables mentioned above. The both data training and the testing is being retrieve from Machine Learning Repository and is attach with the project. The training data contain 614 rows of data while testing data contain 367 rows of data.

```
1 train.info()
```

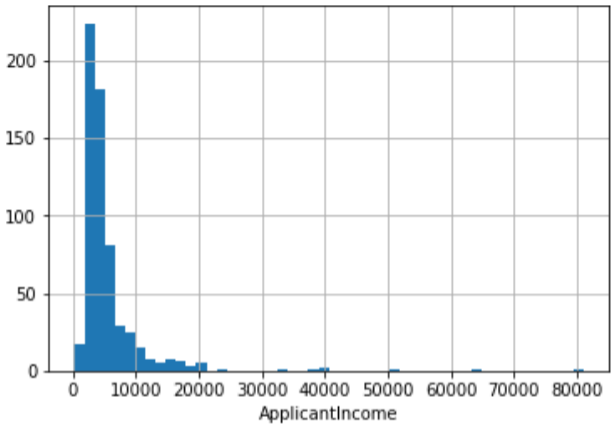
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
Loan_ID              614 non-null object
Gender               601 non-null object
Married              611 non-null object
Dependents           599 non-null object
Education            614 non-null object
Self_Employed        582 non-null object
ApplicantIncome      614 non-null int64
CoapplicantIncome    614 non-null float64
LoanAmount           592 non-null float64
Loan_Amount_Term     600 non-null float64
Credit_History       564 non-null float64
Property_Area        614 non-null object
Loan_Status          614 non-null object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.4+ KB
```

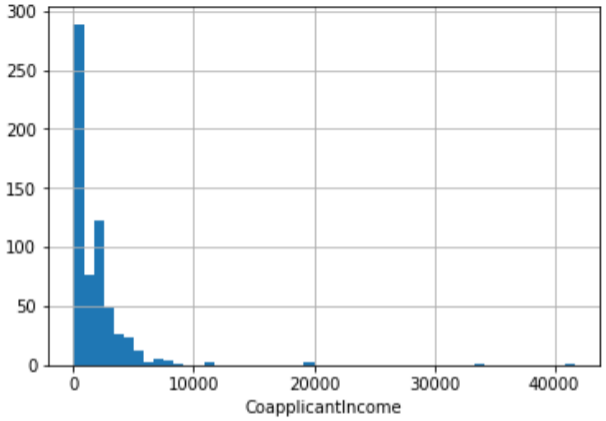
```
1 test.info()
```

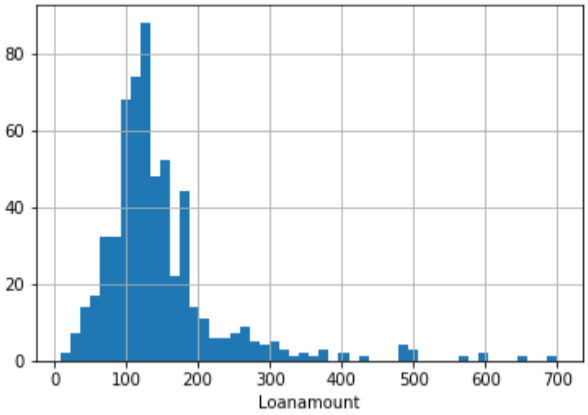
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
Loan_ID              367 non-null object
Gender               356 non-null object
Married              367 non-null object
Dependents           357 non-null object
Education            367 non-null object
Self_Employed        344 non-null object
ApplicantIncome      367 non-null int64
CoapplicantIncome    367 non-null int64
LoanAmount           362 non-null float64
Loan_Amount_Term     361 non-null float64
Credit_History       338 non-null float64
Property_Area        367 non-null object
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB
```

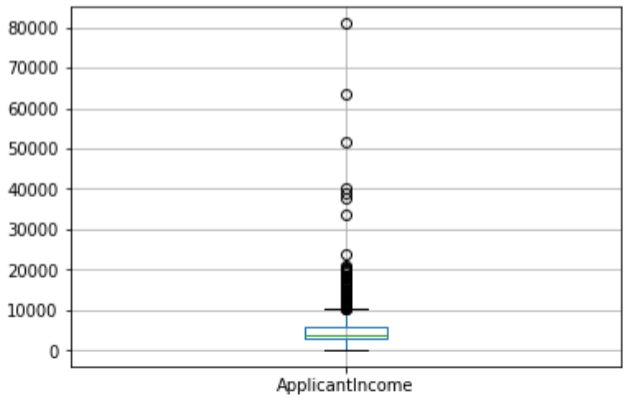
Exploratory Data Analysis:

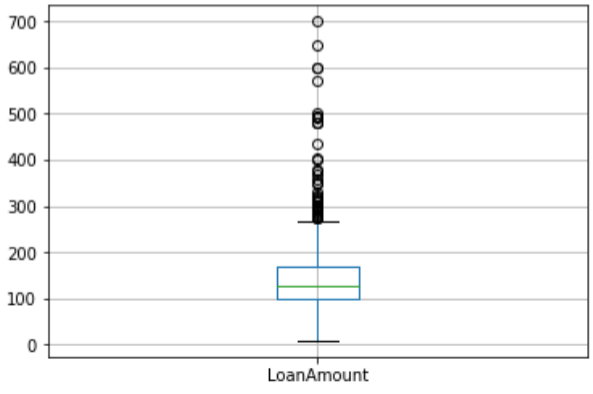
Various Histogram, Bar chart and Box plots are made to perform the Analysis of data. Categorical data is being represented by bar char. While to find the outlier we have made the box plot to identify them. Histogram is used to check the trend in numerical data.



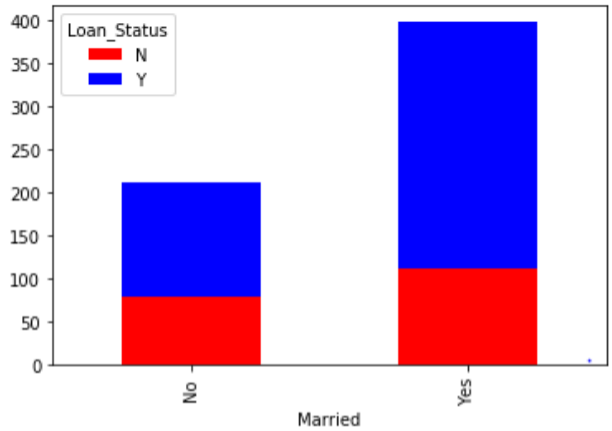


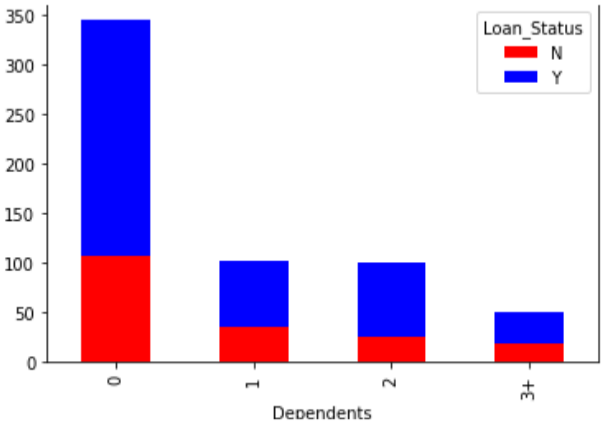


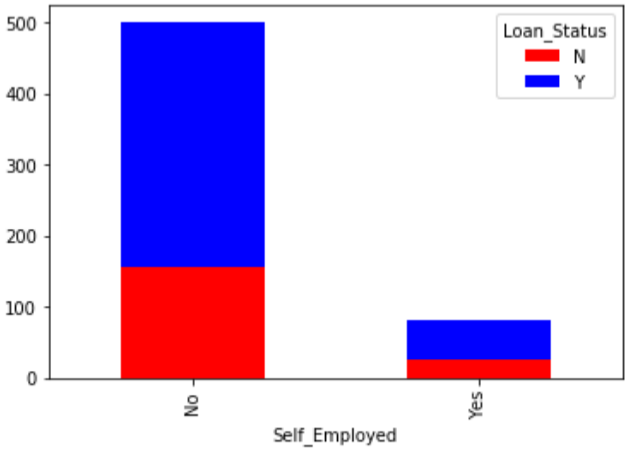


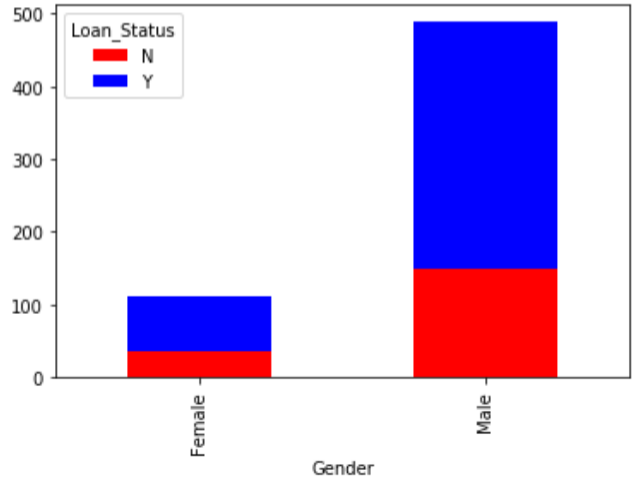



Loan_Status
N
Y
400
300
200
100
0
0.0
1.0
Credit_History


Loan_Status
N
Y
400
350
300
250
200
150
100
50
0
No
Yes
Married


Loan_Status
N
Y
350
300
250
200
150
100
50
0
0
1
2
3+
Dependents


Loan_Status
N
Y
500
400
300
200
100
0
No
Yes
Self_Employed


Loan_Status
N
Y
500
400
300
200
100
0
Female
Male
Gender

Data Preparation:

After EDA we analyze the data present and now we would treat that data in data preparation. In Data Preparation we treat the ram data so that model can be applied and accuracy can be maximized. In Data Preparation we check the missing values present in the data and replace them with mean in numerical data and in categorical data we put the most repeated value.

Missing Values:

Before treatment:

```
#missing values
train.apply(lambda x:sum(x.isnull()),axis=0)
```

```
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
```

Treating Missing Values:

```
train['Gender'].fillna('Male',inplace=True)
train['Married'].fillna('Yes',inplace=True)
train['Dependents'].fillna(0,inplace=True)
train['Self_Employed'].fillna('No',inplace=True)
train['Credit_History'].fillna(1,inplace=True)
train['LoanAmount'].fillna(train['LoanAmount'].mean(), inplace=True)
train['Loan_Amount_Term'].fillna(360,inplace=True)
```

After Treatment:

```
train.apply(lambda x:sum(x.isnull()),axis=0)
```

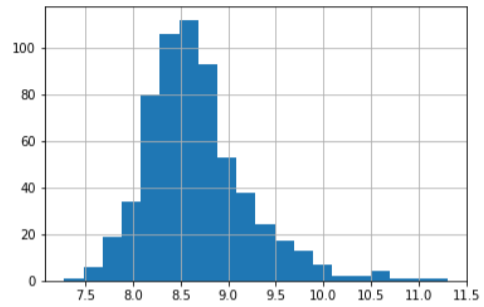
```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

Treating Extreme Values:

Treating Extreme Values by using log as shown in picture below.

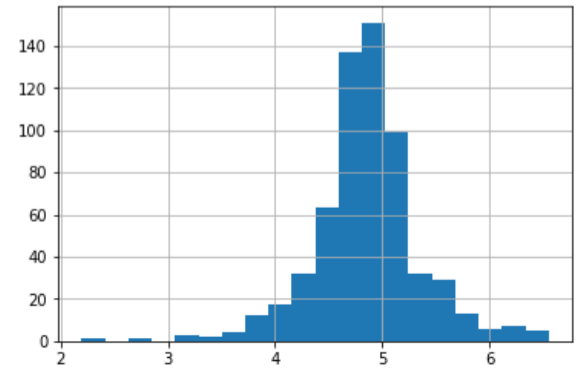
```
train['TotalIncome'] = train['ApplicantIncome'] + train['CoapplicantIncome']
train['TotalIncome'] = np.log(train['TotalIncome'])
train['TotalIncome'].hist(bins=20)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x8dc490c198>
```




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

```
<matplotlib.axes._subplots.AxesSubplot at 0x8dc45c7da0
```



Data type changing:

We change the string data type to integer so that model can be applied. In Dependents column the data is both numeric and string so we change it to numeric.

Data Modelling:

By analyzing we analyze that problem is Classification problem and we apply the following models and then compare accuracy of them.

- Logistic Regression
- KNN with 6 neighbors
- KNN with 10 neighbors
- Decision Tree
- Bagging using KNN
- Bagging using Decision Tree

- Random Forest
- Naïve Baize

```
Logistic Regression:
Accuracy : 82.114%

KNN Model with 10 neighbours:
Accuracy : 82.114%

Bagging using Decision Tree:
Accuracy : 79.675%

Random Forest:
Accuracy : 82.927%
```

```
KNN Model with 6 neighbours:
Accuracy : 80.488%

Decision Tree:
Accuracy : 82.114%

Bagging using KNN Model:
Accuracy : 82.114%

Naive Baise:
Accuracy : 82.114%
```

The best model out of all is random forest which give more accuracy out of all i.e. 82.927%.

The best model is then used in the system to predict the future results and is used to predict the test data.