Project Report: Loan Prediction Using Machine Learning

```
import pandas as pd
import numpy as np
import matplotlib as plt
%matplotlib inline
```

```
dataset = pd.read_csv("loan-train.csv")
```

```
dataset.head()
```

```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001002   Male      No          0      Graduate            No   
1  LP001003   Male     Yes          1      Graduate            No   
2  LP001005   Male     Yes          0      Graduate           Yes   
3  LP001006   Male     Yes          0  Not Graduate            No   
4  LP001008   Male      No          0      Graduate            No   

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5849                0.0         NaN             360.0   
1             4583             1508.0       128.0             360.0   
2             3000                0.0        66.0             360.0   
3             2583             2358.0       120.0             360.0   
4             6000                0.0       141.0             360.0   

   Credit_History Property_Area Loan_Status  
0             1.0         Urban           Y  
1             1.0         Rural           N  
2             1.0         Urban           Y  
3             1.0         Urban           Y  
4             1.0         Urban           Y  
```

```
dataset.shape
```

```
(614, 13)
```

```
dataset.info()
```

```
<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             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 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
```

```
dataset.describe()
```

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

       Credit_History
count      564.000000
mean         0.842199
std          0.364878
min          0.000000
25%          1.000000
50%          1.000000
75%          1.000000
max          1.000000
```

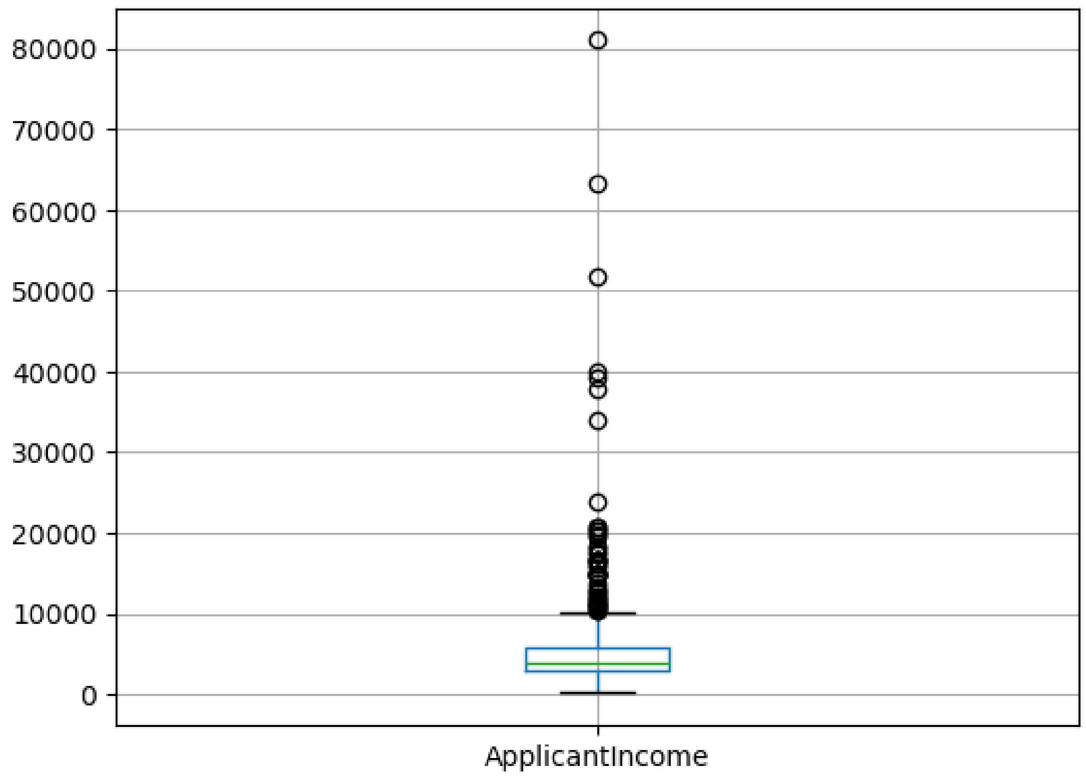
```
pd.crosstab(dataset['Credit_History'],dataset['Loan_Status'],margins=T
rue)
```

```
Loan_Status      N    Y  All
Credit_History
0.0             82    7   89
```

```
1.0                 97  378  475
All                179  385  564
```

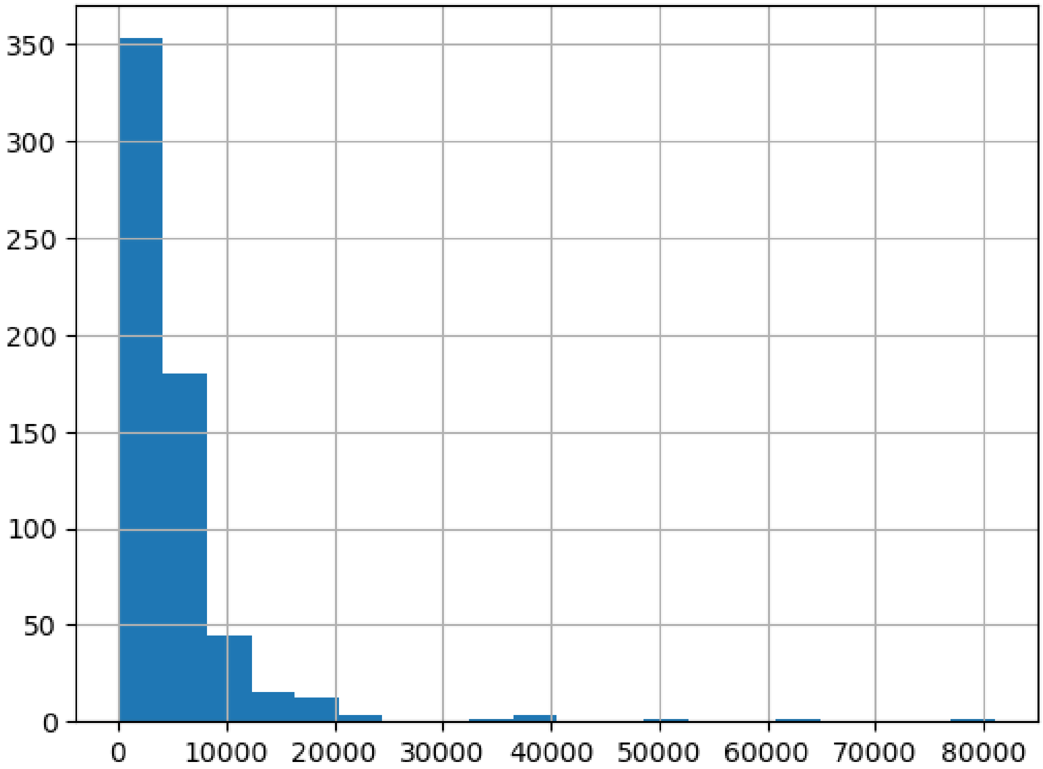
```
dataset.boxplot(column ='ApplicantIncome')
```

```
<Axes: >
```



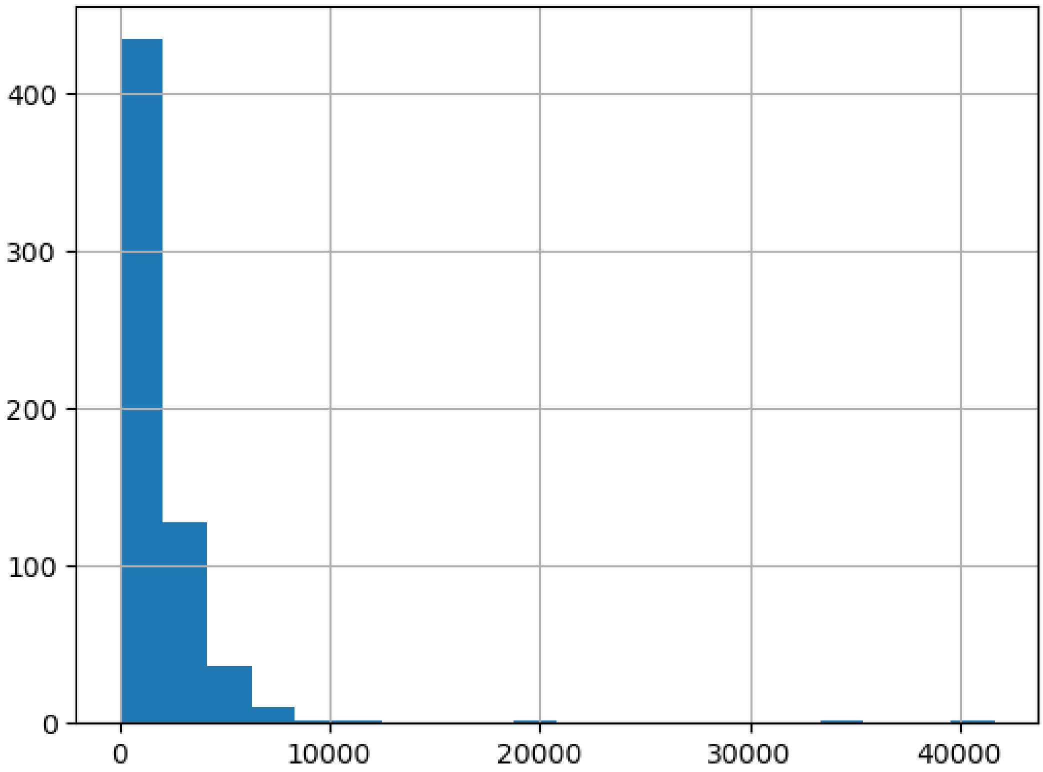
```
dataset['ApplicantIncome'].hist(bins=20)
```

```
<Axes: >
```



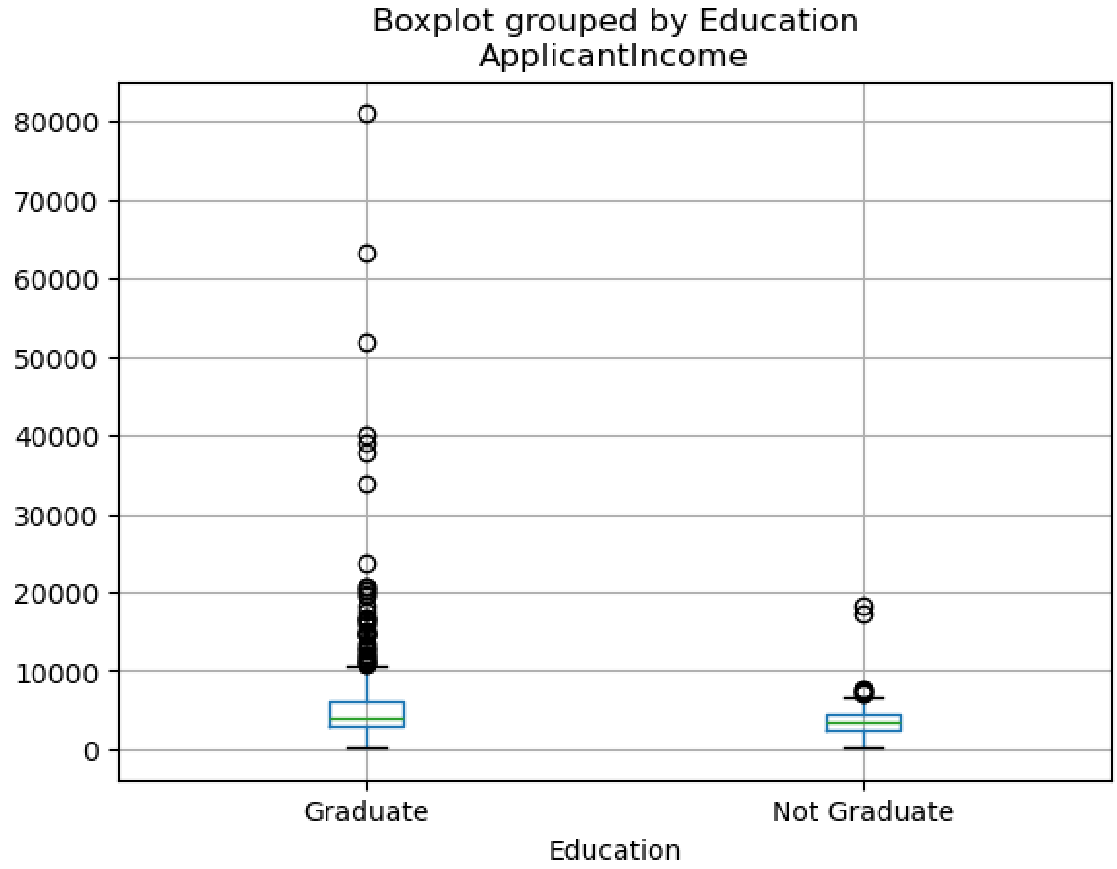
```
dataset['CoapplicantIncome'].hist(bins=20)
```

```
<Axes: >
```



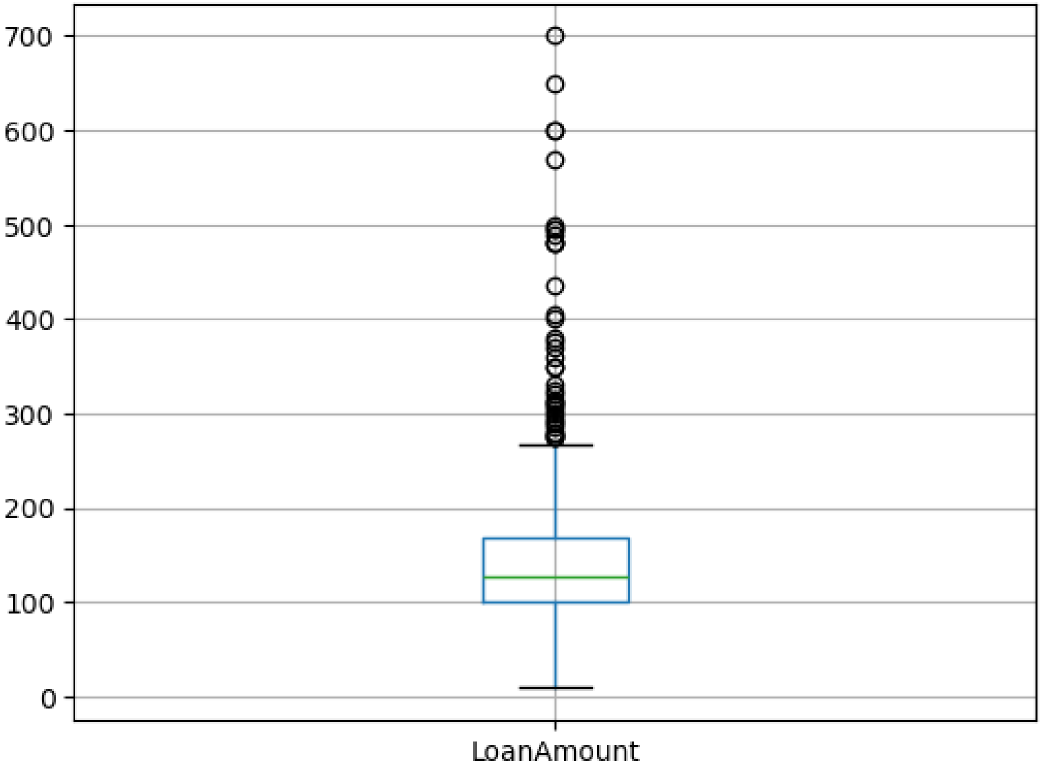
```
dataset.boxplot(column='ApplicantIncome' , by= 'Education')
```

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



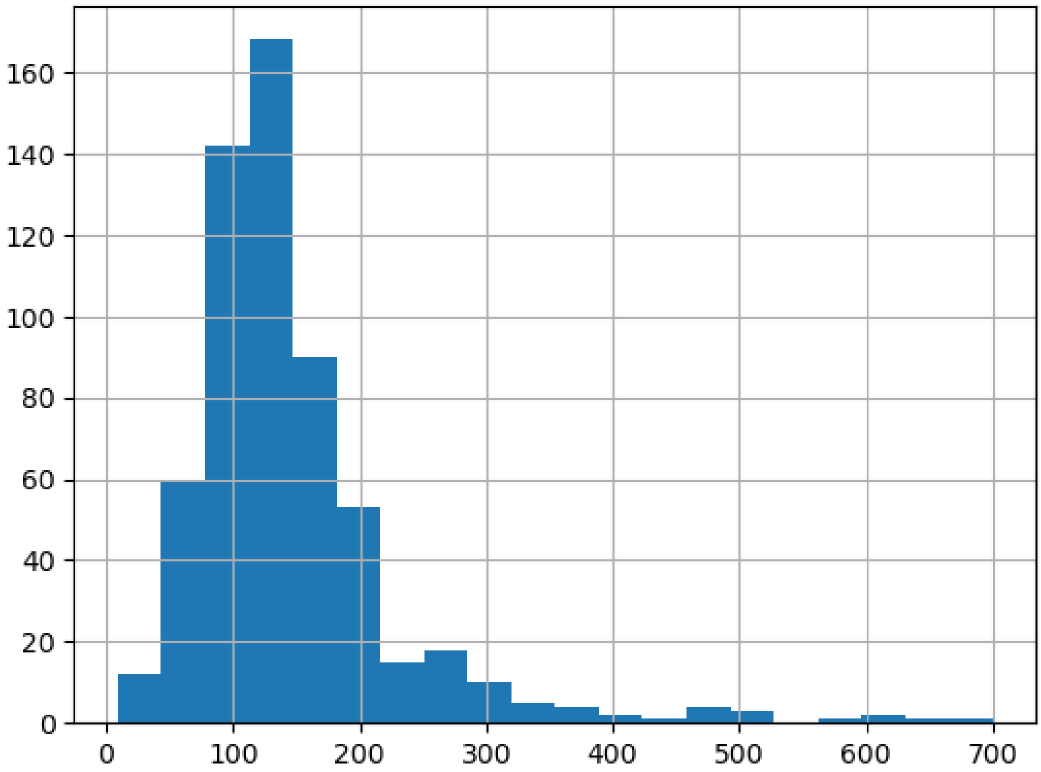
```
dataset.boxplot(column='LoanAmount')
```

```
<Axes: >
```



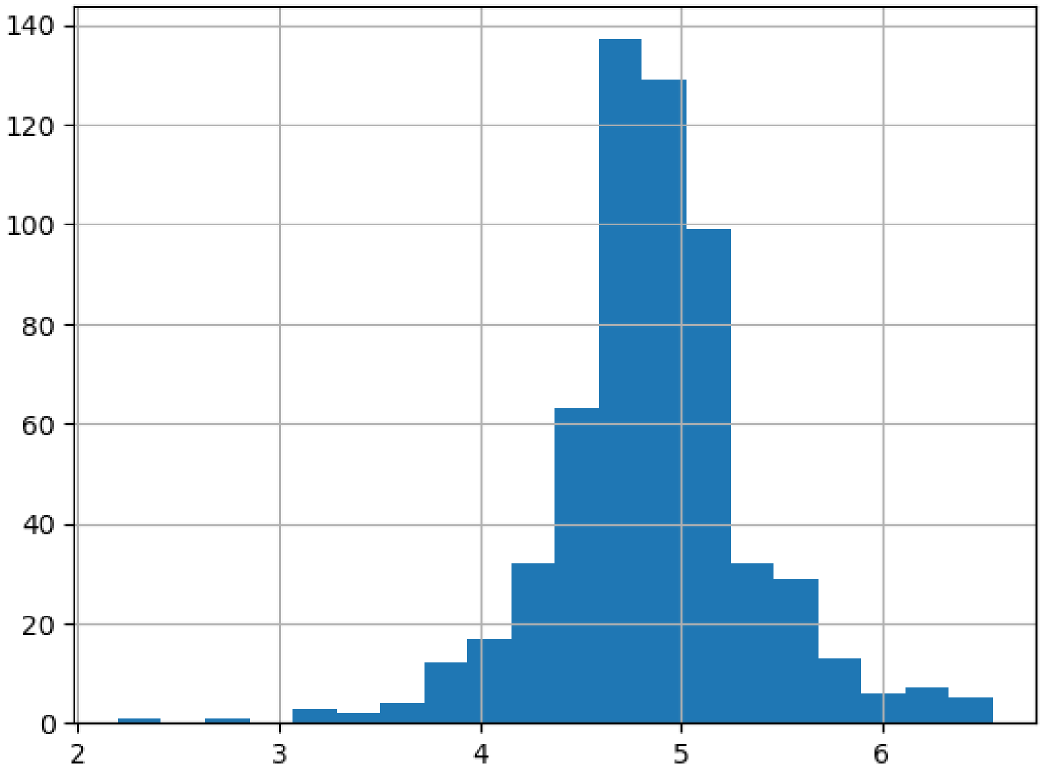
```
dataset['LoanAmount'].hist(bins=20)
```

```
<Axes: >
```



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

```
<Axes: >
```



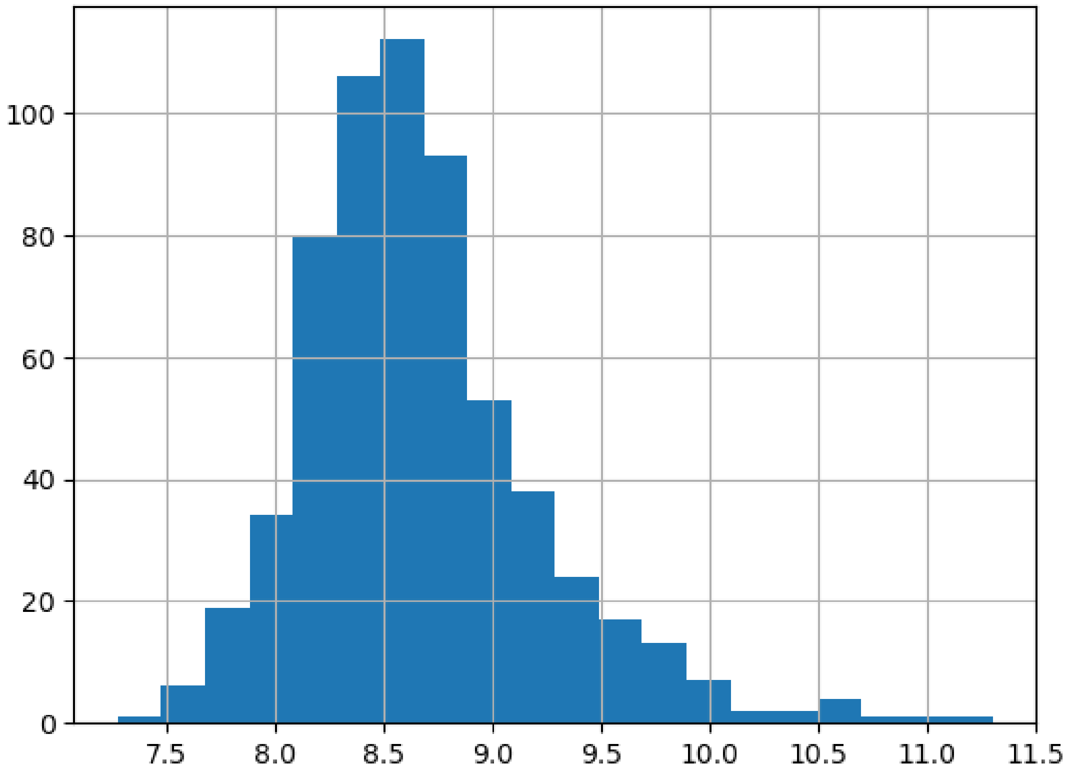
```
dataset.isnull().sum()

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
LoanAmount_log       22
dtype: int64

dataset['Gender'].fillna(dataset['Gender'].mode()[0],inplace=True)

dataset['Married'].fillna(dataset['Married'].mode()[0],inplace=True)

dataset['Dependents'].fillna(dataset['Dependents'].mode()
[0],inplace=True)
```

```
dataset['Self_Employed'].fillna(dataset['Self_Employed'].mode()
[0],inplace=True)

dataset.LoanAmount =
dataset.LoanAmount.fillna(dataset.LoanAmount.mean())
dataset.LoanAmount_log =
dataset.LoanAmount_log.fillna(dataset.LoanAmount_log.mean())

dataset['Loan_Amount_Term'].fillna(dataset['Loan_Amount_Term'].mode()
[0],inplace=True)

dataset['Credit_History'].fillna(dataset['Credit_History'].mode()
[0],inplace=True)

dataset.isnull().sum()
```

```
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
LoanAmount_log       0
dtype: int64
```

```
dataset['TotalIncome'] = dataset['ApplicantIncome'] + dataset
['CoapplicantIncome']
dataset['TotalIncome_log'] = np.log(dataset['TotalIncome'])

dataset['TotalIncome_log'].hist(bins=20)
```

```
<Axes: >
```



```
dataset.head()

    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001002   Male      No          0      Graduate            No
1  LP001003   Male     Yes          1      Graduate            No
2  LP001005   Male     Yes          0      Graduate           Yes
3  LP001006   Male     Yes          0  Not Graduate            No
4  LP001008   Male      No          0      Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5849                0.0  146.412162             360.0
1             4583             1508.0  128.000000             360.0
2             3000                0.0   66.000000             360.0
3             2583             2358.0  120.000000             360.0
4             6000                0.0  141.000000             360.0

   Credit_History Property_Area Loan_Status  LoanAmount_log
TotalIncome  \
0             1.0         Urban           Y        4.857444
5849.0
1             1.0         Rural           N        4.852030
6091.0
2             1.0         Urban           Y        4.189655
3000.0
3             1.0         Urban           Y        4.787492
4941.0
```

```
4              1.0          Urban           Y        4.948760
6000.0

   TotalIncome_log
0         8.674026
1         8.714568
2         8.006368
3         8.505323
4         8.699515

x = dataset.iloc[:,np.r_[1:5,9:11,13:15]].values
y = dataset.iloc[:,12].values

x

array([['Male', 'No', '0', ..., 1.0, 4.857444178729352, 5849.0],
       ['Male', 'Yes', '1', ..., 1.0, 4.852030263919617, 6091.0],
       ['Male', 'Yes', '0', ..., 1.0, 4.189654742026425, 3000.0],
       ...,
       ['Male', 'Yes', '1', ..., 1.0, 5.53338948872752, 8312.0],
       ['Male', 'Yes', '2', ..., 1.0, 5.231108616854587, 7583.0],
       ['Female', 'No', '0', ..., 0.0, 4.890349128221754, 4583.0]],
      dtype=object)

y

array(['Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y',
'Y',
       'N', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'N', 'Y', 'N', 'N', 'N',
'Y',
       'Y', 'Y', 'N', 'Y', 'N', 'N', 'N', 'Y', 'N', 'Y', 'N', 'Y',
'Y',
       'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y',
'Y',
       'N', 'N', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'N',
'N',
       'N', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'N',
'N',
       'N', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
'Y',
       'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
'Y',
       'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
'Y',
       'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N',
'N',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'N', 'N', 'N', 'Y',
'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'N', 'N', 'Y',
'Y',
        'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'N', 'Y',
'N',
        'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'N', 'N',
'N',
        'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y',
'Y',
        'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
'Y',
        'Y', 'N', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y',
'N',
        'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
'Y',
        'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'N',
'Y',
        'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'N', 'Y', 'N', 'N', 'N',
'N',
        'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
'Y',
        'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'N',
'Y',
        'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
'N',
        'N', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'N', 'Y', 'Y',
'Y',
        'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
'Y',
        'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
'Y',
        'N', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
'Y',
        'N', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'N', 'Y', 'Y',
'Y',
        'Y', 'N', 'N', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y',
'Y',
        'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y',
'Y',
        'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'N', 'N',
'Y',
        'Y', 'N', 'Y', 'Y', 'Y', 'N', 'N', 'N', 'Y', 'N', 'Y', 'N',
'Y',
        'N', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'N', 'Y',
'Y',
        'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y',
'Y',
        'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'N', 'N', 'Y', 'N', 'Y',
'Y',
        'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N',
'Y',
        'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N',
'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'N', 'Y', 'Y', 'Y',
'Y',
       'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y',
'Y',
       'N', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'N', 'N',
'N',
       'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
'N',
       'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
'Y',
       'N', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y',
'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'N', 'N', 'Y',
'N',
       'Y', 'N', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N',
'N',
       'N', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N',
'N',
       'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
'Y',
       'Y', 'Y', 'N'], dtype=object)
```

```
from sklearn.model_selection import train_test_split
x_train, x_test , y_train , y_test =
train_test_split(x,y,test_size=0.2, random_state=0)
```

```
print(x_train)
```

```
[['Male' 'Yes' '0' ... 1.0 4.875197323201151 5858.0]
 ['Male' 'No' '1' ... 1.0 5.278114659230517 11250.0]
 ['Male' 'Yes' '0' ... 0.0 5.003946305945459 5681.0]
 ...
 ['Male' 'Yes' '3+' ... 1.0 5.298317366548036 8334.0]
 ['Male' 'Yes' '0' ... 1.0 5.075173815233827 6033.0]
 ['Female' 'Yes' '0' ... 1.0 5.204006687076795 6486.0]]
```

```
from sklearn.preprocessing import LabelEncoder
labelEncoder_x = LabelEncoder()
```

```
for i in range(0,5):
    x_train[:,i]= labelEncoder_x.fit_transform(x_train[:,i])
```

```
x_train[:,7]= labelEncoder_x.fit_transform(x_train[:,7])
```

```
x_train
```

```
array([[1, 7, 0, ..., 1.0, 4.875197323201151, 267],
       [1, 7, 1, ..., 1.0, 5.278114659230517, 407],
       [1, 7, 0, ..., 0.0, 5.003946305945459, 249],
       ...,
       [1, 7, 3, ..., 1.0, 5.298317366548036, 363],
       [1, 7, 0, ..., 1.0, 5.075173815233827, 273],
       [0, 7, 0, ..., 1.0, 5.204006687076795, 301]], dtype=object)
```

```
from sklearn.preprocessing import LabelEncoder
labelEncoder_y = LabelEncoder()
y_train = labelEncoder_y.fit_transform(y_train)
```

```
y_train
```

```
array([1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
1,
       0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1,
1,
       1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1,
0,
       1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1,
1,
       1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0,
0,
       1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1,
1,
       0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1,
0,
       0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1,
1,
       0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1,
1,
       0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1,
1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1,
1,
       1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1,
1,
       1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
1,
       1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0,
0,
       1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1,
1,
       1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,
1,
       1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
0,
       1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
1,
       1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0,
1,
       1, 1, 1, 0, 1, 0, 1])
```

```
for i in range(0,5):
    x_test[:,i]= labelEncoder_x.fit_transform(x_test[:,i])

x_test[:,7]= labelEncoder_x.fit_transform(x_test[:,7])

y_test = labelEncoder_y.fit_transform(y_test)

x_test
```

```
array([[1, 0, 0, 0, 5, 1.0, 4.430816798843313, 85],
       [0, 0, 0, 0, 5, 1.0, 4.718498871295094, 28],
       [1, 1, 0, 0, 5, 1.0, 5.780743515792329, 104],
       [1, 1, 0, 0, 5, 1.0, 4.700480365792417, 80],
       [1, 1, 2, 0, 5, 1.0, 4.574710978503383, 22],
       [1, 1, 0, 1, 3, 0.0, 5.10594547390058, 70],
       [1, 1, 3, 0, 3, 1.0, 5.056245805348308, 77],
       [1, 0, 0, 0, 5, 1.0, 6.003887067106539, 114],
       [1, 0, 0, 0, 5, 0.0, 4.820281565605037, 53],
       [1, 1, 0, 0, 5, 1.0, 4.852030263919617, 55],
       [0, 0, 0, 0, 5, 1.0, 4.430816798843313, 4],
       [1, 1, 1, 0, 5, 1.0, 4.553876891600541, 2],
       [0, 0, 0, 0, 5, 1.0, 5.634789603169249, 96],
       [1, 1, 2, 0, 5, 1.0, 5.4638318050256105, 97],
       [1, 1, 0, 0, 5, 1.0, 4.564348191467836, 117],
       [1, 1, 1, 0, 5, 1.0, 4.204692619390966, 22],
       [1, 0, 1, 1, 5, 1.0, 5.247024072160486, 32],
       [1, 0, 0, 1, 5, 1.0, 4.882801922586371, 25],
       [0, 0, 0, 0, 5, 1.0, 4.532599493153256, 1],
       [1, 1, 0, 1, 5, 0.0, 5.198497031265826, 44],
       [0, 1, 0, 0, 5, 0.0, 4.787491742782046, 71],
       [1, 1, 0, 0, 5, 1.0, 4.962844630259907, 43],
       [1, 1, 2, 0, 5, 1.0, 4.68213122712422, 91],
       [1, 1, 2, 0, 5, 1.0, 5.10594547390058, 111],
       [1, 1, 0, 0, 5, 1.0, 4.060443010546419, 35],
       [1, 1, 1, 0, 5, 1.0, 5.521460917862246, 94],
       [1, 0, 0, 0, 5, 1.0, 5.231108616854587, 98],
       [1, 1, 0, 0, 5, 1.0, 5.231108616854587, 110],
       [1, 1, 3, 0, 5, 0.0, 4.852030263919617, 41],
       [0, 0, 0, 0, 5, 0.0, 4.634728988229636, 50],
       [1, 1, 0, 0, 5, 1.0, 5.429345628954441, 99],
       [1, 0, 0, 1, 5, 1.0, 3.871201010907891, 46],
       [1, 1, 1, 1, 5, 1.0, 4.499809670330265, 52],
       [1, 1, 0, 0, 5, 1.0, 5.19295685089021, 102],
       [1, 1, 0, 0, 5, 1.0, 4.857444178729352, 95],
       [0, 1, 0, 1, 5, 0.0, 5.181783550292085, 57],
       [1, 1, 0, 0, 5, 1.0, 5.147494476813453, 65],
       [1, 0, 0, 1, 5, 1.0, 4.836281906951478, 39],
       [1, 1, 0, 0, 5, 1.0, 4.852030263919617, 75],
       [1, 1, 2, 1, 5, 1.0, 4.68213122712422, 24],
       [0, 0, 0, 0, 5, 1.0, 4.382026634673881, 9],
       [1, 1, 3, 0, 5, 0.0, 4.812184355372417, 68],
       [1, 1, 2, 0, 2, 1.0, 2.833213344056216, 0],
       [1, 1, 1, 1, 5, 1.0, 5.062595033026967, 67],
       [1, 0, 0, 0, 5, 1.0, 4.330733340286331, 21],
       [1, 0, 0, 0, 5, 1.0, 5.231108616854587, 113],
       [1, 1, 1, 0, 5, 1.0, 4.7535901911063645, 18],
       [0, 0, 0, 0, 5, 1.0, 4.74493212836325, 37],
       [1, 1, 1, 0, 5, 1.0, 4.852030263919617, 72],
       [1, 0, 0, 0, 5, 1.0, 4.941642422609304, 78],
       [1, 1, 3, 1, 5, 1.0, 4.30406509320417, 8],
       [1, 1, 0, 0, 5, 1.0, 4.867534450455582, 84],
       [1, 1, 0, 1, 5, 1.0, 4.672828834461906, 31],
       [1, 0, 0, 0, 5, 1.0, 4.857444178729352, 61],
       [1, 1, 0, 0, 5, 1.0, 4.718498871295094, 19],
       [1, 1, 0, 0, 5, 1.0, 5.556828061699537, 107],
       [1, 1, 0, 0, 5, 1.0, 4.553876891600541, 34],
       [1, 0, 0, 1, 5, 1.0, 4.890349128221754, 74],
       [1, 1, 2, 0, 5, 1.0, 5.123963979403259, 62],
       [1, 0, 0, 0, 5, 1.0, 4.787491742782046, 27],
       [0, 0, 0, 0, 5, 0.0, 4.919980925828125, 108],
       [0, 0, 0, 0, 5, 1.0, 5.365976015021851, 103],
       [1, 1, 0, 1, 5, 1.0, 4.74493212836325, 38],
       [0, 0, 0, 0, 5, 0.0, 4.330733340286331, 13],
       [1, 1, 2, 0, 5, 1.0, 4.890349128221754, 69],
       [1, 1, 1, 0, 5, 1.0, 5.752572638825633, 112],
       [1, 1, 0, 0, 5, 1.0, 5.075173815233827, 73],
       [1, 0, 0, 0, 5, 1.0, 4.912654885736052, 47],
       [1, 1, 0, 0, 5, 1.0, 5.204006687076795, 81],
       [1, 0, 0, 1, 5, 1.0, 4.564348191467836, 60],
       [1, 0, 0, 0, 5, 1.0, 4.204692619390966, 83],
       [0, 1, 0, 0, 5, 1.0, 4.867534450455582, 5],
       [1, 1, 2, 1, 5, 1.0, 5.056245805348308, 58],
       [1, 1, 1, 1, 3, 1.0, 4.919980925828125, 79],
       [0, 1, 0, 0, 5, 1.0, 4.969813299576001, 54],
       [1, 1, 0, 1, 4, 1.0, 4.820281565605037, 56],
       [1, 0, 0, 0, 5, 1.0, 4.499809670330265, 120],
       [1, 0, 3, 0, 5, 1.0, 5.768320995793772, 118],
       [1, 1, 2, 0, 5, 1.0, 4.718498871295094, 101],
       [0, 0, 0, 0, 5, 0.0, 4.7535901911063645, 26],
       [0, 0, 0, 0, 6, 1.0, 4.727387818712341, 33],
       [1, 1, 1, 0, 5, 1.0, 6.214608098422191, 119],
       [0, 0, 0, 0, 5, 1.0, 5.267858159063328, 89],
       [1, 1, 2, 0, 5, 1.0, 5.231108616854587, 92],
       [1, 0, 0, 0, 6, 1.0, 4.2626798770413155, 6],
       [1, 1, 0, 0, 0, 1.0, 4.709530201312334, 90],
       [1, 1, 0, 0, 5, 1.0, 4.700480365792417, 45],
       [1, 1, 2, 0, 5, 1.0, 5.298317366548036, 109],
       [1, 0, 1, 0, 3, 1.0, 4.727387818712341, 17],
       [1, 1, 1, 0, 5, 1.0, 4.6443908991413725, 36],
       [0, 1, 0, 1, 5, 1.0, 4.605170185988092, 16],
       [1, 0, 0, 0, 5, 1.0, 4.30406509320417, 7],
       [1, 1, 1, 0, 1, 1.0, 5.147494476813453, 88],
       [1, 1, 3, 0, 4, 0.0, 5.19295685089021, 87],
       [0, 0, 0, 0, 5, 1.0, 4.2626798770413155, 3],
       [1, 0, 0, 1, 3, 0.0, 4.836281906951478, 59],
       [1, 0, 0, 0, 3, 1.0, 5.1647859739235145, 82],
       [1, 0, 0, 0, 5, 1.0, 4.969813299576001, 66],
       [1, 1, 2, 1, 5, 1.0, 4.394449154672439, 51],
       [1, 1, 1, 0, 5, 1.0, 5.231108616854587, 100],
       [1, 1, 0, 0, 5, 1.0, 5.351858133476067, 93],
       [1, 1, 0, 0, 5, 1.0, 4.605170185988092, 15],
       [1, 1, 2, 0, 5, 1.0, 4.787491742782046, 106],
       [1, 0, 0, 0, 3, 1.0, 4.787491742782046, 105],
       [1, 1, 3, 0, 5, 1.0, 4.852030263919617, 64],
       [1, 0, 0, 0, 5, 1.0, 4.8283137373023015, 49],
       [1, 0, 0, 1, 5, 1.0, 4.6443908991413725, 42],
       [0, 0, 0, 0, 5, 1.0, 4.477336814478207, 10],
       [1, 1, 0, 1, 5, 1.0, 4.553876891600541, 20],
       [1, 1, 3, 1, 3, 1.0, 4.394449154672439, 14],
       [1, 0, 0, 0, 5, 1.0, 5.298317366548036, 76],
       [0, 0, 0, 0, 5, 1.0, 4.90527477843843, 11],
       [1, 0, 0, 0, 6, 1.0, 4.727387818712341, 18],
       [1, 1, 2, 0, 5, 1.0, 4.248495242049359, 23],
       [1, 1, 0, 1, 5, 0.0, 5.303304908059076, 63],
       [1, 1, 0, 0, 3, 0.0, 4.499809670330265, 48],
       [0, 0, 0, 0, 5, 1.0, 4.430816798843313, 30],
       [1, 0, 0, 0, 5, 1.0, 4.897839799950911, 29],
       [1, 1, 2, 0, 5, 1.0, 5.170483995038151, 86],
       [1, 1, 3, 0, 5, 1.0, 4.867534450455582, 115],
       [1, 1, 0, 0, 5, 1.0, 6.077642243349034, 116],
       [1, 1, 3, 1, 3, 0.0, 4.248495242049359, 40],
       [1, 1, 1, 0, 5, 1.0, 4.564348191467836, 12]], dtype=object)
```

```
y_test
```

```
array([1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0,
1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1,
1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1,
1,
       1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
0,
       1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1])
```

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
x_train=ss.fit_transform(x_train)
x_test=ss.fit_transform(x_test)
```

```
from sklearn.tree import DecisionTreeClassifier
DTClassifier =
DecisionTreeClassifier(criterion='entropy',random_state=0)
DTClassifier.fit(x_train,y_train)
```

```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

```
y_pred = DTClassifier.predict(x_test)
y_pred
```

```
array([1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0,
1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0,
1,
       0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1,
1,
       1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1])
```

```
from sklearn import metrics
print('The Accuracy of decision Tree is:' ,
metrics.accuracy_score(y_pred,y_test))
```

```
The Accuracy of decision Tree is: 0.7235772357723578
```

```
from sklearn.naive_bayes import GaussianNB
NBclassifier = GaussianNB()
NBclassifier.fit(x_train,y_train)
```

```
GaussianNB()
```

```
y_pred =  NBclassifier.predict(x_test)
```

```
y_pred
```

```
array([1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1])
```

```
print('The Accuracy of Naive Bayes is:' ,
metrics.accuracy_score(y_pred,y_test))
```

```
The Accuracy of Naive Bayes is: 0.8292682926829268
```

```
testdata = pd.read_csv('loan-test.csv')
```

```
testdata.head()
```

```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001015   Male     Yes          0      Graduate            No
1  LP001022   Male     Yes          1      Graduate            No
2  LP001031   Male     Yes          2      Graduate            No
3  LP001035   Male     Yes          2      Graduate            No
4  LP001051   Male      No          0  Not Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5720                  0       110.0             360.0
1             3076               1500       126.0             360.0
2             5000               1800       208.0             360.0
3             2340               2546       100.0             360.0
4             3276                  0        78.0             360.0

   Credit_History Property_Area
0             1.0         Urban
1             1.0         Urban
2             1.0         Urban
3             NaN         Urban
4             1.0         Urban
```

```
testdata.isnull().sum()
```

```
Loan_ID               0
Gender               11
Married               0
Dependents           10
Education             0
Self_Employed        23
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            5
Loan_Amount_Term      6
Credit_History       29
```

```
Property_Area         0
dtype: int64
```

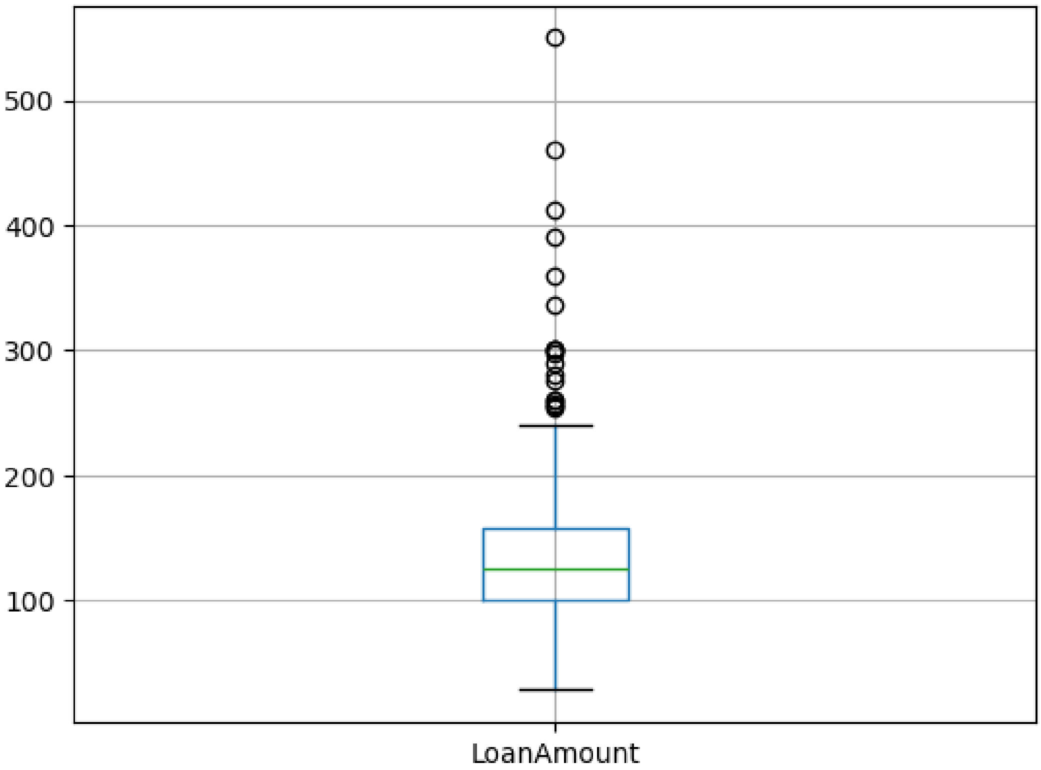
```
testdata['Gender'].fillna(testdata['Gender'].mode()[0],inplace=True)
testdata['Dependents'].fillna(testdata['Dependents'].mode()
[0],inplace=True)
testdata['Self_Employed'].fillna(testdata['Self_Employed'].mode()
[0],inplace=True)
testdata['Loan_Amount_Term'].fillna(testdata['Loan_Amount_Term'].mode(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

```
testdata.isnull().sum()
```

```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           5
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
dtype: int64
```

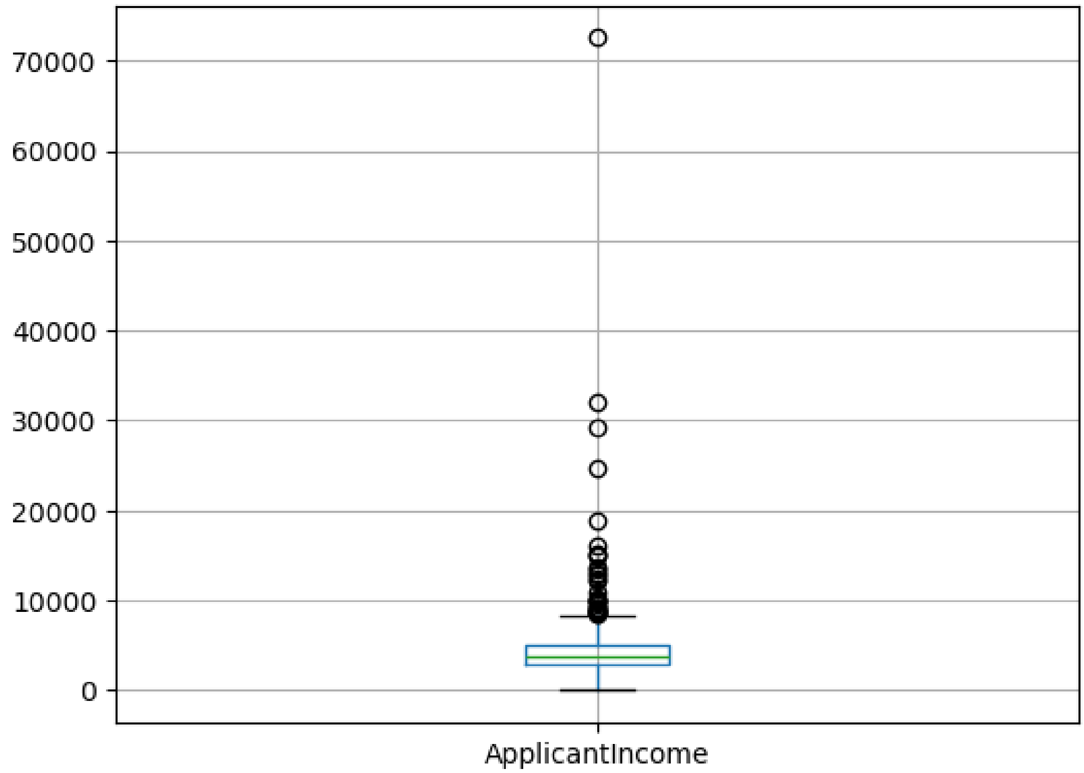
```
testdata.boxplot(column='LoanAmount')
```

```
<Axes: >
```



```
testdata.boxplot(column='ApplicantIncome')
```

```
<Axes: >
```



```
testdata.LoanAmount=
testdata.LoanAmount.fillna(testdata.LoanAmount.mean())

testdata['LoanAmount_log'] = np.log(testdata['LoanAmount'])

testdata.isnull().sum()
```

```
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
LoanAmount_log       0
dtype: int64
```

```
testdata['TotalIncome'] = testdata['ApplicantIncome'] +
testdata['CoapplicantIncome']
testdata['TotalIncome_log'] =  np.log(testdata['TotalIncome'])
```

```
testdata.head()
```

```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001015   Male     Yes          0      Graduate            No
1  LP001022   Male     Yes          1      Graduate            No
2  LP001031   Male     Yes          2      Graduate            No
3  LP001035   Male     Yes          2      Graduate            No
4  LP001051   Male      No          0  Not Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5720                  0       110.0             360.0
1             3076               1500       126.0             360.0
2             5000               1800       208.0             360.0
3             2340               2546       100.0             360.0
4             3276                  0        78.0             360.0

   Credit_History Property_Area  LoanAmount_log  TotalIncome
TotalIncome_log
0             1.0         Urban        4.700480         5720
8.651724
1             1.0         Urban        4.836282         4576
8.428581
2             1.0         Urban        5.337538         6800
8.824678
3             1.0         Urban        4.605170         4886
8.494129
4             1.0         Urban        4.356709         3276
8.094378
```

```
test = testdata.iloc[:,np.r_[1:5,9:11,13:15]].values

for i in range(0,5):
    test[:,i]= labelEncoder_x.fit_transform(test[:,i])

test[:,7] = labelEncoder_x.fit_transform(test[:,i])

test
```

```
array([[1, 1, 0, ..., 1.0, 5720, 10],
       [1, 1, 1, ..., 1.0, 4576, 10],
       [1, 1, 2, ..., 1.0, 6800, 10],
       ...,
       [1, 0, 0, ..., 1.0, 5243, 10],
       [1, 1, 0, ..., 1.0, 7393, 10],
       [1, 0, 0, ..., 1.0, 9200, 6]], dtype=object)
```

```
test = ss.fit_transform(test)

pred = NBclassifier.predict(test)

pred
```

```
array([1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1,
1,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1,
0,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0,
1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1,
0,
       1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```