In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import svm
```

In [2]:
```
df = pd.read_csv('loan.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Terr
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.

In [4]:
```
df.info()
```

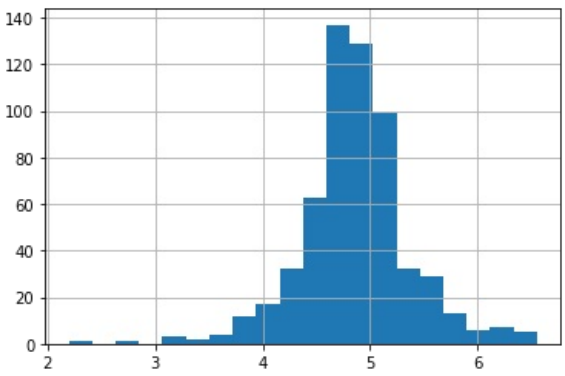
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 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
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
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
```

In [6]:
```
df['loanAmount_log']= np.log(df['LoanAmount'])
df['loanAmount_log'].hist(bins=20)
```

Out[6]:
```
<AxesSubplot:>
```

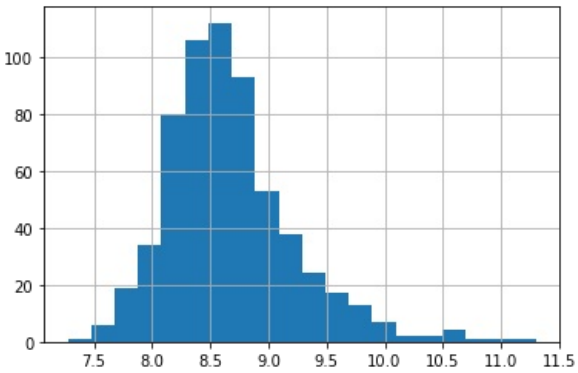


In [7]:
```
df.isnull().sum()
```

Out[7]:
```
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
```

In [8]:
```
df['TotalIncome']= df['ApplicantIncome']+ df['CoapplicantIncome']
df['TotalIncome_log']=np.log(df['TotalIncome'])
df['TotalIncome_log'].hist(bins=20)
```

Out[8]:
```
<AxesSubplot:>
```



In [9]:
```
df['Gender'].fillna(df['Gender'].mode()[0],inplace = True)
df['Married'].fillna(df['Married'].mode()[0],inplace = True)
df['Self_Employed'].fillna(df['Self_Employed'].mode()[0],inplace = True)
df['Dependents'].fillna(df['Dependents'].mode()[0],inplace = True)

df.LoanAmount = df.LoanAmount.fillna(df.LoanAmount.mean())
df.loanAmount_log = df.loanAmount_log.fillna(df.loanAmount_log.mean())

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

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

Out[9]:
```
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
TotalIncome          0
TotalIncome_log      0
dtype: int64
```

In [10]:
```
x= df.iloc[:,np.r_[1:5, 9:11, 13:15]].values
y= df.iloc[:,12].values

x
```

Out[10]:
```
array([['Male', 'No', '0', ..., 1.0, 4.857444178729353, 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)
```

In [11]:
```
y
```

Out[11]:
```
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)
```

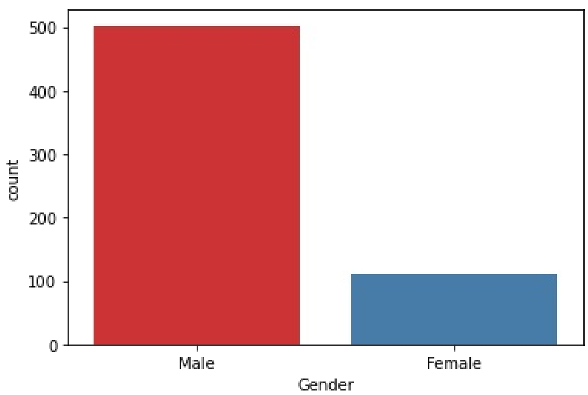
In [12]:
```python
print("per of missing gender is 2%f%%" %((df['Gender'].isnull().sum()/df.shape[0])*100))
```

```
per of missing gender is 20.000000%
```

In [13]:
```python
print("number of people who take loan as group by gender:")
print(df['Gender'].value_counts())
sns.countplot(x='Gender', data=df, palette = 'Set1')
```

```
number of people who take loan as group by gender:
Male      502
Female    112
Name: Gender, dtype: int64
```

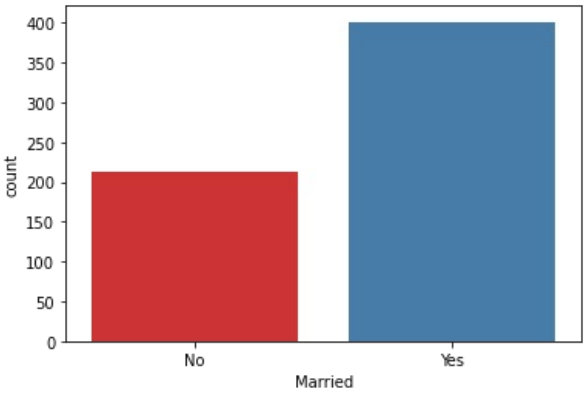
Out[13]:
```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



In [14]:
```python
print("number of people who take loan as group by marital status:")
print(df['Married'].value_counts())
sns.countplot(x='Married', data=df, palette = 'Set1')
```

```
number of people who take loan as group by marital status:
Yes    401
No     213
Name: Married, dtype: int64
```

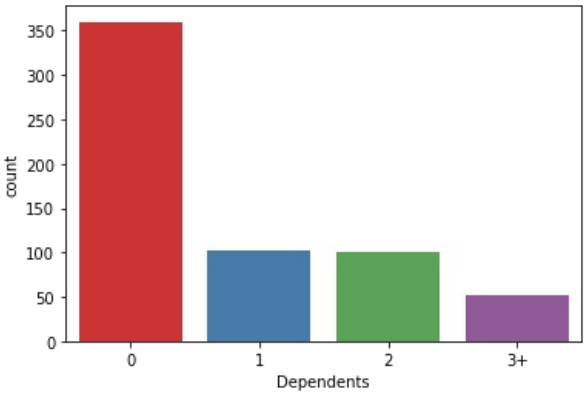
Out[14]:
```
<AxesSubplot:xlabel='Married', ylabel='count'>
```



In [15]:
```
print("number of people who take loan as group by dependents:")
print(df['Dependents'].value_counts())
sns.countplot(x='Dependents', data=df, palette = 'Set1')
```

```
number of people who take loan as group by dependents:
0     360
1     102
2     101
3+     51
Name: Dependents, dtype: int64
```

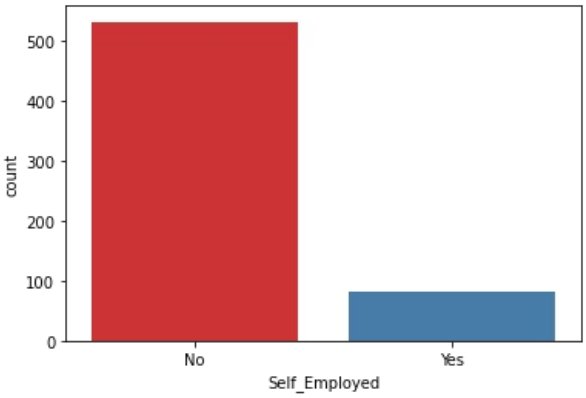
Out[15]:
```
<AxesSubplot:xlabel='Dependents', ylabel='count'>
```



In [16]:
```
print("number of people who take loan as group by self employed:")
print(df['Self_Employed'].value_counts())
sns.countplot(x='Self_Employed', data=df, palette = 'Set1')
```

```
number of people who take loan as group by self employed:
No     532
Yes     82
Name: Self_Employed, dtype: int64
```

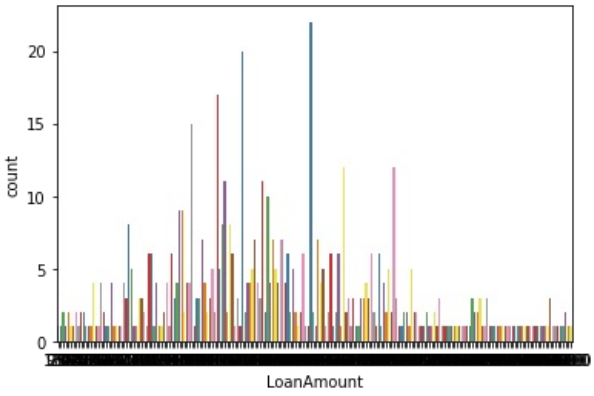
Out[16]:
```
<AxesSubplot:xlabel='Self_Employed', ylabel='count'>
```



In [17]:
```
print("number of people who take loan as group by Loanamount:")
print(df['LoanAmount'].value_counts())
sns.countplot(x='LoanAmount', data=df, palette = 'Set1')
```

```
number of people who take loan as group by Loanamount:
146.412162    22
120.000000    20
110.000000    17
100.000000    15
160.000000    12
              ..
240.000000     1
214.000000     1
59.000000      1
166.000000     1
253.000000     1
Name: LoanAmount, Length: 204, dtype: int64
```

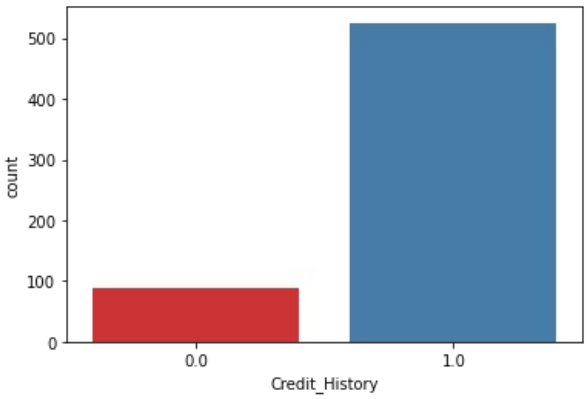
Out[17]:
```
<AxesSubplot:xlabel='LoanAmount', ylabel='count'>
```



In [18]:
```
print("number of people who take loan as group by self Credit history:")
print(df['Credit_History'].value_counts())
sns.countplot(x='Credit_History', data=df, palette = 'Set1')
```

```
number of people who take loan as group by self Credit history:
1.0    525
0.0     89
Name: Credit_History, dtype: int64
```

Out[18]:
```
<AxesSubplot:xlabel='Credit_History', ylabel='count'>
```



In [19]:
```
from sklearn.model_selection import train_test_split
```

In [20]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state= 0)

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

In [21]:
```
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
```

Out[21]:
```
array([[1, 1, 0, ..., 1.0, 4.875197323201151, 267],
       [1, 0, 1, ..., 1.0, 5.278114659230517, 407],
       [1, 1, 0, ..., 0.0, 5.003946305945459, 249],
       ...,
       [1, 1, 3, ..., 1.0, 5.298317366548036, 363],
       [1, 1, 0, ..., 1.0, 5.075173815233827, 273],
       [0, 1, 0, ..., 1.0, 5.204006687076795, 301]], dtype=object)
```

In [22]:
```
Labelencoder_y = LabelEncoder()
y_train = Labelencoder_y.fit_transform(y_train)

y_train
```

Out[22]:
```
array([1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 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, 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, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
       0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 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, 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])
```

In [23]:
```
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])

x_test
```

Out[23]:
```
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, 4.857444178729353, 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, 4.857444178729353, 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)
```

In [24]:
```
Labelencoder_y = LabelEncoder()

y_test= Labelencoder_y.fit_transform(y_test)

y_test
```

Out[24]:
```
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, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 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])
```

In [25]:
```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()
X_train = ss.fit_transform(x_train)
x_test = ss.fit_transform(x_test)
```

In [26]:
```
from sklearn.tree import DecisionTreeClassifier
DT=DecisionTreeClassifier(criterion='entropy',random_state=0)
DT.fit(x_train,y_train)
```

Out[26]:
```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [27]:
```python
y_pred=DT.predict(x_test)
y_pred
```

Out[27]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [28]:
```python
from sklearn import metrics
ac_dt=metrics.accuracy_score(y_pred,y_test)
print("Accuracy of Decision Tree Classifier is: ",ac_dt)
```

```
Accuracy of Decision Tree Classifier is:  0.2682926829268293
```

In [29]:
```python
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
kf_dt=KFold(n_splits=5,shuffle=True)
cv_dt=cross_val_score(DT,X=x_train,y=y_train,cv=kf_dt)
Ac_DT=np.mean(cv_dt)
Ac_DT
```

Out[29]:
```
0.6925376211090497
```

In [30]:
```python
from sklearn.ensemble import RandomForestClassifier

rf_clf = RandomForestClassifier()
rf_clf.fit(x_train, y_train)
```

Out[30]:
```
RandomForestClassifier()
```

In [31]:
```python
from sklearn import metrics
y_pred = rf_clf.predict(x_test)

print("acc of random forest clf is", metrics.accuracy_score(y_pred, y_test))

y_pred
```

```
acc of random forest clf is 0.2682926829268293
```

Out[31]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [32]:
```python
from sklearn.naive_bayes import GaussianNB
nb_classifier = GaussianNB()
nb_classifier.fit(x_train, y_train)
```

Out[32]:
```
GaussianNB()
```

In [33]:
```python
y_pred = nb_classifier.predict(x_test)
print("acc of gaussianNB is %.", metrics.accuracy_score(y_pred, y_test) )
```

```
acc of gaussianNB is %. 0.2682926829268293
```

In [34]:
```python
y_pred
```

Out[34]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [35]:
```python
ac_nb=metrics.accuracy_score(y_pred,y_test)
print("Accuracy of Naive Bayes is: ",ac_nb)
```

```
Accuracy of Naive Bayes is:  0.2682926829268293
```

In [36]:
```python
kf_nb=KFold(n_splits=5,shuffle=True)
cv_nb=cross_val_score(nb_classifier,X=x_train,y=y_train,cv=kf_nb)
AC_NB=np.mean(cv_nb)
AC_NB
```

Out[36]:
```
0.8003710575139147
```

In [37]:
```python
from sklearn.svm import SVC
svm=SVC()
svm.fit(x_train,y_train)
```

Out[37]:
```
SVC()
```

In [38]:
```python
y_pred=svm.predict(x_test)
```

In [38]:
```
y_pred=svm.predict(x_test)
y_pred
```

Out[38]:
```
array([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, 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, 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, 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, 1, 1, 1, 1, 1, 1, 1])
```

In [39]:
```
ac_svm=metrics.accuracy_score(y_pred,y_test)
print("Accuracy of Support Vector Machine is: ",ac_svm)
```

```
Accuracy of Support Vector Machine is:  0.7317073170731707
```

In [40]:
```
from sklearn.neighbors import KNeighborsClassifier
kn_clf = KNeighborsClassifier()
kn_clf.fit(x_train, y_train)
```

Out[40]:
```
KNeighborsClassifier()
```

In [41]:
```
y_pred= kn_clf.predict(x_test)
print("acc of KN is", metrics.accuracy_score(y_pred, y_test))
```

```
acc of KN is 0.2682926829268293
```

In [42]:
```
compare=pd.DataFrame({'Model':['Decision Tree','Naive bayes','Support Vector Machine'],'Accuracy':[ac_dt*100,ac
compare
```

Out[42]:

	Model	Accuracy
0	Decision Tree	26.829268
1	Naive bayes	26.829268
2	Support Vector Machine	73.170732

In [43]:
```
testdata=pd.read_csv('loan.csv')
```

In [44]:
```
testdata.isnull().sum()
```

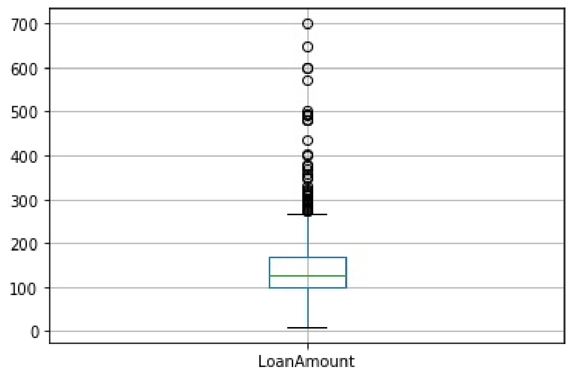
Out[44]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [45]:
```
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)
```

In [46]:
```
testdata['LoanAmount_log']=np.log(testdata['LoanAmount'])
```

In [47]:
```
testdata.boxplot(column="LoanAmount")
```

Out[47]:
```
&lt;AxesSubplot:&gt;
```



In [48]:
```
testdata['TotalIncome']=testdata['ApplicantIncome']+testdata['CoapplicantIncome']
testdata['TotalIncome_log']=np.log(testdata['TotalIncome'])
```

In [49]:
```
testdata.head()
```

Out[49]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Terr
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.

In [50]:
```
df.to_csv('output.csv',index=False)
```

In [51]:
```
pd.read_csv('output.csv')
```

Out[51]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_T
0	LP001002	Male	No	0	Graduate	No	5849	0.0	146.412162	3
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.000000	3
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.000000	3
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.000000	3
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.000000	3
...	...	...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.000000	3
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.000000	1
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.000000	3
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.000000	3
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.000000	3

614 rows × 16 columns

In [ ]: