LOAN ELIGIBILITY PREDICTION

GradStellar

TEAM QUARTZ

NAMES	G_ID
AYESHA SHAHANAZ	2216126
B. VENKATA LAKSHMI VYSHNAVI	2216129
SHIVANI POOSALA	2216141

CONTENTS

1. **ABSTRACT**

2. **INTRODUCTION**

3. **PROPOSED SYSTEM**

4. **MODEL DESIGN(FLOW DIAGRAMS)**

5. **IMPLEMENTATION(STEPS DETAILING)**

6. **DEPLOYMENT**

7. **RESULTS**

8. **CONCLUSION**

9. **FUTURE SCOPE**

NOTE: Each and every section in contents should be done in a different pages, if you are having any various types of data include them as **1.1 ,1.2**
if you have sub variations in data **1.1.1, 1.1.**

ABSTRACT

Banks are making major part of profits through loans. Though lot of people are applying for loans. It's hard to select the genuine applicant, who will repay the loan. While doing the process manually, lot of misconception may happen to select the genuine applicant. Therefore we are developing loan prediction system using machine learning, so the system automatically selects the eligible candidates. This is helpful to both bank staff and applicant. The time period for the sanction of loan will be drastically reduced. In this paper we are predicting the loan data by using some machine learning algorithms that is Decision Tree, Random Forest, K-Nearest Neighbor, XG-Boost, Naive Bayes, Support Vector Machine, Logistic Regression

INTRODUCTION

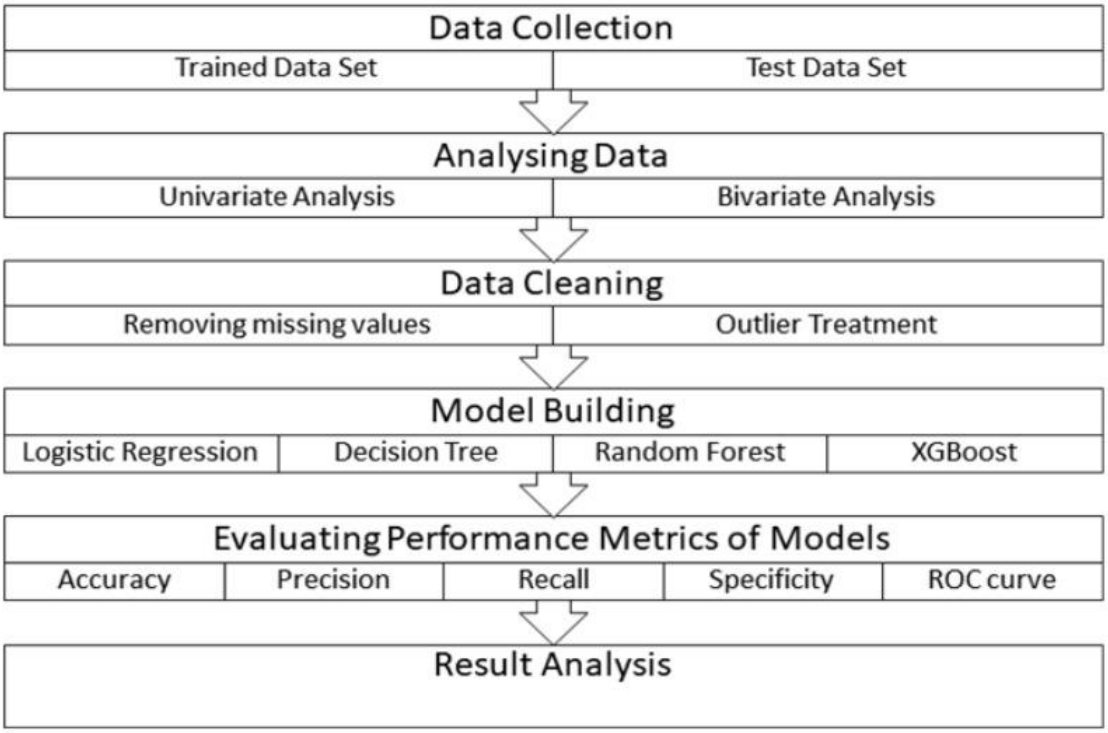
A loan is the core business part of banks. The main portion the bank's profit is directly come from the profit earned from the loans. Though bank approves loan after a regress process of verification and testimonial but still there's no surety whether the chosen hopeful is the right hopeful or not. This process takes fresh time while doing it manually. We can prophesy whether that particular hopeful is safe or not and the whole process of testimonial is automated by machine literacy style. Loan Prognostic is really helpful for retainer of banks as well as for the hopeful also.

PROPOSED SYSTEM

To deal with the problem, we developed automatic loan prediction using machine learning techniques. We will train the machine with previous dataset.so machine can analyse and understand
the process.Then machine will check for eligible applicant and give us result.
Advantages

- Time period for loan sanctioning will be reduced
- Whole process will be automated, so human error. Will be avoided.
- Eligible applicant will be sanctioned loan without any delay.

MODEL DESIGN



Machine Learning and Concepts

Four machine learning models have been used for the prediction of loan approvals. Below are the description of the models used:

Logistic Regression

This is a classification algorithm which uses a logistic function to predict binary outcome (True/False, 0/1, Yes/No) given an independent variable. The aim of this model is to find a relationship between features and probability of particular outcome. The logistic function used is a logit function which is a log of odds in the favor of the

event. Logit function develops a s-shaped curve with the probability estimate similar to a step function.

Decision Tree

This is a supervised machine learning algorithm mostly used for classification problems. All features should be discretized in this model, so that the population can be split into two or more homogeneous sets or subsets. This model uses a different algorithm to split a node into two or more sub-nodes. With the creation of more sub-nodes, homogeneity and purity of the nodes increases with respect to the dependent variable.

Random Forest

This is a tree based ensemble model which helps in improving the accuracy of the model . It combines a large number of Decision trees to build a powerful predicting model. It takes a random sample of rows and features of each individual tree to prepare a decision tree model. Final prediction class is either the mode of all the predictors or the mean of all the predictors.

XG-Boost

This algorithm only works with the quantitative variable. It is a gradient boosting algorithm which forms strong rules for the model by boosting weak learners to a strong learner. It is a fast and efficient algorithm which recently dominated machine learning because of its high performance and speed.

SVM (support vector machine)

support Vector Machine or SVM is a supervised and linear Machine Learning algorithm most commonly used for solving classification problems and is also referred to as Support Vector Classification. There is also a subset of SVM called SVR which stands for Support Vector Regression which uses the same principles to solve regression problems. SVM also supports the kernel method also called the kernel SVM which allows us to tackle non-linearity

KNN (K-nearest neighbor)

This algorithm is used to solve the classification model problems. K-nearest neighbor or K-NN algorithm basically creates an imaginary boundary to classify the data. When new data points come in, the algorithm will try to predict that to the nearest of the boundary line.

Therefore, larger k value means smother curves of separation resulting in less complex models. Whereas, smaller k value tends to overfit the data and resulting in complex models.

IMPLEMENTATION

DATASET

Reading Data

```
df= pd.read_csv('/content/sample_data/LOAN-DATASET.csv')
df
```

	Loan-ID	Gender	Married	Dependents	Education	Self_Employed	ApplicationIncome	CoapplicationIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property Area	Loan Status
0	1001	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	1002	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	1003	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	1004	Male	Yes	0	Not Graduate	No	2583	2358.0	12.0	360.0	1.0	Urban	Y
4	1005	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...	...
)55	7060	Male	Yes	3+	Not Graduate	Yes	4009	1777.0	113.0	360.0	1.0	Urban	Y
)56	7061	Male	Yes	0	Graduate	No	4158	709.0	115.0	360.0	1.0	Urban	Y
)57	7062	Male	No	0	Graduate	No	3250	1993.0	126.0	360.0	1.0	Semiurban	Y
)58	7063	Male	Yes	0	Graduate	No	5000	2393.0	158.0	360.0	1.0	Rural	N
)59	7064	Male	No	0	Graduate	Yes	9200	0.0	98.0	180.0	1.0	Rural	Y

50 rows × 13 columns

Importing the required modules

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

Understanding the Dataset

The machine learning model is trained using the training data set. Every new applicant details filled at the time of application form acts as a test data set. On the basis of the training data sets, the model will predict whether a loan would be approved or not. We have 13 features in total out of which we have 12 independent variables and 1 dependent variable i.e. Loan_Status in train dataset and 12 independent variables in test dataset. The Loan_ID, Gender, Married, Dependents, Education, Self_Employed, Property_Area, Loan_Status are all categorical

Understanding the Data

```
df.columns
```

```
Index(['Loan-ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicationIncome', 'CoapplicationIncome',
       'LoanAmount', 'Loan_Amount_Term', 'Credit_History', 'Property Area',
       'Loan Status'],
      dtype='object')
```

```
df.dtypes
```

```
Loan-ID               int64
Gender               object
Married              object
Dependents           object
Education            object
Self_Employed        object
ApplicationIncome     int64
CoapplicationIncome  float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property Area         object
Loan Status           object
dtype: object
```

Target Variable - Loan Status

We will start first with an independent variable which is our target variable as well. We will analyse this categorical variable using a bar chart as shown below. The bar chart shows that loan of 422 (around 69 %) people out of 614 was approved

```
df['Loan Status'].value_counts()
```

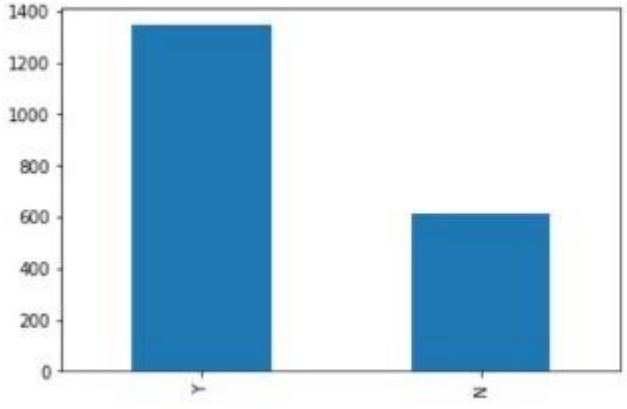
```
Y    1345
N     615
Name: Loan Status, dtype: int64
```

```
df['Loan Status'].value_counts(normalize=True)
```

```
Y    0.686224
N    0.313776
Name: Loan Status, dtype: float64
```

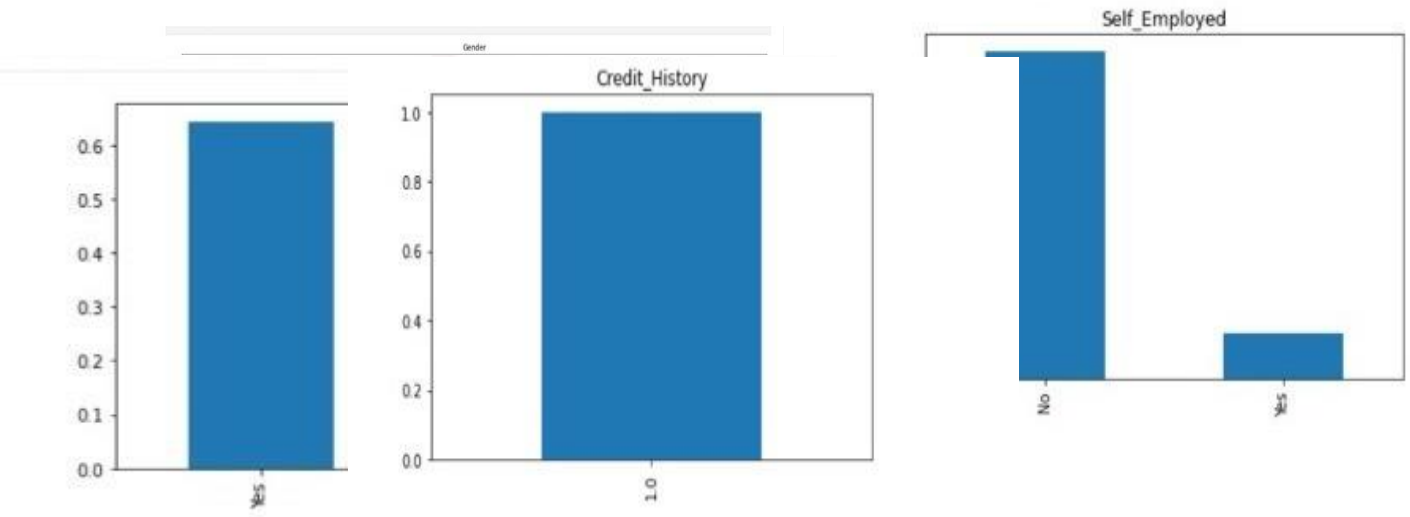
```
df['Loan Status'].value_counts().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd35a11fa10>
```



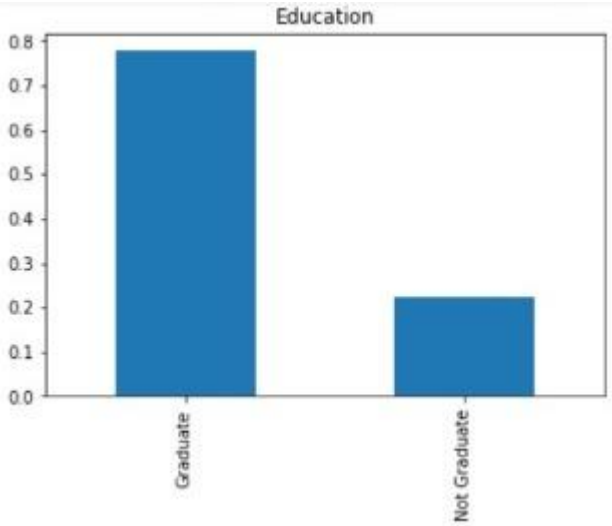
Predictor Variables

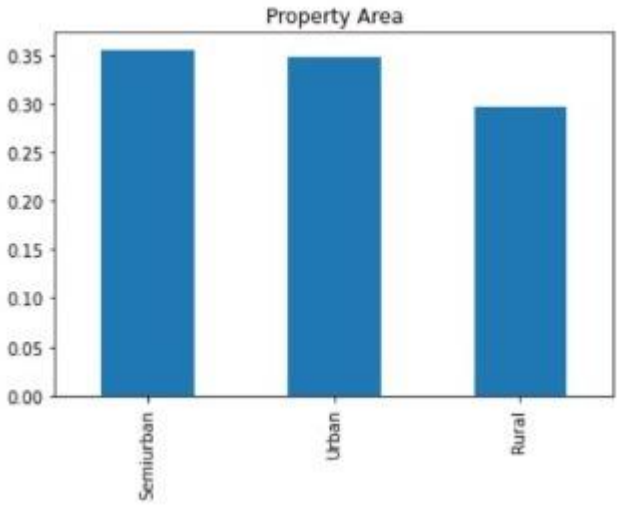
There are 3 types of Independent Variables:
Categorical, Ordinal & Numerical.

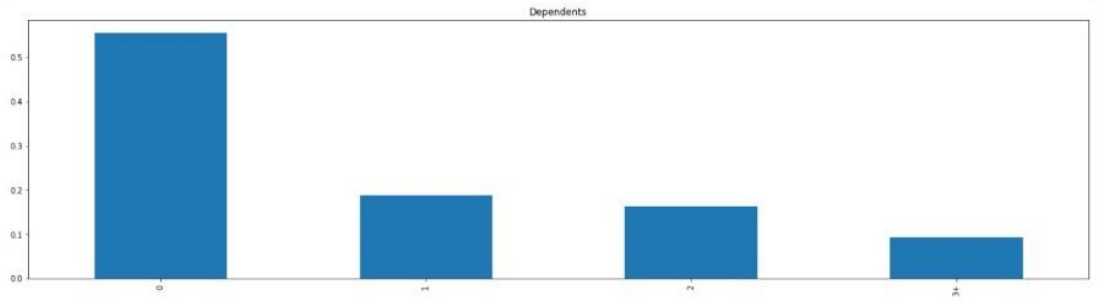


Independent Variable(Ordinal)

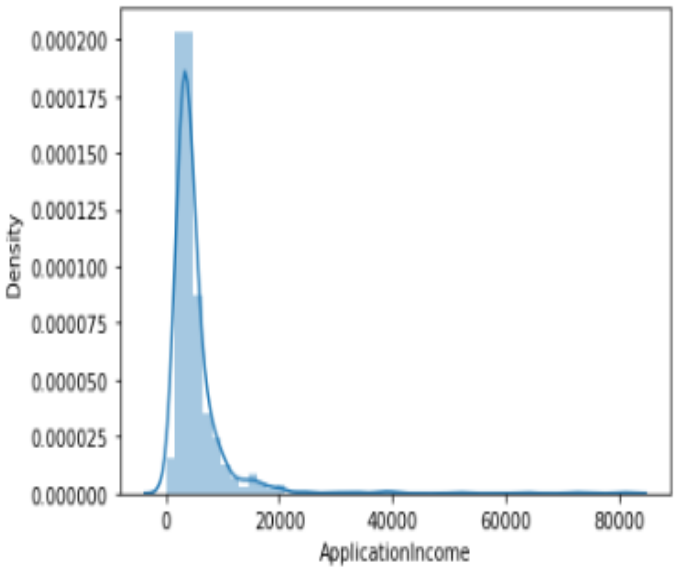
```
df['Dependents'].value_counts(normalize=True).plot.bar(figsize=(24,6),title='Dependents')
plt.show()
df['Education'].value_counts(normalize=True).plot.bar(title='Education')
plt.show()
df['Property Area'].value_counts(normalize=True).plot.bar(title='Property Area')
plt.show()
```

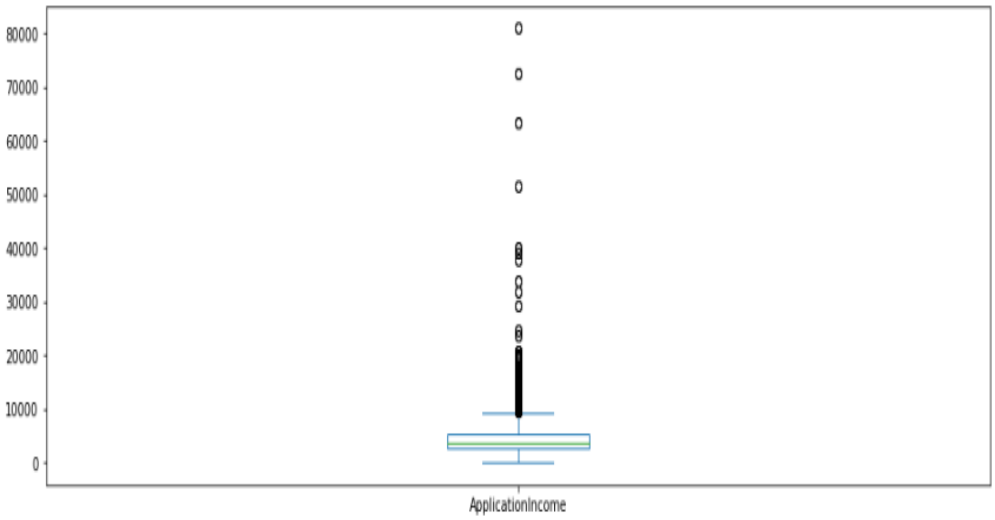






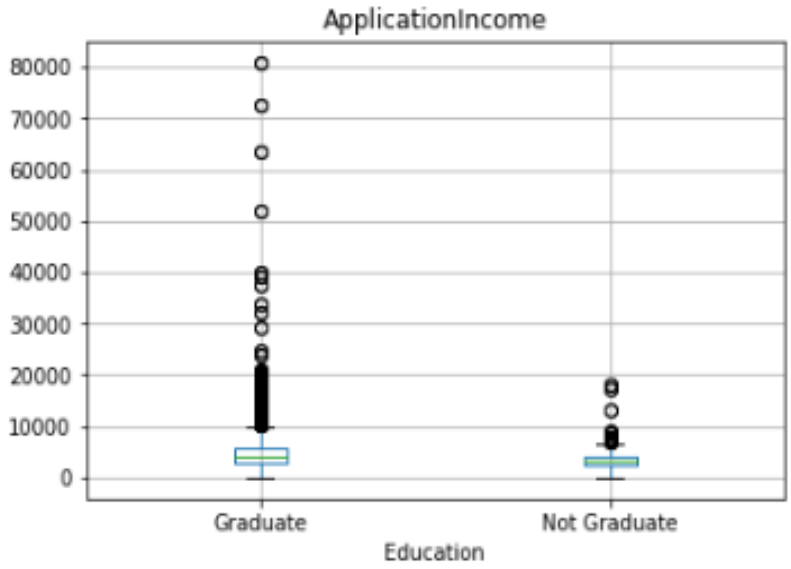
```
sns.distplot(df['ApplicationIncome'])
plt.show()
df['ApplicationIncome'].plot.box(figsize=(16,5))
plt.show()
```



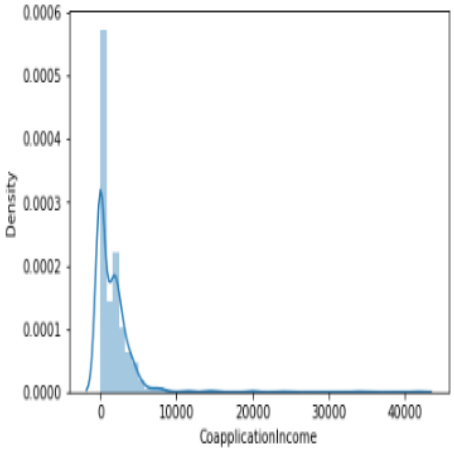


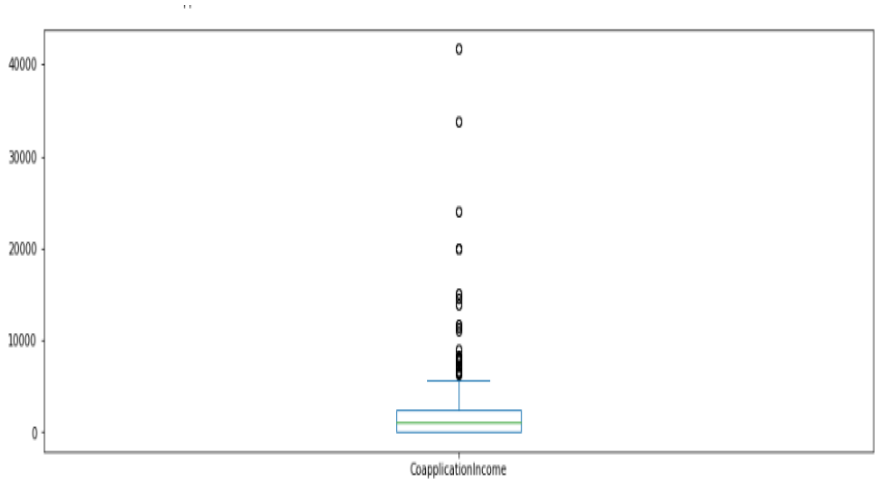
```
df.boxplot(column='ApplicationIncome',by='Education')
plt.suptitle("")
```

```
Text(0.5, 0.98, '')
```



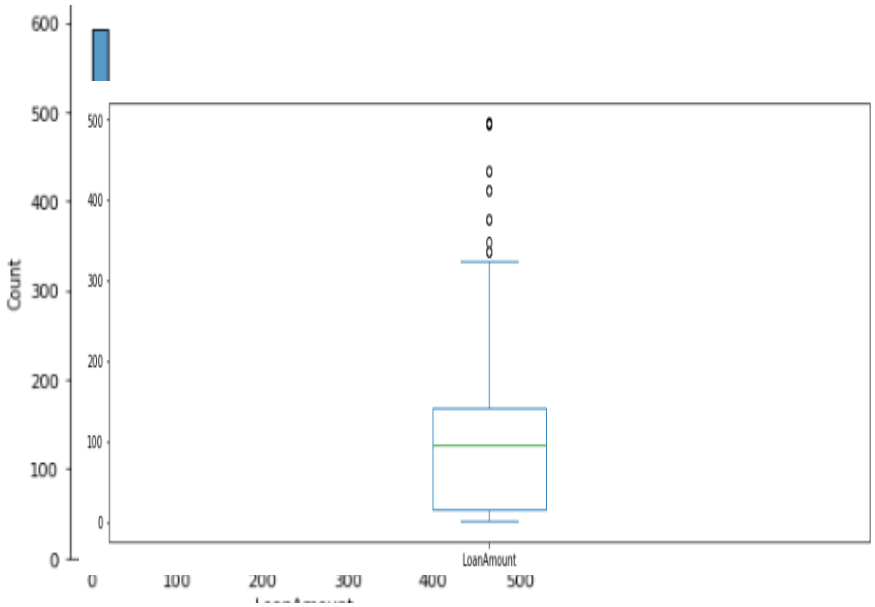
```
sns.distplot(df['CoapplicationIncome'])
plt.show()
df['CoapplicationIncome'].plot.box(figsize=(16,5))
plt.show()
```





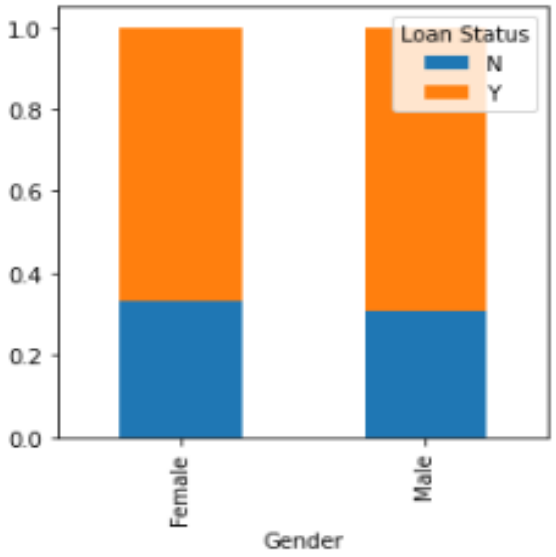

```
df.notna()

sns.displot(df['LoanAmount'])
plt.show()
df['LoanAmount'].plot.box(figsize=(16,5))
plt.show()
```

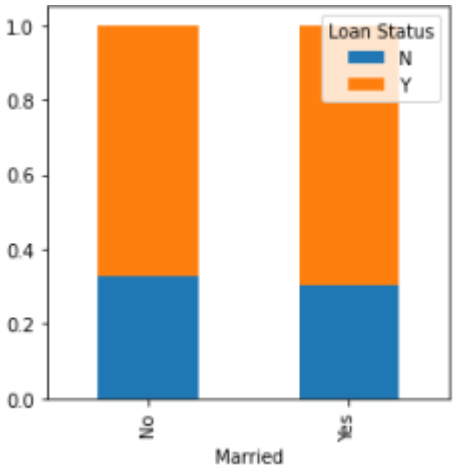

600
500
400
300
200
100
0
Count
500
400
300
200
100
0
LoanAmount
0
100
200
300
400
500

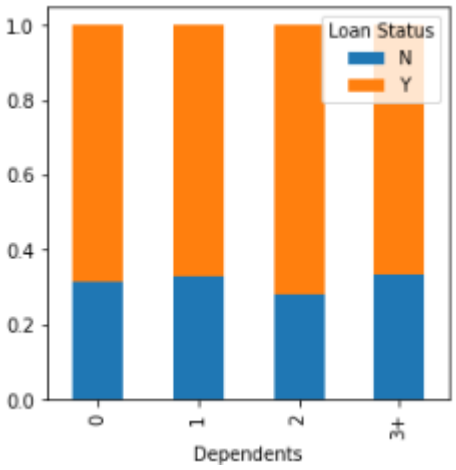
Categorical Independent Variables vs Target Variable

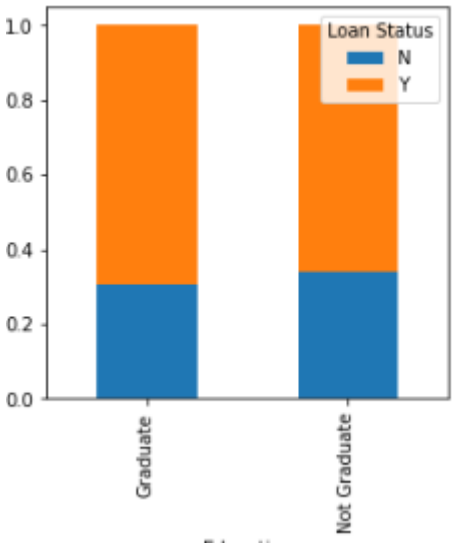
```
Gender=pd.crosstab(df['Gender'],df['Loan Status'])
Gender.div(Gender.sum(1).astype(float),axis=0).plot(kind="bar",stacked=True,figsize=(4,4))
plt.show()
```

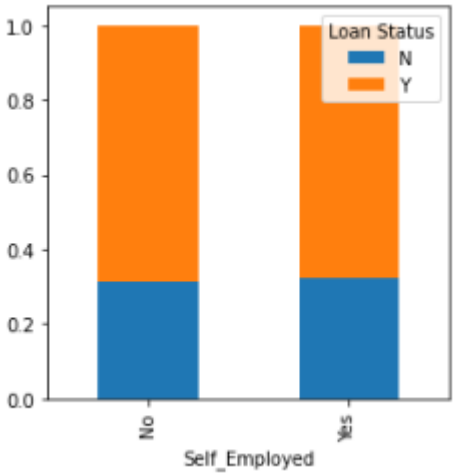


```
Married=pd.crosstab(df['Married'],df['Loan Status'])
Dependents=pd.crosstab(df['Dependents'],df['Loan Status'])
Education=pd.crosstab(df['Education'],df['Loan Status'])
Self_Employed=pd.crosstab(df['Self_Employed'],df['Loan Status'])
Married.div(Married.sum(1).astype(float),axis=0).plot(kind="bar",stacked=True,figsize=(4,4))
plt.show()
Dependents.div(Dependents.sum(1).astype(float),axis=0).plot(kind="bar",stacked=True,figsize=(4,4))
plt.show()
Education.div(Education.sum(1).astype(float),axis=0).plot(kind="bar",stacked=True,figsize=(4,4))
plt.show()
Self_Employed.div(Self_Employed.sum(1).astype(float),axis=0).plot(kind="bar",stacked=True,figsize=(4,4))
plt.show()
```









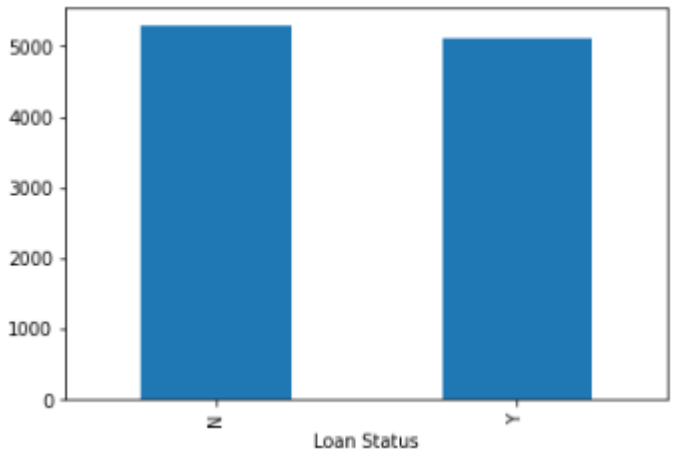
```
Credit_History=pd.crosstab(df['Credit_History'],df['Loan Status'])
Credit_History.div(Credit_History.sum(1).astype(float),axis=0).plot(kind="bar",stacked=True,figsize=(4,4))
plt.show()

'''Property Area=pd.crosstab(df['Property Area'],df['Loan Status'])
Property Area.div(Property Area.sum(1).astype(float),axis=0).plot(kind="bar",stacked=True)
plt.show()'''
```

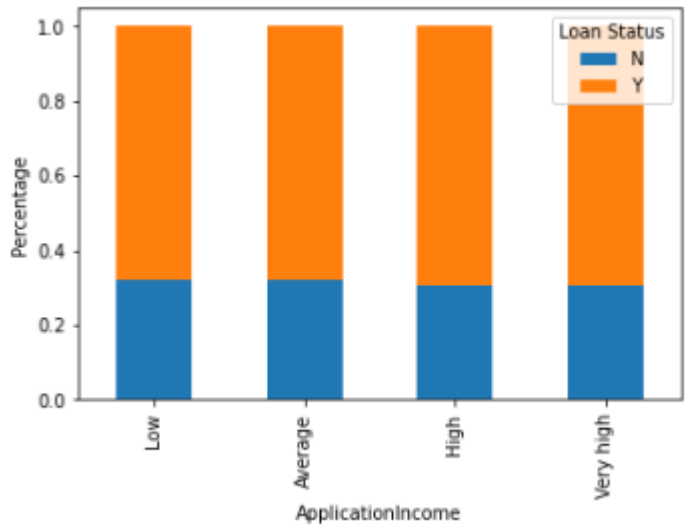
Numerical Independent Variable vs Target Variable

```
df.groupby('Loan Status')['ApplicationIncome'].mean().plot.bar()
```

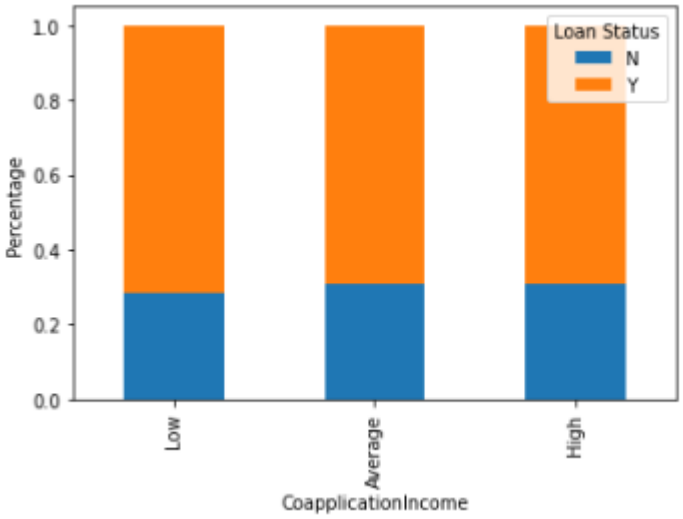
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd35a076710>
```



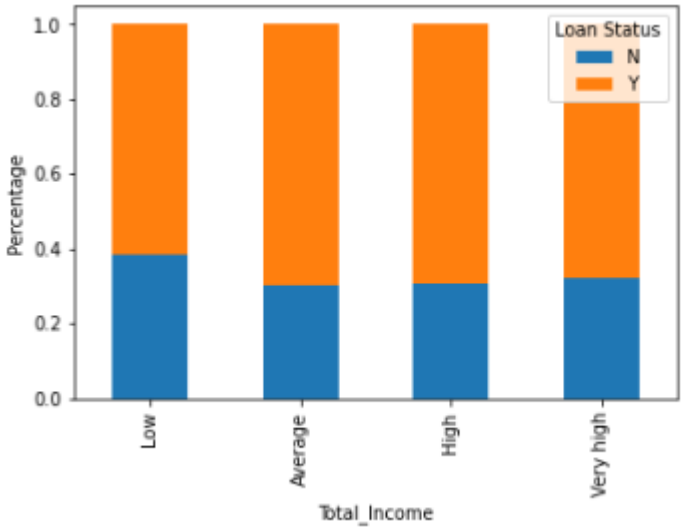

```
bins=[0,2500,4000,6000,81000]
group=['Low','Average','High','Very high']
df['Income_bin']=pd.cut(df['ApplicationIncome'],bins,labels=group)
Income_bin=pd.crosstab(df['Income_bin'],df['Loan Status'])
Income_bin.div(Income_bin.sum(1).astype(float), axis=0).plot(kind="bar",stacked=True)
plt.xlabel('ApplicationIncome')
P=plt.ylabel('Percentage')
```



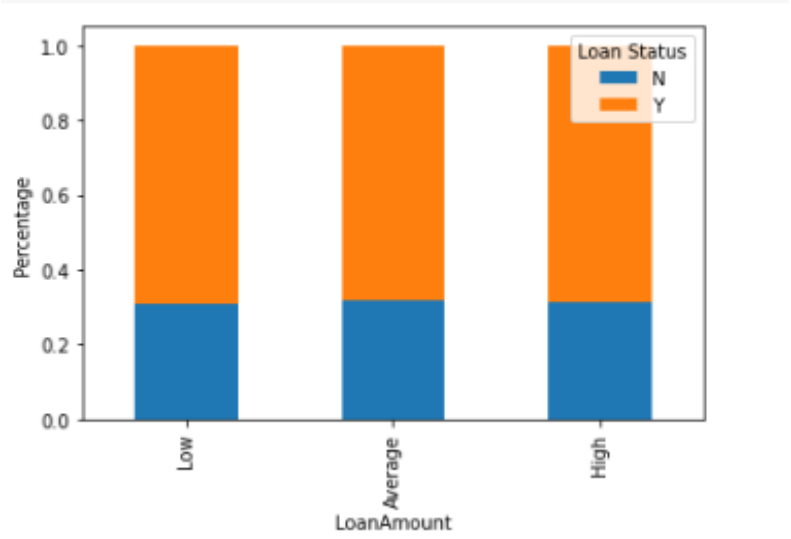
```
df['Total_Income']=df['ApplicationIncome']+df['CoapplicationIncome']
bins=[0,2500,4000,6000,81000]
group=['Low','Average','High','Very high']
df['Total_Income_bin']=pd.cut(df['Total_Income'],bins,labels=group)
Total_Income_bin=pd.crosstab(df['Total_Income_bin'],df['Loan Status'])
Total_Income_bin.div(Total_Income_bin.sum(1).astype(float), axis=0).plot(kind="bar",stacked=True)
plt.xlabel('Total_Income')
P=plt.ylabel('Percentage')
```




```
bins=[0,1000,3000,42000]
group=['Low','Average','High']
df['CoapplicationIncome_bin']=pd.cut(df['CoapplicationIncome'],bins,labels=group)
CoapplicationIncome_bin=pd.crosstab(df['CoapplicationIncome_bin'],df['Loan Status'])
Coappl                                                                    lot(kind="bar",stacked=True)
plt.xl
P=plt.
```



```
bins=[0,100,200,700]
group=['Low','Average','High']
df['LoanAmount_bin']=pd.cut(df['LoanAmount'],bins,labels=group)
LoanAmount_bin=pd.crosstab(df['LoanAmount_bin'],df['Loan Status'])
LoanAmount_bin.div(LoanAmount_bin.sum(1).astype(float), axis=0).plot(kind="bar",stacked=True)
plt.xlabel('LoanAmount')
P=plt.ylabel('Percentage')
```



Missing Value Imputation

```
df.isnull().sum()
```

```
Loan-ID                0
Gender                 0
Married                0
Dependents             0
Education              0
Self_Employed          0
ApplicationIncome      0
CoapplicationIncome    0
LoanAmount            55
Loan_Amount_Term      40
Credit_History       296
Property Area          0
Loan Status            0
dtype: int64
```

```
df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)
df['Married'].fillna(df['Married'].mode()[0], inplace=True)
df['Dependents'].fillna(df['Dependents'].mode()[0], inplace=True)
df['Self_Employed'].fillna(df['Self_Employed'].mode()[0], inplace=True)
df['Credit_History'].fillna(df['Credit_History'].mode()[0], inplace=True)
```

```
df['Loan_Amount_Term'].value_counts()
```

```
360.0    1650
180.0     132
480.0      46
300.0      40
240.0      16
84.0       13
120.0      12
600.0       8
350.0       2
840.0       1
Name: Loan_Amount_Term, dtype: int64
```

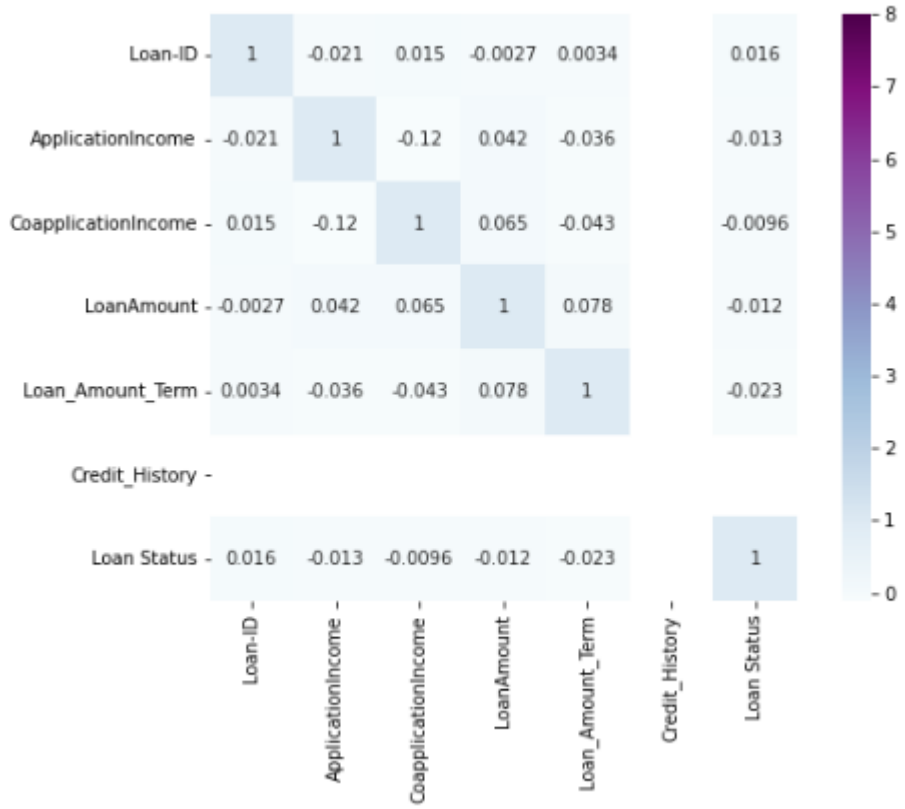
```
df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)
df['Married'].fillna(df['Married'].mode()[0], inplace=True)
df['Dependents'].fillna(df['Dependents'].mode()[0], inplace=True)
df['Self_Employed'].fillna(df['Self_Employed'].mode()[0], inplace=True)
df['Credit_History'].fillna(df['Credit_History'].mode()[0], inplace=True)
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mode()[0], inplace=True)
df['LoanAmount'].fillna(df['LoanAmount'].median(), inplace=True)
```

	Loan-ID	Gender	Married	Dependents	Education	Self_Employed	ApplicationIncome	CoapplicationIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property Area	Loan Status
0	1001	Male	No	0	Graduate	No	5849	0.0	96.0	360.0	1.0	Urban	1
1	1002	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	0
2	1003	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	1
3	1004	Male	Yes	0	Not Graduate	No	2583	2358.0	12.0	360.0	1.0	Urban	1
4	1005	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0	Urban	1

Visualizing correlation via Headmap

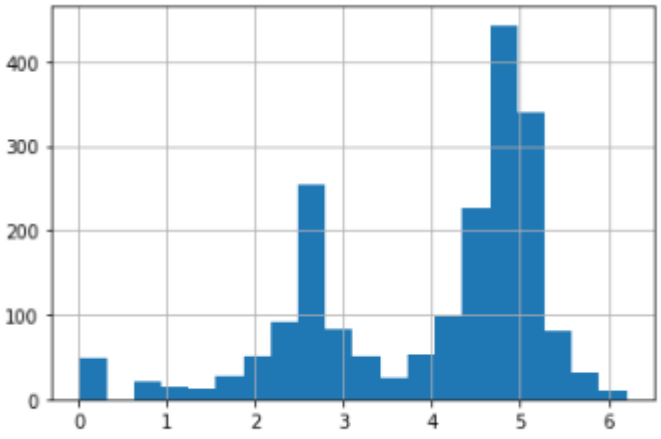
```
matrix=df.corr()
f,ax = plt.subplots(figsize=(9,6))
sns.heatmap(matrix,vmax=8,square=True,cmap="BuPu",annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd359dcf410>
```



Outlier Treatment

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



Dummy variables for categorical variables

```
x=pd.get_dummies(x)
df=pd.get_dummies(df)
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
print(x_train)
print(f"datatype is {type(x_train)}")
```

```
      ApplicationIncome  CoapplicationIncome  LoanAmount  Loan_Amount_Term  \
286                4124                  0.0       115.0             360.0
1062               6000               2250.0       265.0             360.0
281                1811               1666.0        54.0             360.0
100               13650                  0.0        96.0             360.0
773                2964                  0.0        84.0             360.0
...                 ...                  ...         ...               ...
221                7578               1010.0       175.0             360.0
1113               4895                  0.0        12.0             360.0
474                6700               1750.0        23.0             300.0
942                5509                  0.0       143.0             360.0
349                8750               4167.0        38.0             360.0

      Credit_History  LoanAmount_log  Gender_Female  Gender_Male  Married_No  \
286              1.0        4.744932              1            0           1
1062             1.0        5.579730              0            1           0
281              1.0        3.988984              1            0           1
100              1.0        4.564348              0            1           0
773              1.0        4.430817              0            1           1
...              ...             ...            ...          ...         ...
221              1.0        5.164786              0            1           0
1113             1.0        2.484907              0            1           1
474              1.0        3.135494              0            1           0
942              1.0        4.962845              0            1           0
349              1.0        3.637586              0            1           1

      Married_Yes  ...  Dependents_0  Dependents_1  Dependents_2  \
286             0  ...             1             0             0
1062            1  ...             1             0             0
281             0  ...             1             0             0
100             1  ...             0             1             0
773             0  ...             1             0             0
...           ...  ...           ...           ...           ...
221             1  ...             1             0             0
1113            0  ...             1             0             0
474             1  ...             0             0             1
```

```
942             1  ...             1             0             0
349             0  ...             1             0             0

      Education_Graduate  Education_Not Graduate  Self_Employed_No  \
286                    1                       0                 1
1062                   1                       0                 1
281                    1                       0                 1
100                    1                       0                 1
773                    1                       0                 1
...                  ...                     ...               ...
221                    1                       0                 1
1113                   1                       0                 1
474                    1                       0                 1
942                    1                       0                 1
349                    1                       0                 1

      Self_Employed_Yes  Property Area_Rural  Property Area_Semiurban  \
286                   0                    0                        1
1062                  0                    0                        1
281                   0                    0                        0
100                   0                    0                        0
773                   0                    0                        1
...                 ...                  ...                      ...
221                   0                    0                        1
1113                  0                    0                        1
474                   0                    0                        1
942                   0                    1                        0
349                   0                    1                        0

      Property Area_Urban
286                      0
1062                     0
281                      1
100                      1
773                      0
...                    ...
221                      0
1113                     0
474                      0
942                      0
349                      0

[1372 rows x 21 columns]
datatype is <class 'pandas.core.frame.DataFrame'>
```

DEPLOYMENT

Predicting Using 0/1 :

OUTPUT:

Result

```
print('predicted training data',forest.score(x_train,y_train))
print('predicted testing data',forest.score(x_test,y_test))
```

```
predicted training data 0.8220277169948942
predicted testing data 0.5952380952380952
```

```
[ ] accuracy=forest.score(x,y)*100
    print(accuracy)
```

```
75.39561000510464
```

CONCLUSION

We did exploratory data analysis on the features of this data set and saw how each feature od this dataset and saw how each feature is distributed. Analysis each variable to check if data is normally distribued created dummy variables for constructing model finally we got a model coapplicant income and credit history as independent variables with high accuracy.

We tested the data got the accuracy of 75%

Future scope

It is necessary to create and test new strategies that outperform the performance of common data mining models for the domain. As result in the near future, the so called algorithm might be made more reliable, efficient, and robust. This prediction module may be integrated with the automated processing system module in the near future. The system is currently trained on an existing training dataset, but algorithms in the future to allow additional testing data to be included in the training dataset.