In [9]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import svm
```

In [10]:
```python
df = pd.read_csv("loan_prediction.csv")
```

In [11]:
```python
df.head()
```

Out[11]:

	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

In [12]:
```python
df.info()
```

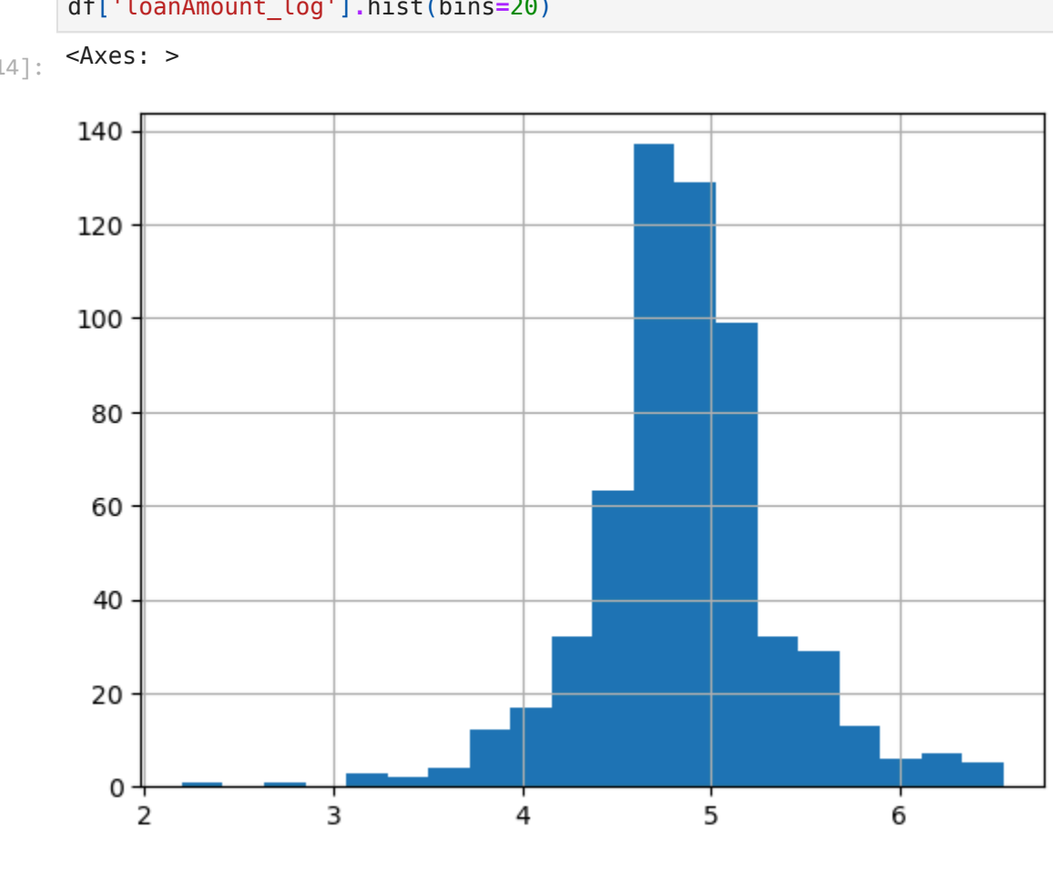
```
<class 'pandas.core.frame.DataFrame'>
Index: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            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: 67.2+ KB
```

In [13]:
```python
df.isnull().sum()
```

Out[13]:
```
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 [14]:
```python
df['loanAmount_log'] = np.log(df['LoanAmount'])
df['loanAmount_log'].hist(bins=20)
```

Out[14]:
```
<Axes: >
```

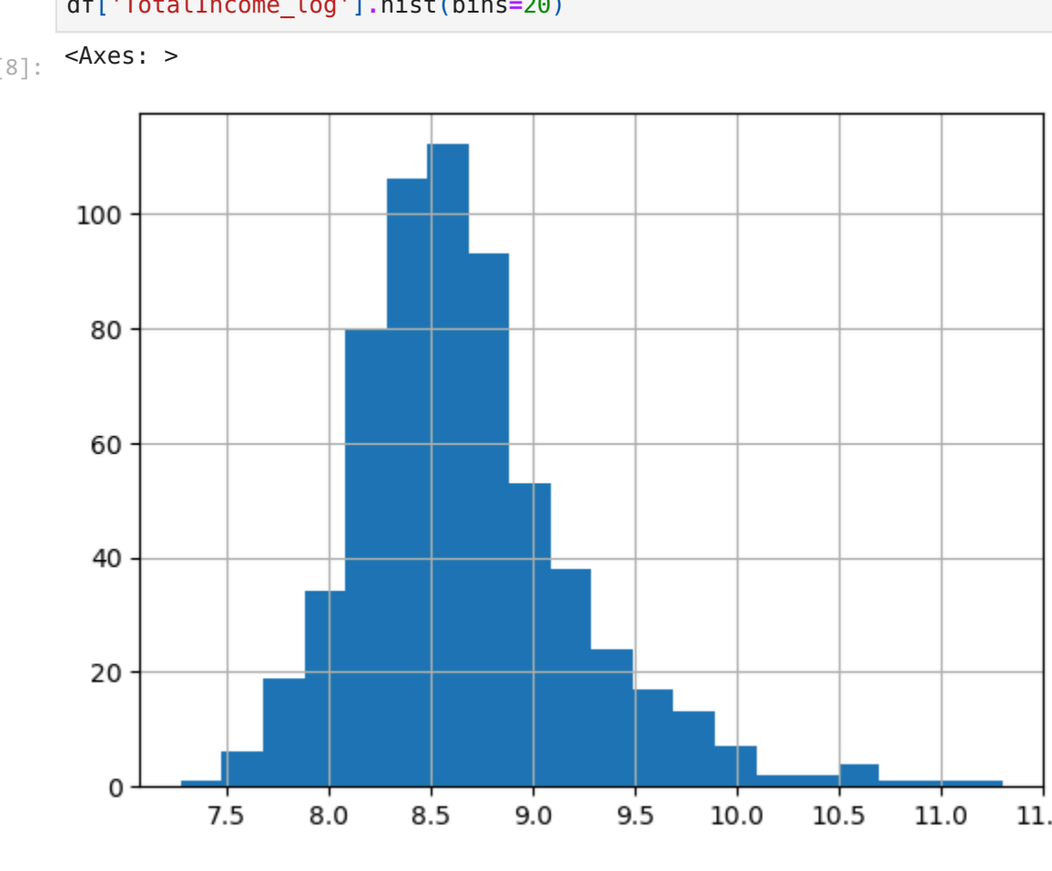


In [15]:
```python
df.isnull().sum()
```

Out[15]:
```
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]:
```python
df['TotalIncome'] = df['ApplicantIncome'] + df['CoapplicantIncome']
df['TotalIncome_log'] = np.log(df['TotalIncome'])
df['TotalIncome_log'].hist(bins=20)
```

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



In [23]:
```python
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[23]:
```
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
```

In [28]:
```python
x= df.iloc[:,np.r_[1:5,9:11,13:14]].values
y= df.iloc[:,12].values

x
```

Out[28]:
```
array([['Male', 'No', '0', ..., 360.0, 1.0, 4.857444178729352],
       ['Male', 'Yes', '1', ..., 360.0, 1.0, 4.852030263919617],
       ['Male', 'Yes', '0', ..., 360.0, 1.0, 4.189654742026425],
       ...,
       ['Male', 'Yes', '1', ..., 360.0, 1.0, 5.53338948872752],
       ['Male', 'Yes', '2', ..., 360.0, 1.0, 5.231108616854587],
       ['Female', 'No', '0', ..., 360.0, 0.0, 4.890349128221754]],
      dtype=object)
```

In [29]:
```python
y
```

Out[29]:
```
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', 'Y', 'N', 'N', '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', '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', 'Y', 'Y',
       'Y', 'N', 'Y', 'Y', 'Y', 'N', 'N', 'N', 'Y', 'N', 'N', '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', 'Y', 'N',
       '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', 'N', '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', 'Y', '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)
```

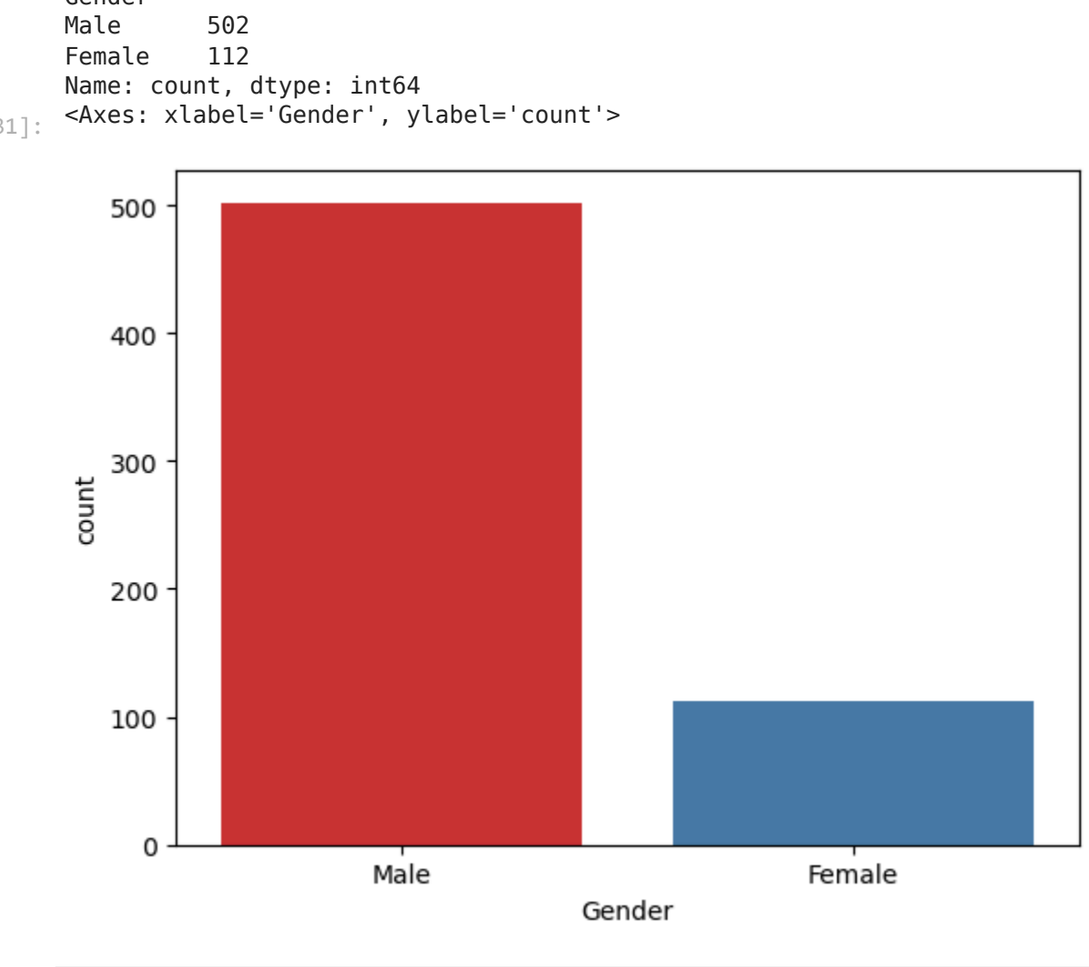
In [30]:
```python
print("per of missing gender is %2f%%" %((df['Gender'].isnull().sum()/df.shape[0])*100))
```

```
per of missing gender is 0.000000%
```

In [31]:
```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:
Gender
Male      502
Female    112
Name: count, dtype: int64
```

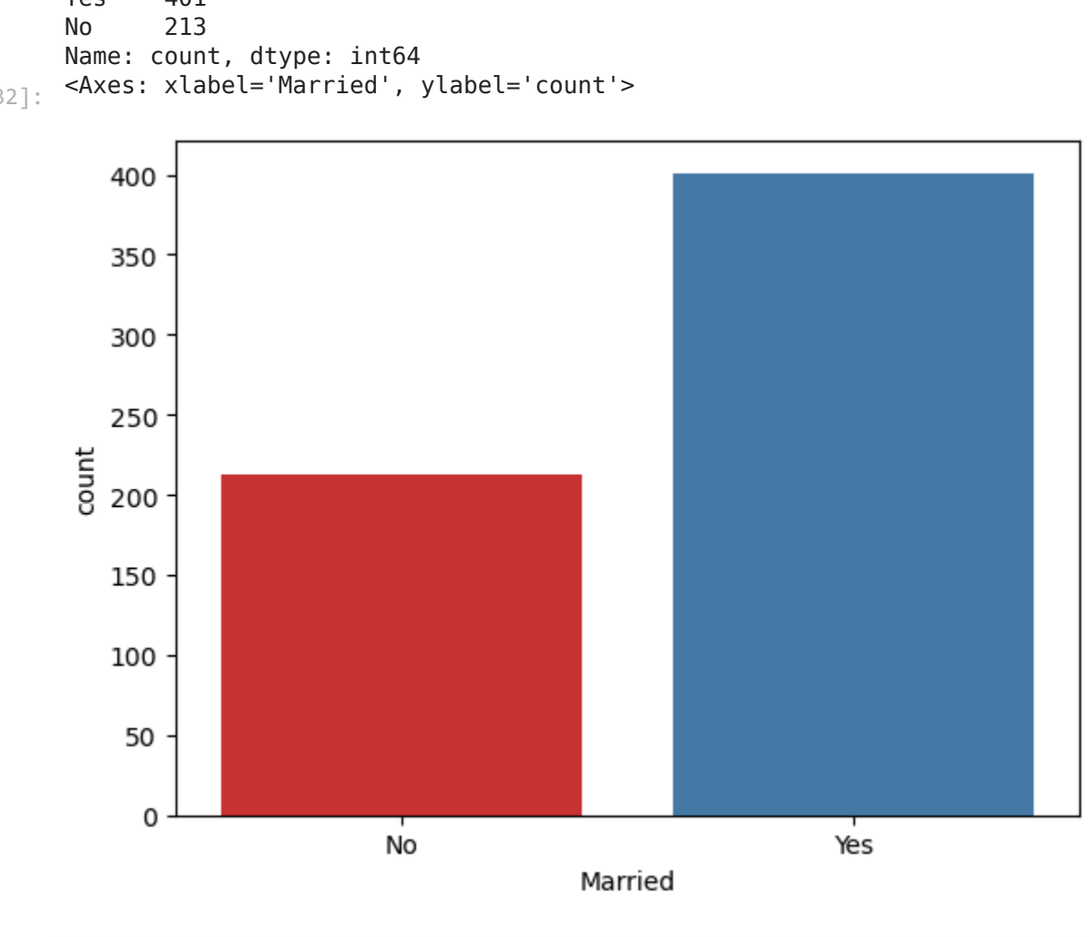
Out[31]:
```
<Axes: xlabel='Gender', ylabel='count'>
```



In [32]:
```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:
Married
Yes    401
No     213
Name: count, dtype: int64
```

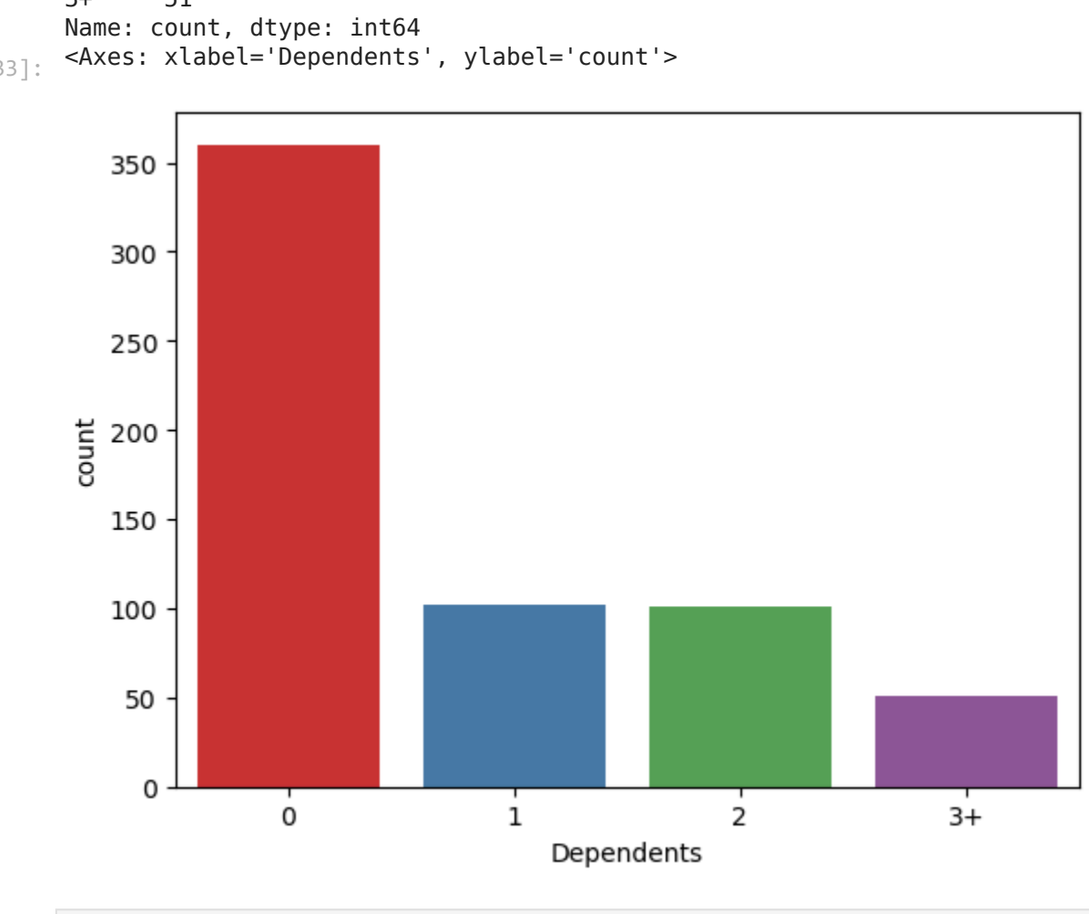
Out[32]:
```
<Axes: xlabel='Married', ylabel='count'>
```



In [33]:
```python
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:
Dependents
0     360
1     102
2     101
3+     51
Name: count, dtype: int64
```

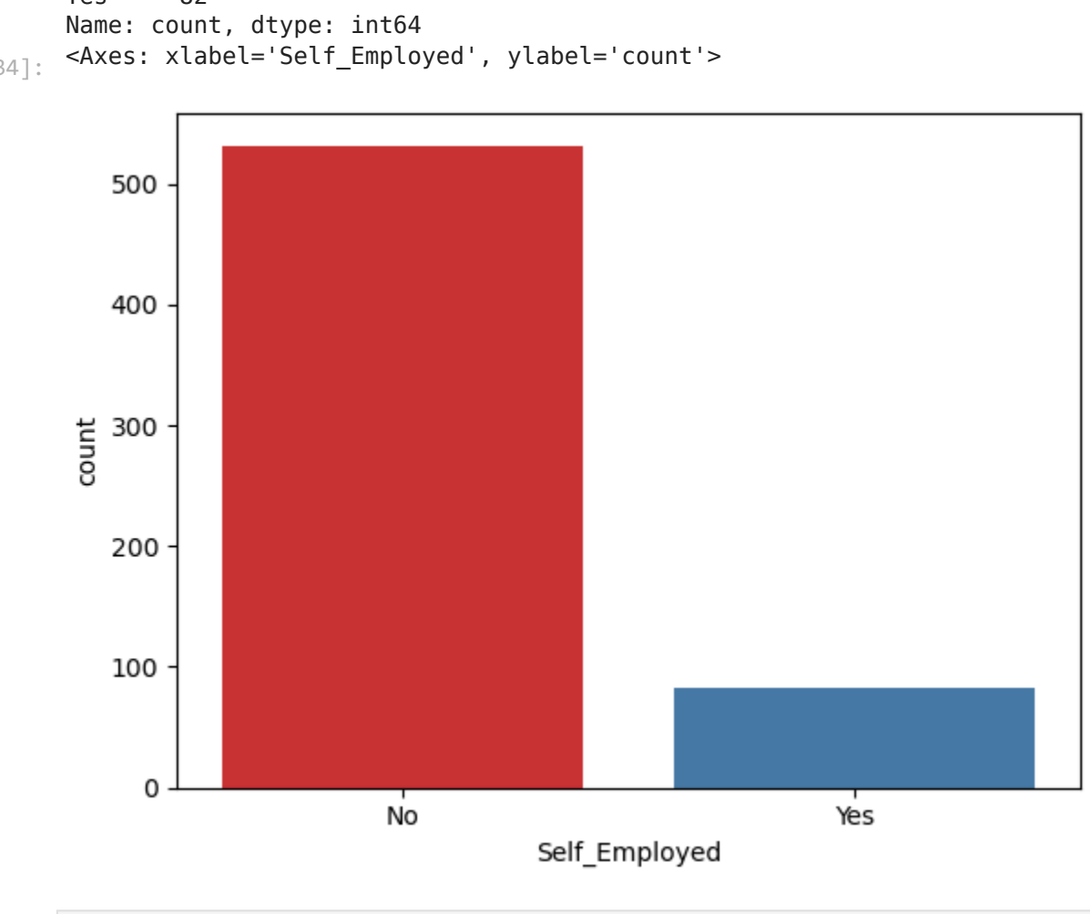
Out[33]:
```
<Axes: xlabel='Dependents', ylabel='count'>
```



In [34]:
```python
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:
Self_Employed
No     532
Yes     82
Name: count, dtype: int64
```

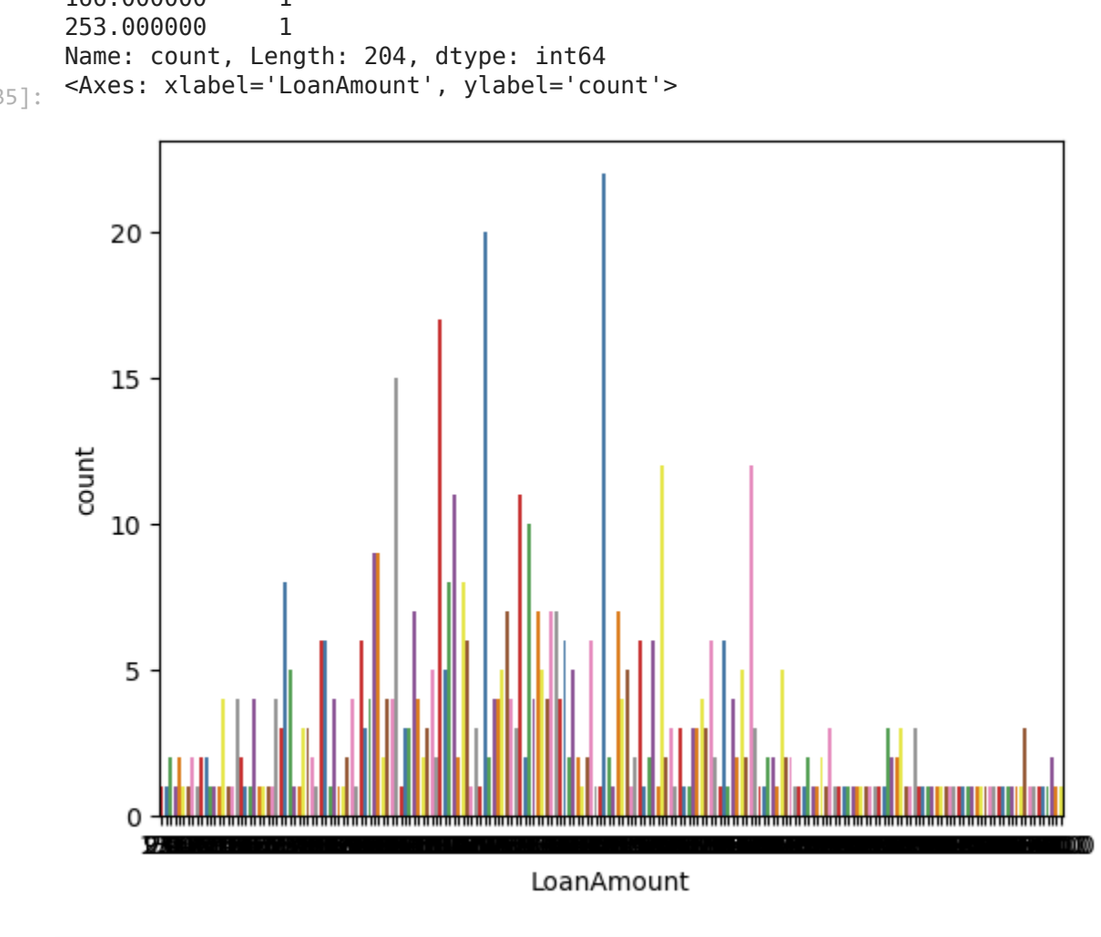
Out[34]:
```
<Axes: xlabel='Self_Employed', ylabel='count'>
```



In [35]:
```python
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:
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: count, Length: 204, dtype: int64
```

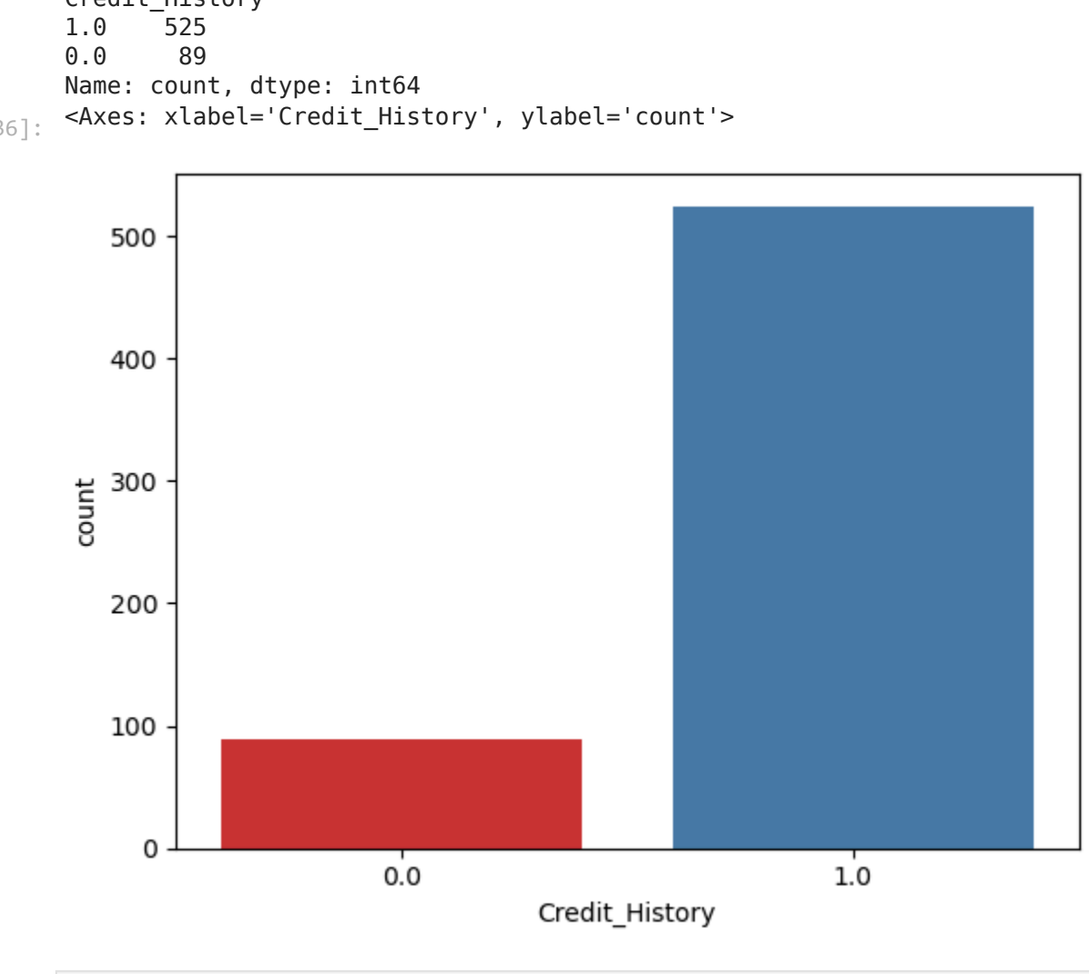
Out[35]:
```
<Axes: xlabel='LoanAmount', ylabel='count'>
```



In [36]:
```python
print("number of people who take loan as group by 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 Credit history:
Credit_History
1.0    525
0.0     89
Name: count, dtype: int64
```

Out[36]:
```
<Axes: xlabel='Credit_History', ylabel='count'>
```



In [37]:
```python
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)

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

In [46]:
```python
for i in range(0, 5):
    X_train[:,i]= Labelencoder_x.fit_transform(X_train[:,i])
    X_train[:,6]= Labelencoder_x.fit_transform(X_train[:,6])

X_train
```

Out[46]:
```
array([[1, 1, 0, ..., 7, 1.0, 81],
       [1, 0, 1, ..., 7, 1.0, 131],
       [1, 1, 0, ..., 7, 0.0, 98],
       ...,
       [1, 1, 3, ..., 7, 1.0, 132],
       [1, 1, 0, ..., 7, 1.0, 108],
       [0, 1, 0, ..., 7, 1.0, 122]], dtype=object)
```

In [47]:
```python
Labelencoder_y = LabelEncoder()
y_train = Labelencoder_y.fit_transform(y_train)

y_train
```

Out[47]:
```
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, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1,
       1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 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,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 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, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1,
       0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 1, 0, 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, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 0, 1])
```

In [49]:
```python
for i in range(0,5):
    X_test[:,i]= Labelencoder_x.fit_transform(X_test[:,i])
    X_test[:,6]= Labelencoder_x.fit_transform(X_test[:,6])

X_test
```

Out[49]:
```
array([[0, 0, 0, 0, 0, 1.0, 10],
       [0, 0, 0, 0, 0, 1.0, 24],
       [1, 1, 1, 1, 1, 1.0, 71],
       [1, 1, 1, 1, 1, 1.0, 22],
       [1, 1, 1, 1, 1, 1.0, 16],
       [1, 1, 1, 1, 1, 0.0, 48],
       [1, 1, 1, 1, 1, 1.0, 45],
       [0, 0, 0, 0, 0, 1.0, 72],
       [0, 0, 0, 0, 0, 0.0, 30],
       [1, 1, 1, 1, 1, 1.0, 33],
       [0, 0, 0, 0, 0, 1.0, 10],
       [1, 1, 1, 1, 1, 1.0, 14],
       [0, 0, 0, 0, 0, 1.0, 68],
       [1, 1, 1, 1, 1, 1.0, 65],
       [1, 1, 1, 1, 1, 1.0, 15],
       [1, 1, 1, 1, 1, 1.0, 3],
       [0, 0, 0, 0, 0, 1.0, 58],
       [0, 0, 0, 0, 0, 1.0, 36],
       [0, 0, 0, 0, 0, 1.0, 13],
       [1, 1, 1, 1, 1, 0.0, 55],
       [1, 1, 1, 1, 1, 0.0, 28],
       [1, 1, 1, 1, 1, 1.0, 43],
       [1, 1, 1, 1, 1, 1.0, 21],
       [1, 1, 1, 1, 1, 1.0, 48],
       [1, 1, 1, 1, 1, 1.0, 2],
       [1, 1, 1, 1, 1, 1.0, 66],
       [0, 0, 0, 0, 0, 1.0, 57],
       [1, 1, 1, 1, 1, 1.0, 57],
       [1, 1, 1, 1, 1, 0.0, 33],
       [0, 0, 0, 0, 0, 0.0, 18],
       [1, 1, 1, 1, 1, 1.0, 64],
       [0, 0, 0, 0, 0, 1.0, 1],
       [1, 1, 1, 1, 1, 1.0, 12],
       [1, 1, 1, 1, 1, 1.0, 54],
       [1, 1, 1, 1, 1, 1.0, 34],
       [1, 1, 1, 1, 1, 0.0, 53],
       [1, 1, 1, 1, 1, 1.0, 50],
       [0, 0, 0, 0, 0, 1.0, 32],
       [1, 1, 1, 1, 1, 1.0, 33],
       [1, 1, 1, 1, 1, 1.0, 21],
       [0, 0, 0, 0, 0, 1.0, 8],
       [1, 1, 1, 1, 1, 0.0, 29],
       [1, 1, 1, 1, 1, 1.0, 0],
       [1, 1, 1, 1, 1, 1.0, 46],
       [0, 0, 0, 0, 0, 1.0, 7],
       [0, 0, 0, 0, 0, 1.0, 57],
       [1, 1, 1, 1, 1, 1.0, 27],
       [0, 0, 0, 0, 0, 1.0, 26],
       [1, 1, 1, 1, 1, 1.0, 33],
       [0, 0, 0, 0, 0, 1.0, 42],
       [1, 1, 1, 1, 1, 1.0, 6],
       [1, 1, 1, 1, 1, 1.0, 35],
       [1, 1, 1, 1, 1, 1.0, 20],
       [0, 0, 0, 0, 0, 1.0, 34],
       [1, 1, 1, 1, 1, 1.0, 24],
       [1, 1, 1, 1, 1, 1.0, 67],
       [1, 1, 1, 1, 1, 1.0, 14],
       [0, 0, 0, 0, 0, 1.0, 37],
       [1, 1, 1, 1, 1, 1.0, 49],
       [0, 0, 0, 0, 0, 1.0, 28],
       [0, 0, 0, 0, 0, 0.0, 41],
       [0, 0, 0, 0, 0, 1.0, 63],
       [1, 1, 1, 1, 1, 1.0, 26],
       [0, 0, 0, 0, 0, 0.0, 7],
       [1, 1, 1, 1, 1, 1.0, 37],
       [1, 1, 1, 1, 1, 1.0, 69],
       [1, 1, 1, 1, 1, 1.0, 47],
       [0, 0, 0, 0, 0, 1.0, 40],
       [1, 1, 1, 1, 1, 1.0, 56],
       [0, 0, 0, 0, 0, 1.0, 15],
       [0, 0, 0, 0, 0, 1.0, 3],
       [1, 1, 1, 1, 1, 1.0, 35],
       [1, 1, 1, 1, 1, 1.0, 45],
       [1, 1, 1, 1, 1, 1.0, 41],
       [1, 1, 1, 1, 1, 1.0, 44],
       [1, 1, 1, 1, 1, 1.0, 30],
       [0, 0, 0, 0, 0, 1.0, 12],
       [0, 0, 0, 0, 0, 1.0, 70],
       [1, 1, 1, 1, 1, 1.0, 24],
       [0, 0, 0, 0, 0, 0.0, 27],
       [0, 0, 0, 0, 0, 1.0, 25],
       [1, 1, 1, 1, 1, 1.0, 74],
       [0, 0, 0, 0, 0, 1.0, 59],
       [1, 1, 1, 1, 1, 1.0, 57],
       [0, 0, 0, 0, 0, 1.0, 5],
       [1, 1, 1, 1, 1, 1.0, 23],
       [1, 1, 1, 1, 1, 1.0, 22],
       [1, 1, 1, 1, 1, 1.0, 60],
       [0, 0, 0, 0, 0, 1.0, 25],
       [1, 1, 1, 1, 1, 1.0, 19],
       [1, 1, 1, 1, 1, 1.0, 17],
       [0, 0, 0, 0, 0, 1.0, 6],
       [1, 1, 1, 1, 1, 1.0, 50],
       [1, 1, 1, 1, 1, 0.0, 54],
       [0, 0, 0, 0, 0, 1.0, 5],
       [0, 0, 0, 0, 0, 0.0, 32],
       [0, 0, 0, 0, 0, 1.0, 51],
       [0, 0, 0, 0, 0, 1.0, 44],
       [1, 1, 1, 1, 1, 1.0, 9],
       [1, 1, 1, 1, 1, 1.0, 57],
       [1, 1, 1, 1, 1, 1.0, 62],
       [1, 1, 1, 1, 1, 1.0, 17],
       [1, 1, 1, 1, 1, 1.0, 28],
       [0, 0, 0, 0, 0, 1.0, 28],
       [1, 1, 1, 1, 1, 1.0, 33],
       [0, 0, 0, 0, 0, 1.0, 31],
       [0, 0, 0, 0, 0, 1.0, 19],
       [0, 0, 0, 0, 0, 1.0, 11],
       [1, 1, 1, 1, 1, 1.0, 14],
       [1, 1, 1, 1, 1, 1.0, 9],
       [0, 0, 0, 0, 0, 1.0, 60],
       [0, 0, 0, 0, 0, 1.0, 39],
       [0, 0, 0, 0, 0, 1.0, 25],
       [1, 1, 1, 1, 1, 1.0, 4],
       [1, 1, 1, 1, 1, 0.0, 61],
       [1, 1, 1, 1, 1, 0.0, 12],
       [0, 0, 0, 0, 0, 1.0, 10],
       [0, 0, 0, 0, 0, 1.0, 38],
       [1, 1, 1, 1, 1, 1.0, 52],
       [1, 1, 1, 1, 1, 1.0, 35],
       [1, 1, 1, 1, 1, 1.0, 73],
       [1, 1, 1, 1, 1, 0.0, 4],
       [1, 1, 1, 1, 1, 1.0, 15]], dtype=object)
```

In [50]:
```python
Labelencoder_y = LabelEncoder()

y_test = Labelencoder_y.fit_transform(y_test)

y_test
```

Out[50]:
```
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, 0, 1, 0, 1, 1,
       1, 1, 1, 1, 0, 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, 0, 1, 1, 1, 1, 1, 0,
       1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1])
```

In [51]:
```python
from sklearn.preprocessing import StandardScaler

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

In [52]:
```python
from sklearn.ensemble import RandomForestClassifier

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

Out[52]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [83]:
```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 0.6341463414634146
```

Out[83]:
```
array([1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0,
       1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1])
```

In [84]:
```python
from sklearn.naive_bayes import GaussianNB
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
```

Out[84]:
```
▾ GaussianNB
GaussianNB()
```

In [85]:
```python
y_pred = nb_classifier.predict(X_test)
print("acc of gaussianNB is %.", metrics.accuracy_score(y_pred, y_test) )
```

```
acc of gaussianNB is %. 0.2764227642276423
```

In [86]:
```python
y_pred
```

Out[86]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0])
```

In [93]:
```python
from sklearn.tree import DecisionTreeClassifier
df_clf = DecisionTreeClassifier()
df_clf.fit(X_train, y_train)
```

Out[93]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [99]:
```python
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics

dt_clf = DecisionTreeClassifier(random_state=42)
dt_clf.fit(X_train, y_train)

y_pred = dt_clf.predict(X_test)
print("acc of DT is", metrics.accuracy_score(y_pred, y_test))
```

```
acc of DT is 0.37398373983739835
```

In [97]:
```python
y_pred
```

Out[97]:
```
array([1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0,
       1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0])
```

In [100…
```python
from sklearn.neighbors import KNeighborsClassifier
kn_clf = KNeighborsClassifier()
kn_clf.fit(X_train, y_train)
```

Out[100]:
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [101…
```python
y_pred = kn_clf.predict(X_test)
print("acc of KN is", metrics.accuracy_score(y_pred, y_test))
```

```
acc of KN is 0.7317073170731707
```

In [ ]: