Loan Eligibility Prediction

October 9, 2024

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

[5]:
```
dataset = pd.read_csv(r"C:\Users\ADMIN\Desktop\Future Interns\Loan␣
↪Prediction\loan.csv")
```

[9]:
```
#First few rows of the dataset
dataset.head()
```

[9]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

[11]:
```
#shape of the data
dataset.shape
```

[11]:
```
(614, 13)
```

[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
```

[15]:
```
dataset.describe()
```

[15]:

	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	

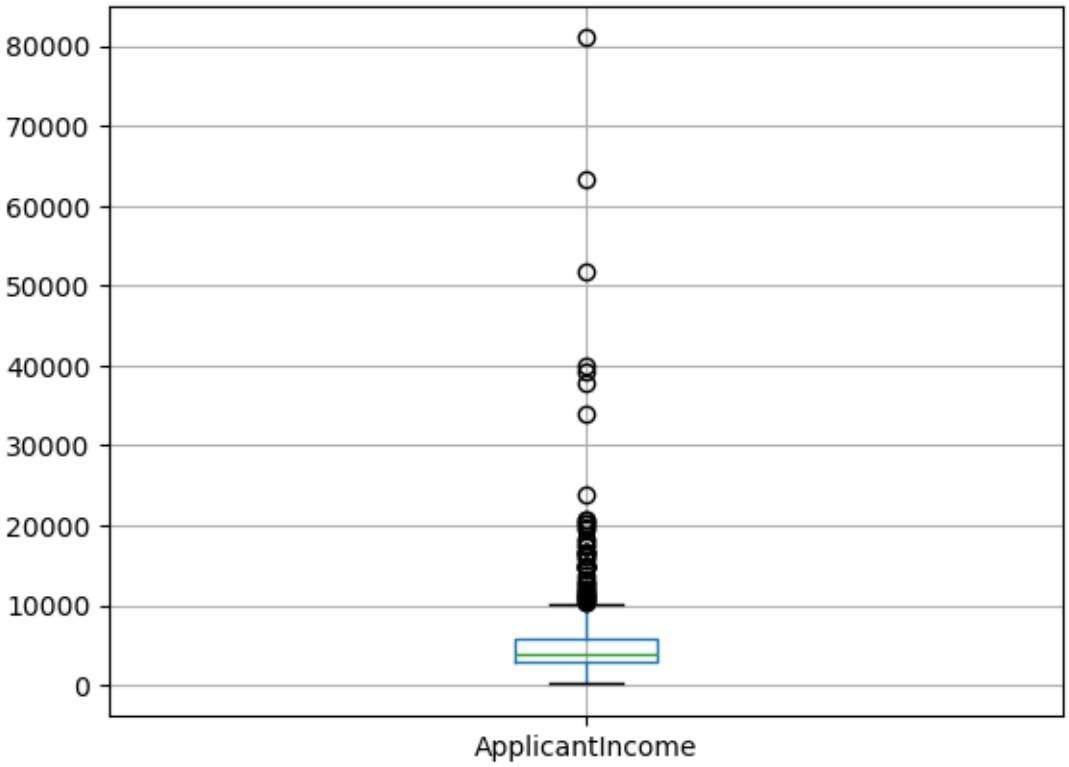
	Credit_History
count	564.000000
mean	0.842199
std	0.364878
min	0.000000
25%	1.000000
50%	1.000000
75%	1.000000
max	1.000000

[19]:
```
#understand how credit history affects the loan status of each applicant
pd.crosstab(dataset['Credit_History'],dataset['Loan_Status'],margins=True)
```

[19]:
```
Loan_Status        N    Y  All
Credit_History
0.0               82    7   89
1.0               97  378  475
All              179  385  564
```

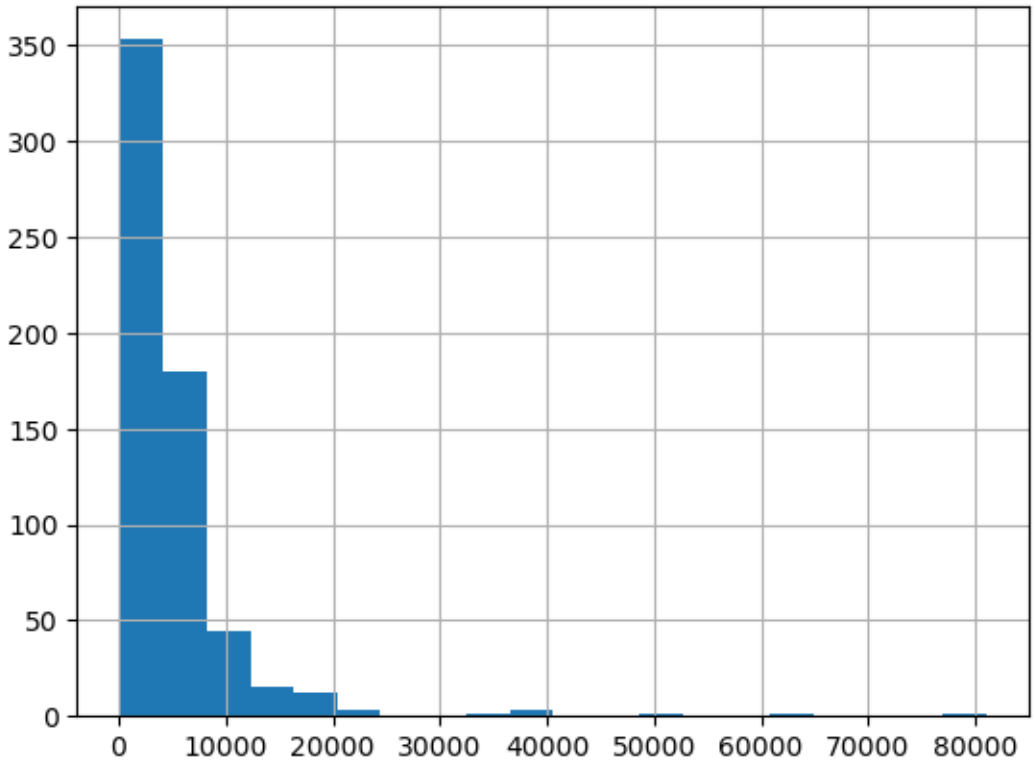
[21]:
```
dataset.boxplot(column='ApplicantIncome')
```

[21]:
```
<Axes: >
```



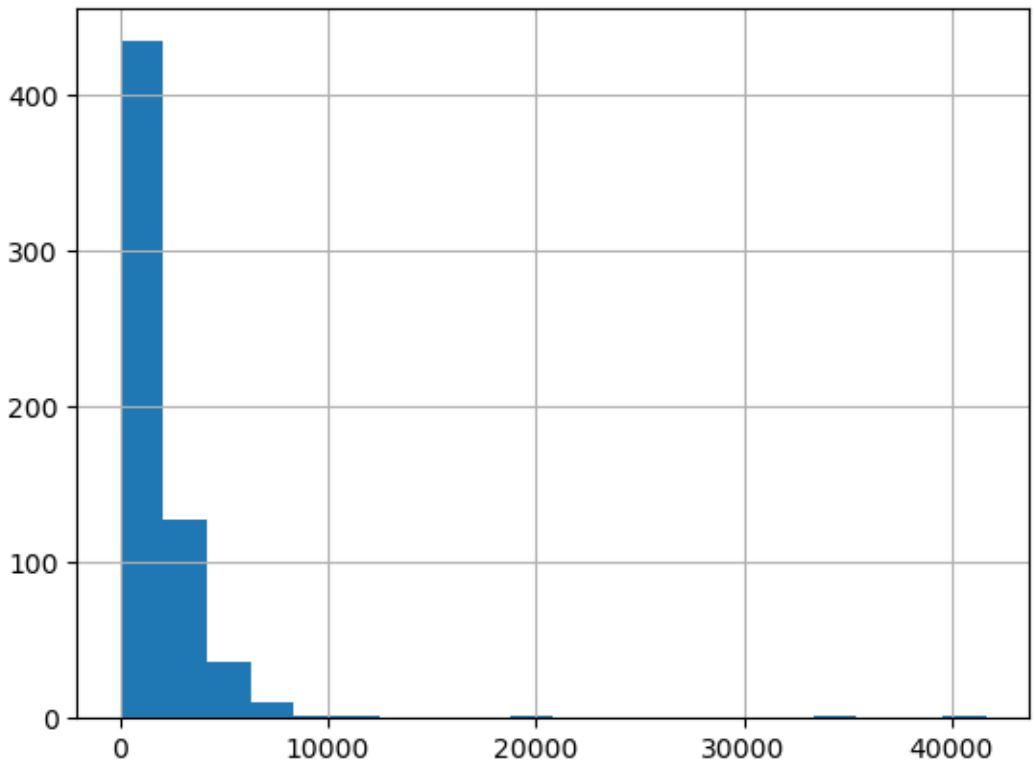
[23]:
```
dataset['ApplicantIncome'].hist(bins=20)
```

[23]:
```
<Axes: >
```



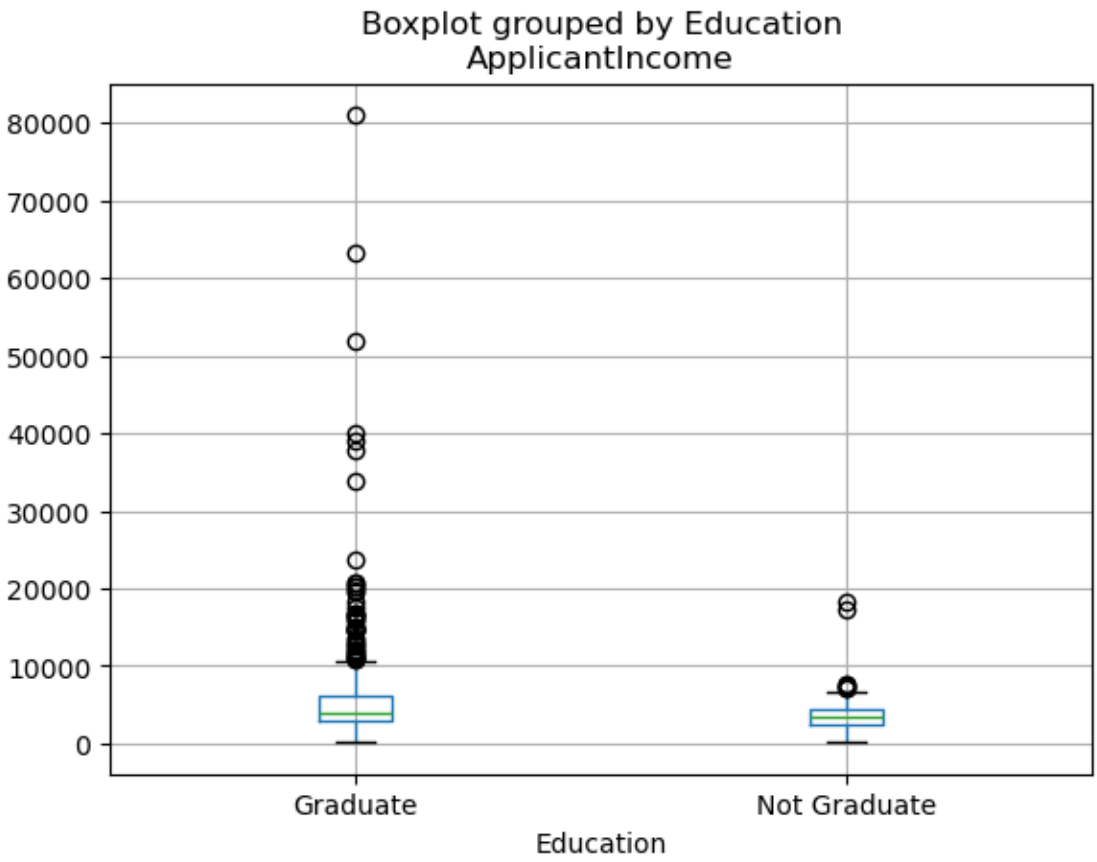
[25]:
```
dataset['CoapplicantIncome'].hist(bins=20)
```

[25]: <Axes: >



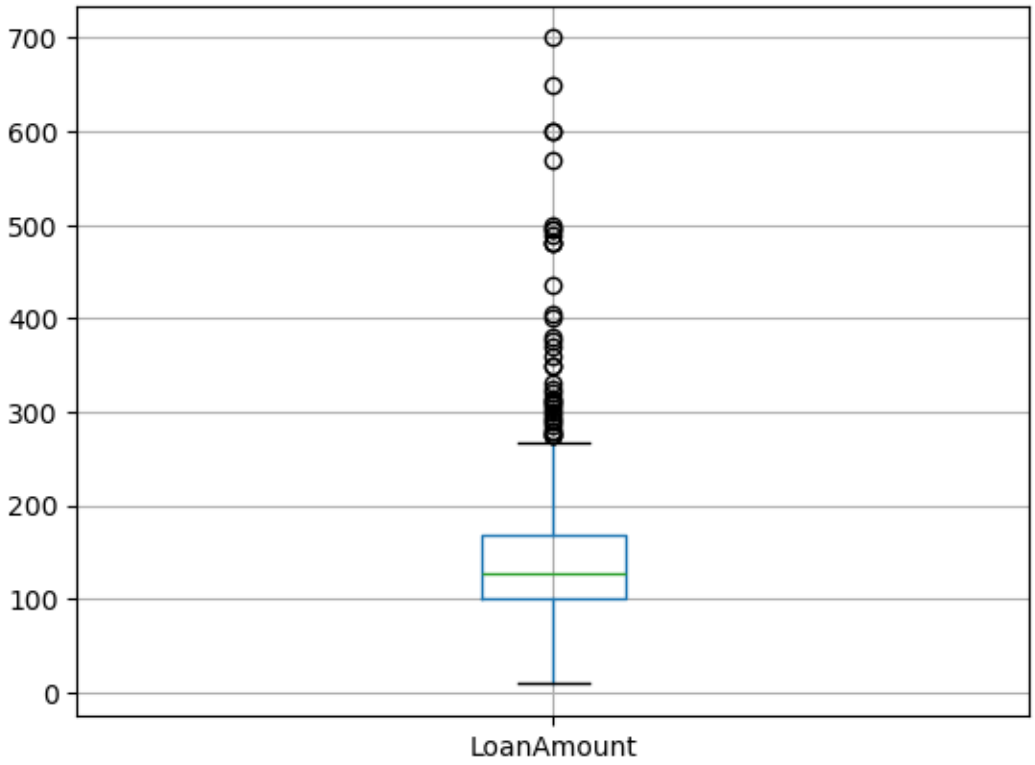
[29]:
```
dataset.boxplot(column='ApplicantIncome',by='Education')
```

[29]: <Axes: title={'center': 'ApplicantIncome'}, xlabel='Education'>



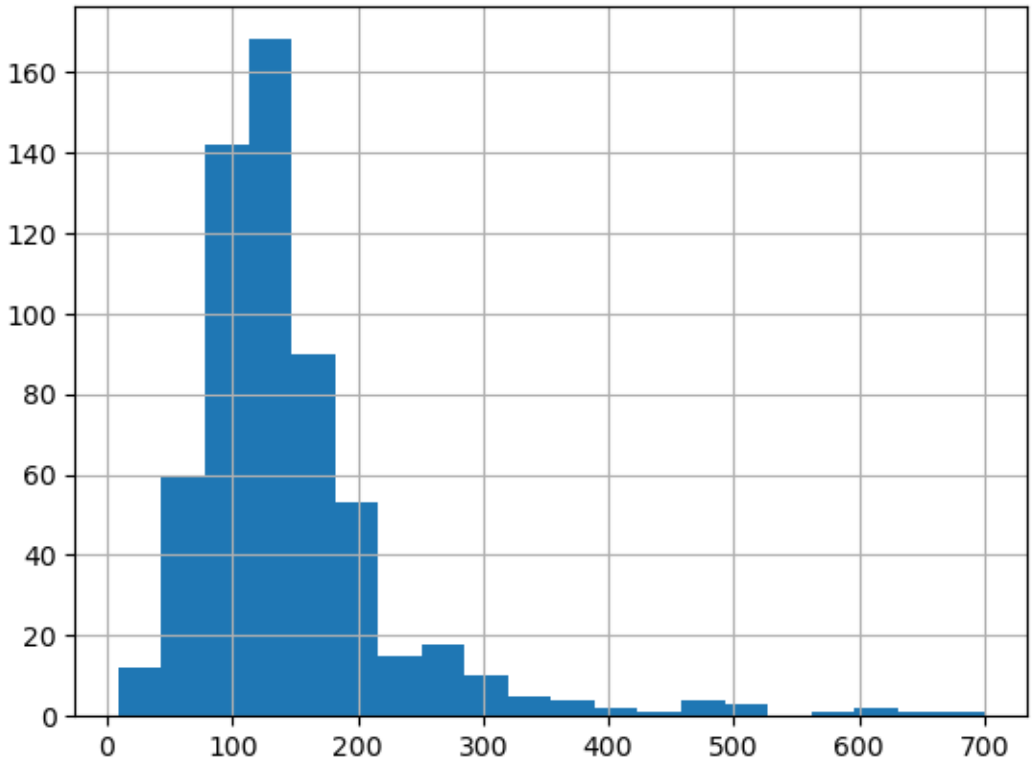
[31]:
```
dataset.boxplot(column='LoanAmount')
```

[31]: <Axes: >



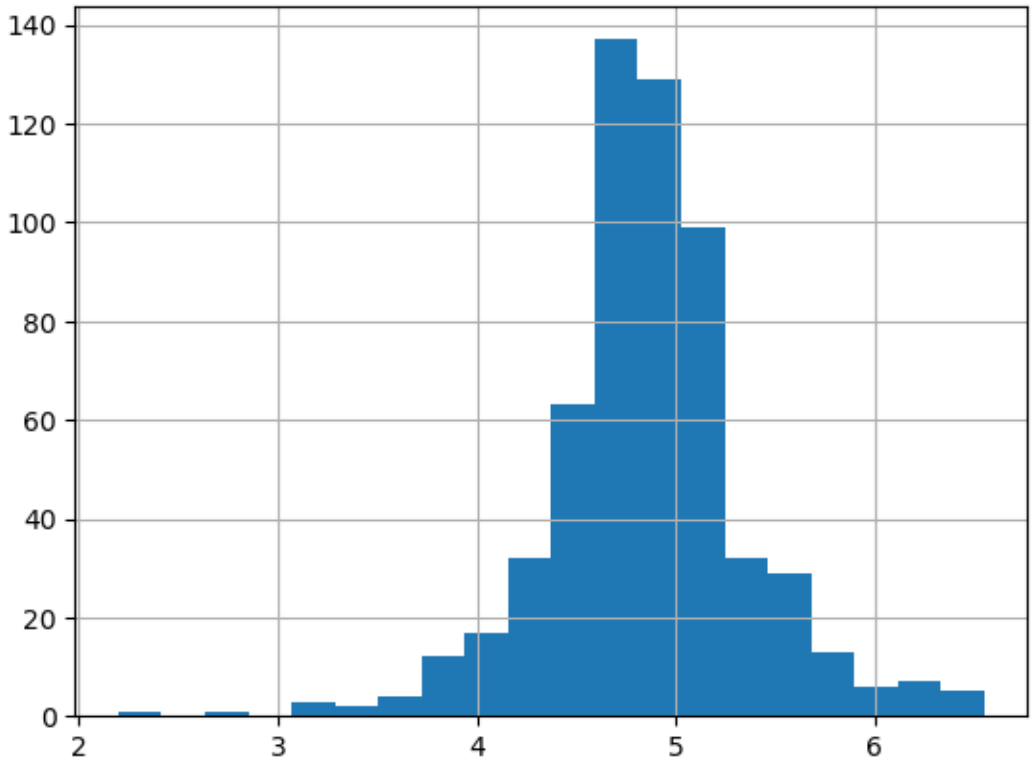
[33]:
```
dataset['LoanAmount'].hist(bins=20)
```

[33]: <Axes: >



[37]:
```
#normalizing
dataset['LoanAmount_log']=np.log(dataset['LoanAmount'])
dataset['LoanAmount_log'].hist(bins=20)
```

[37]: <Axes: >



[39]:
```
#missing values
dataset.isnull().sum()
```

[39]:
```
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
```

[59]:
```
dataset['Gender'] = dataset['Gender'].fillna(dataset['Gender'].mode()[0])
```

[61]:
```
dataset['Married'] = dataset['Married'].fillna(dataset['Married'].mode()[0])
```

[63]:
```
dataset['Dependents'] = dataset['Dependents'].fillna(dataset['Dependents'].
↪mode()[0])
```

[65]:
```
dataset['Self_Employed'] = dataset['Self_Employed'].
↪fillna(dataset['Self_Employed'].mode()[0])
```

[67]:
```
dataset['LoanAmount'] = dataset['LoanAmount'].fillna(dataset['LoanAmount'].
↪mean())
dataset['LoanAmount_log'] = dataset['LoanAmount_log'].
↪fillna(dataset['LoanAmount'].mean())
```

[69]:
```
dataset['Loan_Amount_Term'] = dataset['Loan_Amount_Term'].
↪fillna(dataset['Loan_Amount_Term'].mode()[0])
```

[71]:
```
dataset['Credit_History'] = dataset['Credit_History'].
↪fillna(dataset['Credit_History'].mode()[0])
```

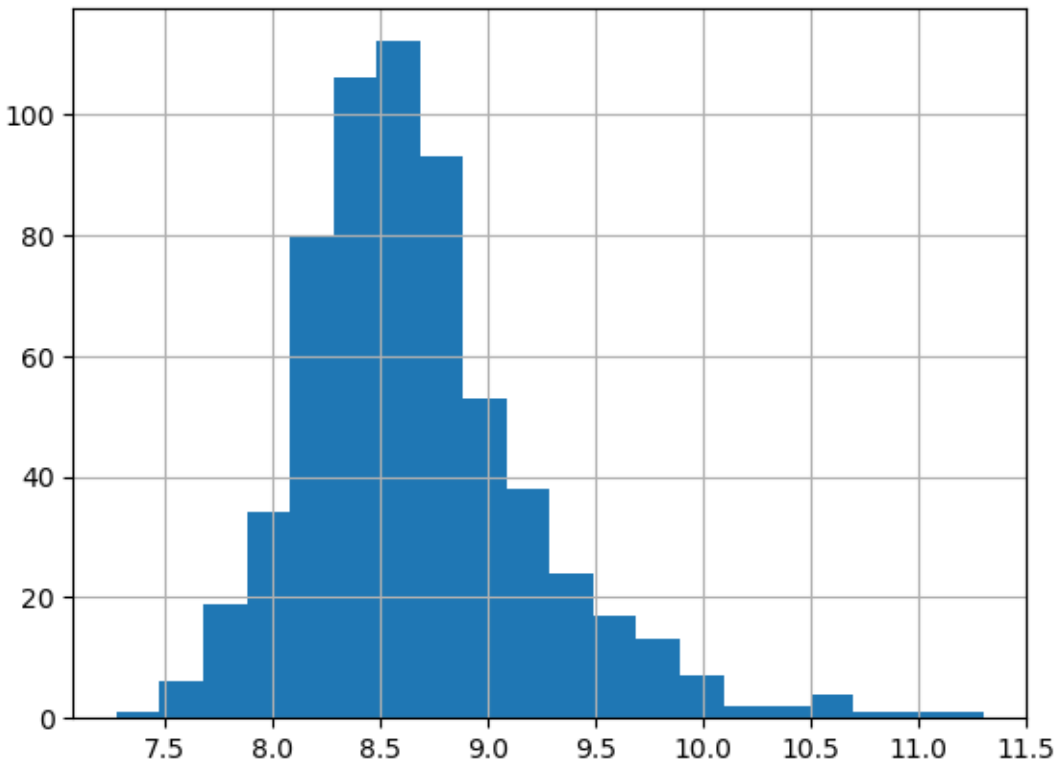
[77]:
```
dataset.isnull().sum()
```

[77]:
```
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
```

[84]:
```
dataset['TotalIncome'] = dataset['ApplicantIncome'] +␣
↪dataset['CoapplicantIncome']
dataset['TotalIncome_log'] = np.log(dataset['TotalIncome'])
```

[86]:
```
dataset['TotalIncome_log'].hist(bins=20)
```

[86]:
```
<Axes: >
```



[88]:
```
dataset.head()
```

[88]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status	LoanAmount_log	TotalIncome
0	LP001002	Male	No	0	Graduate	No	5849	0.0	146.412162	360.0	1.0	Urban	Y	146.412162	5849.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.000000	360.0	1.0	Rural	N	4.852030	6091.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.000000	360.0	1.0	Urban	Y	4.189655	3000.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.000000	360.0	1.0	Urban	Y	4.787492	4941.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.000000	360.0	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
```

[96]:
```
#dependent and independent variables
X= dataset.iloc[:,np.r_[1:5,9:11,13:15]].values
y= dataset.iloc[:,12].values
```

[98]:
```
X
```

[98]:
```
array([['Male', 'No', '0', …, 1.0, 146.41216216216216, 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)
```

[100]:
```
y
```

[100]:
```
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', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', '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)
```

[104]:
```
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)
```

[106]:
```
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]]
```

[114]:
```
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
```

[116]:
```
for i in range(0, 5):
    X_train[:,i]= labelencoder_X.fit_transform(X_train[:,i])
```

[118]:
```
X_train
```

[118]:
```
array([[1, 1, 0, …, 1.0, 4.875197323201151, 5858.0],
       [1, 0, 1, …, 1.0, 5.278114659230517, 11250.0],
       [1, 1, 0, …, 0.0, 5.003946305945459, 5681.0],
       …,
       [1, 1, 3, …, 1.0, 5.298317366548036, 8334.0],
       [1, 1, 0, …, 1.0, 5.075173815233827, 6033.0],
       [0, 1, 0, …, 1.0, 5.204006687076795, 6486.0]], dtype=object)
```

[122]:
```
labelencoder_y=LabelEncoder()
y_train= labelencoder_y.fit_transform(y_train)
```

[124]:
```
y_train
```

[124]:
```
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, 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])
```

[132]:
```
for i in range(0, 5):
    X_test[:,i]= labelencoder_X.fit_transform(X_test[:,i])
```

[136]:
```
X_test[:,7]= labelencoder_X.fit_transform(X_test[:,7])
```

[138]:
```
labelencoder_y=LabelEncoder()
y_test= labelencoder_y.fit_transform(y_test)
```

[140]:
```
X_test
```

```
[140]: 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, 146.41216216216216, 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, 146.41216216216216, 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)
```

[142]:
```
y_test
```

[142]:
```
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])
```

[144]:
```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
X_train=ss.fit_transform(X_train)
X_test=ss.fit_transform(X_test)
```

[146]:
```
from sklearn.tree import DecisionTreeClassifier
DTClassifier= DecisionTreeClassifier(criterion='entropy', random_state=0)
DTClassifier.fit(X_train,y_train)
```

[146]:
```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

[148]:
```
y_pred= DTClassifier.predict(X_test)
y_pred
```

[148]:
```
array([1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1])
```

[150]:
```
from sklearn import metrics
print('The accuracy of decision tree is: ', metrics.
 ↪accuracy_score(y_pred,y_test))
```

```
The accuracy of decision tree is:  0.6991869918699187
```

[152]:
```
from sklearn.naive_bayes import GaussianNB
NBClassifier = GaussianNB()
NBClassifier.fit(X_train,y_train)
```

[152]:
```
GaussianNB()
```

[154]:
```
y_pred= NBClassifier.predict(X_test)
```

[156]:
```
y_pred
```

[156]:
```
array([1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 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, 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])
```

[158]:
```
print('The accuracy of Naive Bayes is: ',metrics.accuracy_score(y_pred,y_test))
```

```
The accuracy of Naive Bayes is:  0.8130081300813008
```

[162]:
```
testdata= pd.read_csv("loan.csv")
```

[164]:
```
testdata.head()
```

[164]:

	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
```

[166]:
```
testdata.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
```

[170]:
```
testdata.isnull().sum()
```

[170]:
```
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
```

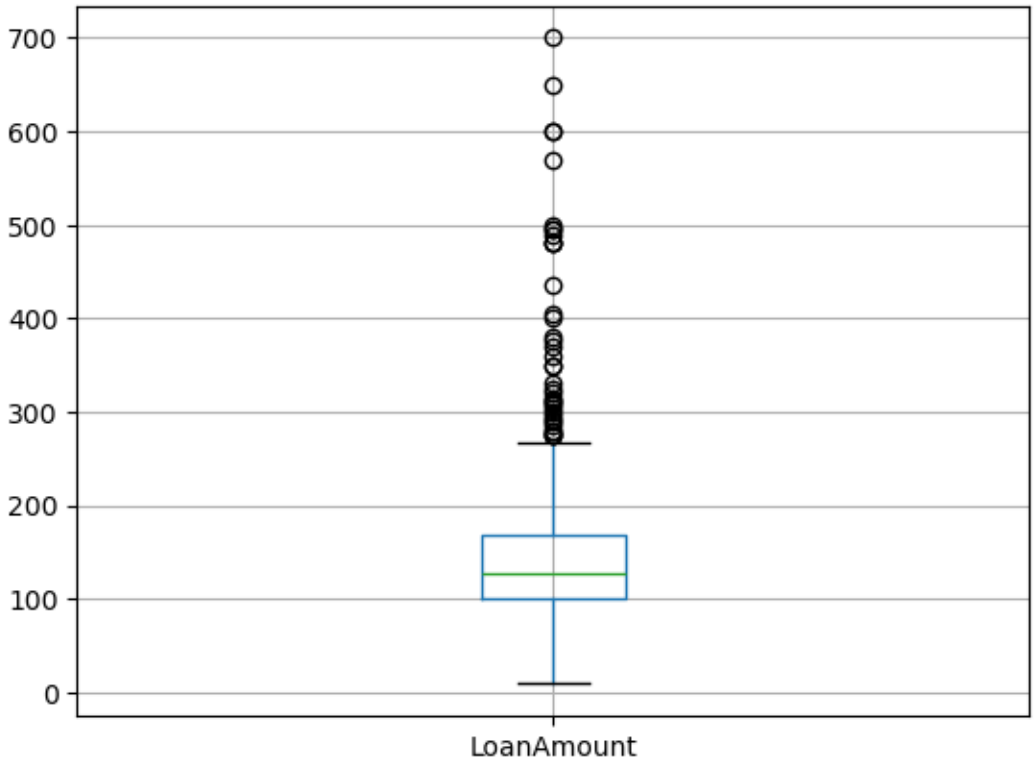
[182]:
```
testdata['Gender'] = testdata['Gender'].fillna(testdata['Gender'].mode()[0])
testdata['Married'] = testdata['Married'].fillna(testdata['Married'].mode()[0])
testdata['Dependents'] = testdata['Dependents'].fillna(testdata['Dependents'].
 ↪mode()[0])
testdata['Self_Employed'] = testdata['Self_Employed'].
 ↪fillna(testdata['Self_Employed'].mode()[0])
testdata['Loan_Amount_Term'] = testdata['Loan_Amount_Term'].
 ↪fillna(testdata['Loan_Amount_Term'].mode()[0])
testdata['Credit_History'] = testdata['Credit_History'].
 ↪fillna(testdata['Credit_History'].mode()[0])
```

[184]:
```
testdata.isnull().sum()
```

[184]:
```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount          22
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

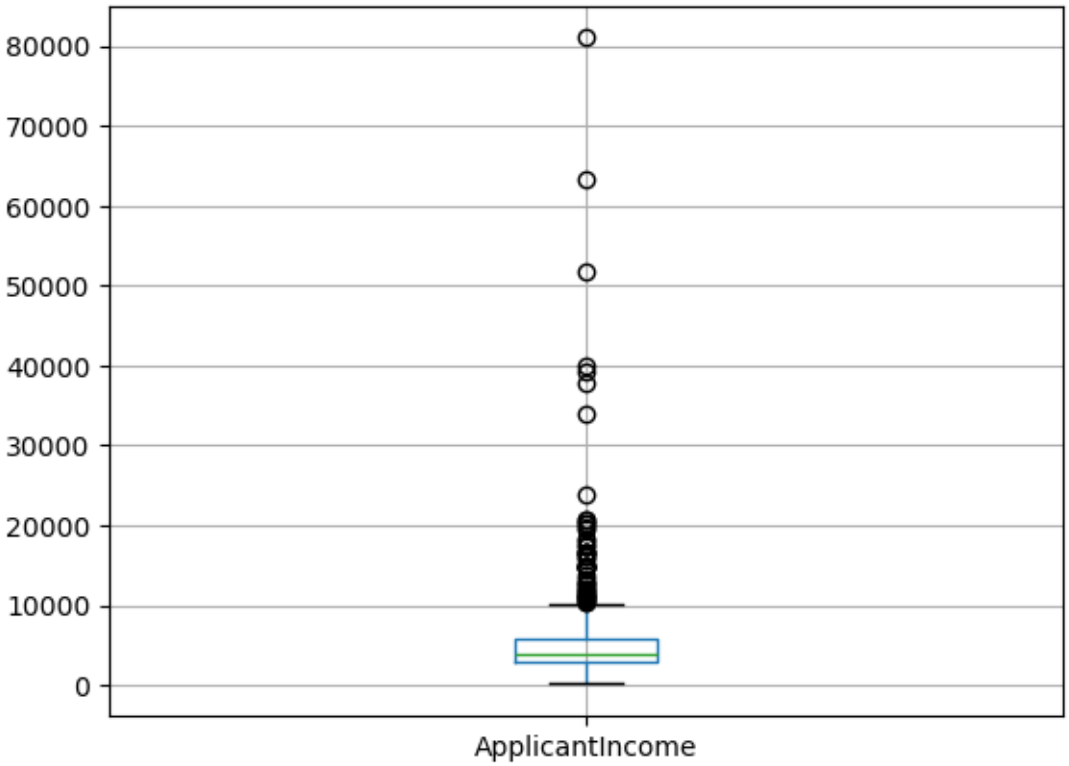
[186]:
```
testdata.boxplot(column='LoanAmount')
```

[186]:
```
<Axes: >
```



[188]:
```
testdata.boxplot(column='ApplicantIncome')
```

[188]: <Axes: >



[190]:
```
testdata.LoanAmount= testdata.LoanAmount.fillna(testdata.LoanAmount.mean())
```

[192]:
```
testdata['LoanAmount_log']=np.log(testdata['LoanAmount'])
```

[194]:
```
testdata.isnull().sum()
```

[194]:
```
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
```

[196]:
```
testdata['TotalIncome']=␣
 ↪testdata['ApplicantIncome']+testdata['CoapplicantIncome']
testdata['TotalIncome_log']= np.log(testdata['TotalIncome'])
```

[198]:
```
testdata.head()
```

[198]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status	LoanAmount_log	TotalIncome	TotalIncome_log
0	LP001002	Male	No	0	Graduate	No	5849	0.0	146.412162	360.0	1.0	Urban	Y	4.986426	5849.0	8.674026
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.000000	360.0	1.0	Rural	N	4.852030	6091.0	8.714568
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.000000	360.0	1.0	Urban	Y	4.189655	3000.0	8.006368
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.000000	360.0	1.0	Urban	Y	4.787492	4941.0	8.505323
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.000000	360.0	1.0	Urban	Y	4.948760	6000.0	8.699515

[200]:
```
test= testdata.iloc[:,np.r_[1:5,9:11,13:15]].values
```

[211]:
```
for i in range(0,5):
    test[:,i]=labelencoder_X.fit_transform(test[:,i])
```

[219]:
```
test[:,7]=labelencoder_X.fit_transform(test[:,7])
```

[221]:
```
test
```

[221]:
```
array([[1, 0, 0, …, 1.0, 4.986425672954842, 320],
       [1, 1, 1, …, 1.0, 4.852030263919617, 333],
       [1, 1, 0, …, 1.0, 4.189654742026425, 42],
       …,
```

```
       [1, 1, 1, …, 1.0, 5.53338948872752, 436],
       [1, 1, 2, …, 1.0, 5.231108616854587, 416],
       [0, 0, 0, …, 0.0, 4.890349128221754, 185]], dtype=object)
```

[223]:
```
test= ss.fit_transform(test)
```

[225]:
```
pred= NBClassifier.predict(test)
```

[227]:
```
pred
```

[227]:
```
array([1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 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,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 0, 0, 1, 1, 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, 1, 1, 0, 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, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 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, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 0, 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, 0, 1, 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, 1,
       0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0,
       1, 1, 1, 1, 0, 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, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0])
```

[ ]:
```

```