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_excel('C:/Users/vasam/Downloads/loan.xlsx')
df.head()
```

Out[2]:

	Loan ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Co
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	

In [3]:

```
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
```

Null Values

In [4]:

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

Out[4]:

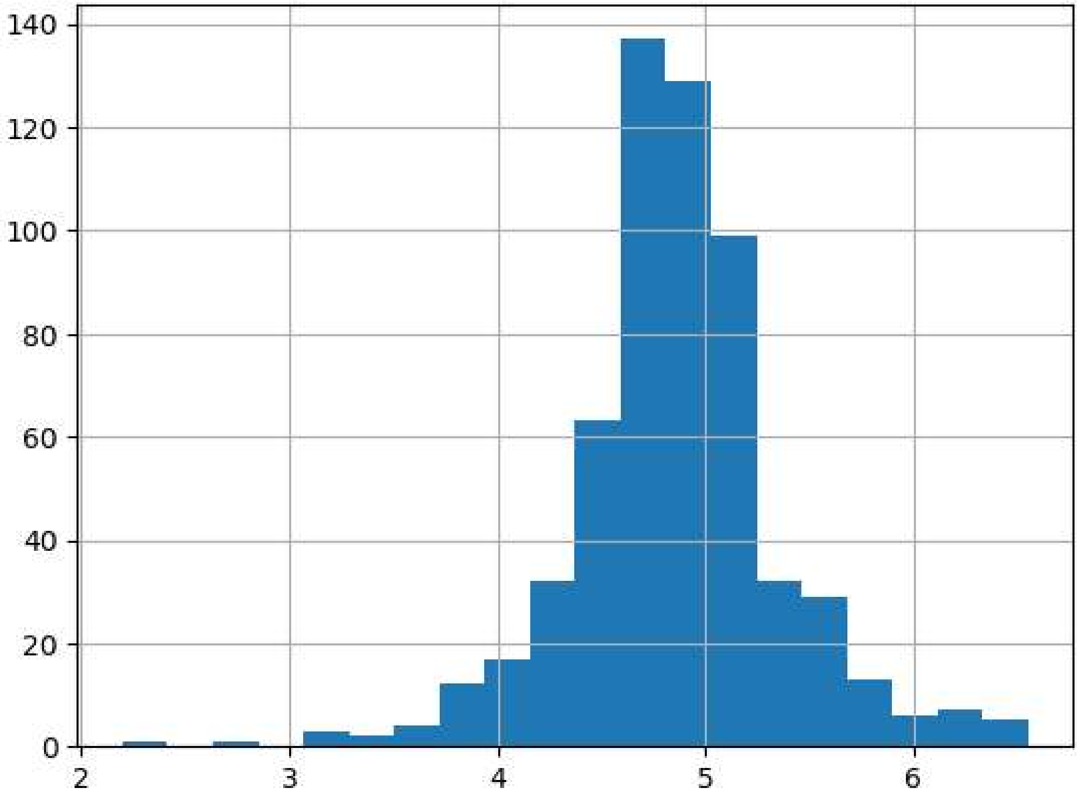
```
 Loan ID              0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [5]:

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

Out[5]:

```
<AxesSubplot:>
```

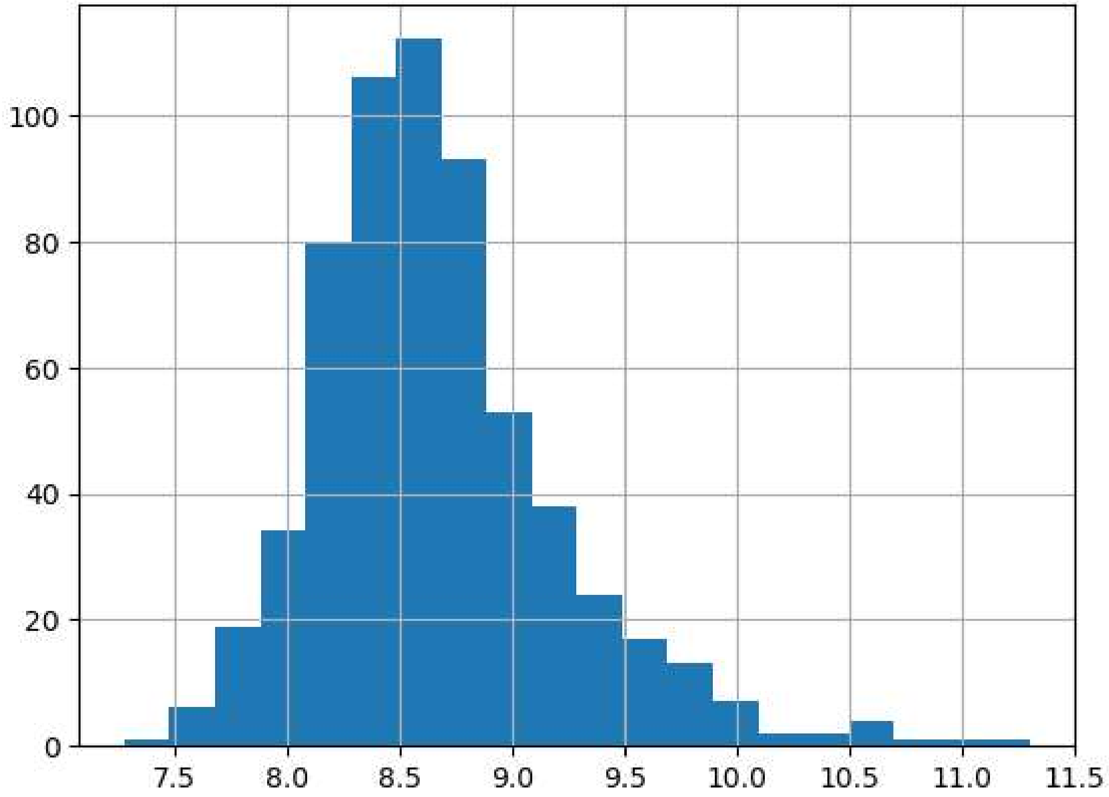


In [6]:

```
df['TotalIncome']=df['ApplicantIncome']+df['CoapplicantIncome']
df['TotalIncome']=np.log(df['TotalIncome'])
df['TotalIncome'].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
TotalIncome           0
dtype: int64
```

Replacing Null values By Mode and Mean

In [8]:

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

In [9]:

```
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
dtype: int64
```

In [10]:

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

In [11]:

```
x
```

Out[11]:

```
array([['Male', 'No', 0, ..., 1.0, 4.857444178729353, 8.674025985443025],
       ['Male', 'Yes', 1, ..., 1.0, 4.852030263919617, 8.71456755083648
5],
       ['Male', 'Yes', 0, ..., 1.0, 4.189654742026425, 8.00636756765024
6],
       ...,
       ['Male', 'Yes', 1, ..., 1.0, 5.53338948872752, 9.025455532779063],
       ['Male', 'Yes', 2, ..., 1.0, 5.231108616854587, 8.93366417870093
5],
       ['Female', 'No', 0, ..., 0.0, 4.890349128221754,
        8.430109084509125]], dtype=object)
```

In [12]:

```
y
```

Out[12]:

```
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)
```

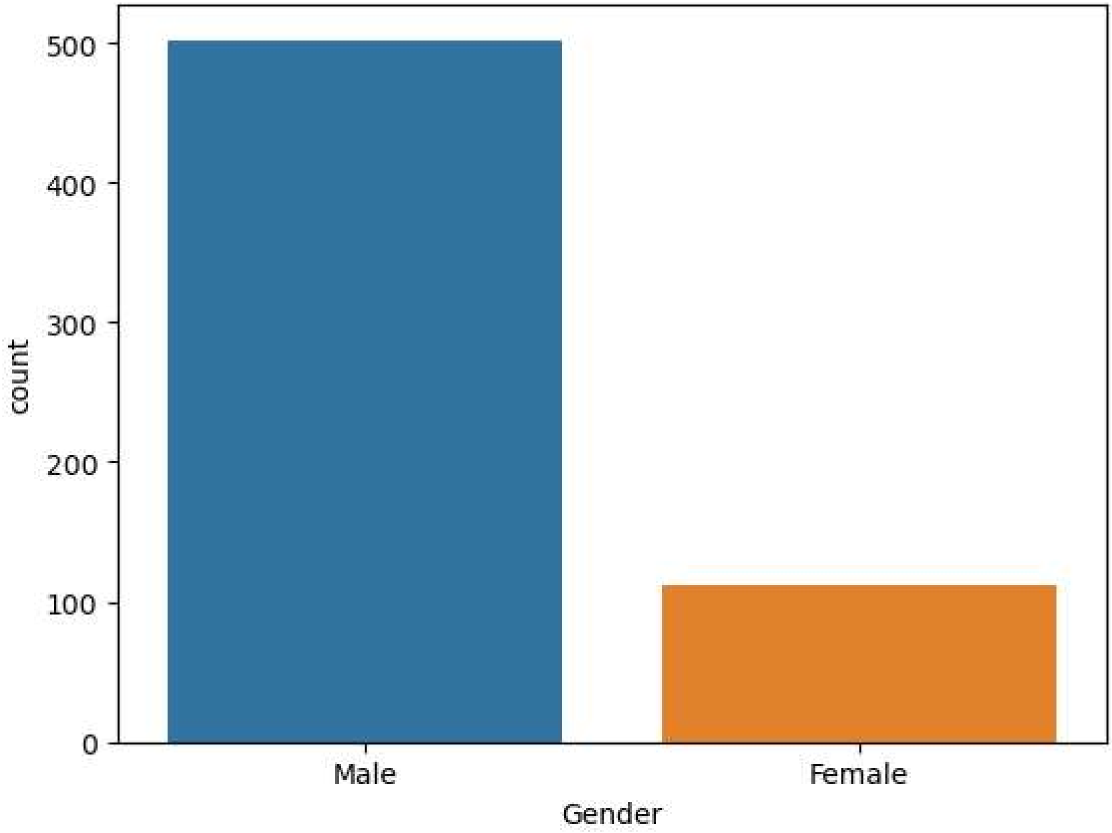
In [13]:

```
print('Number of people who takes loan as group by gender')
print(df['Gender'].value_counts())
sns.countplot(x='Gender',data=df)
```

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

Out[13]:

```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



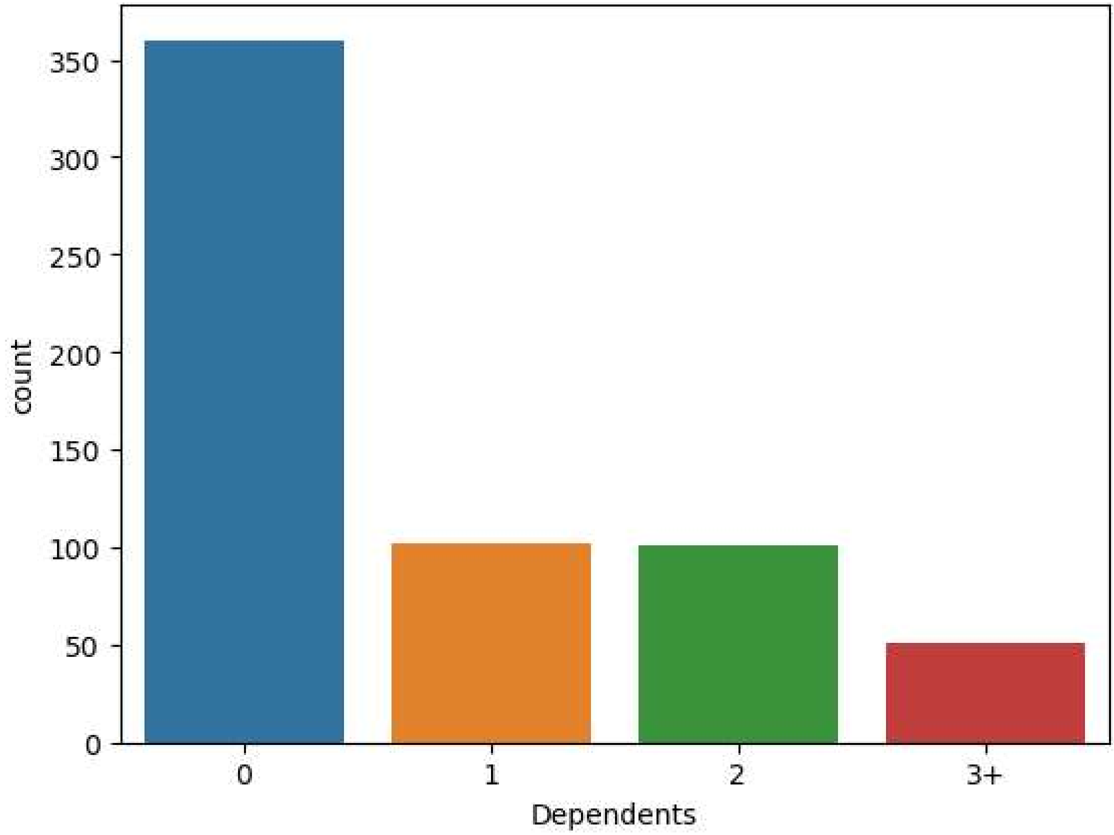
In [14]:

```
print('Number of people who takes loan as group by Dependents')
print(df['Dependents'].value_counts())
sns.countplot(x='Dependents',data=df)
```

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

Out[14]:

```
<AxesSubplot:xlabel='Dependents', ylabel='count'>
```



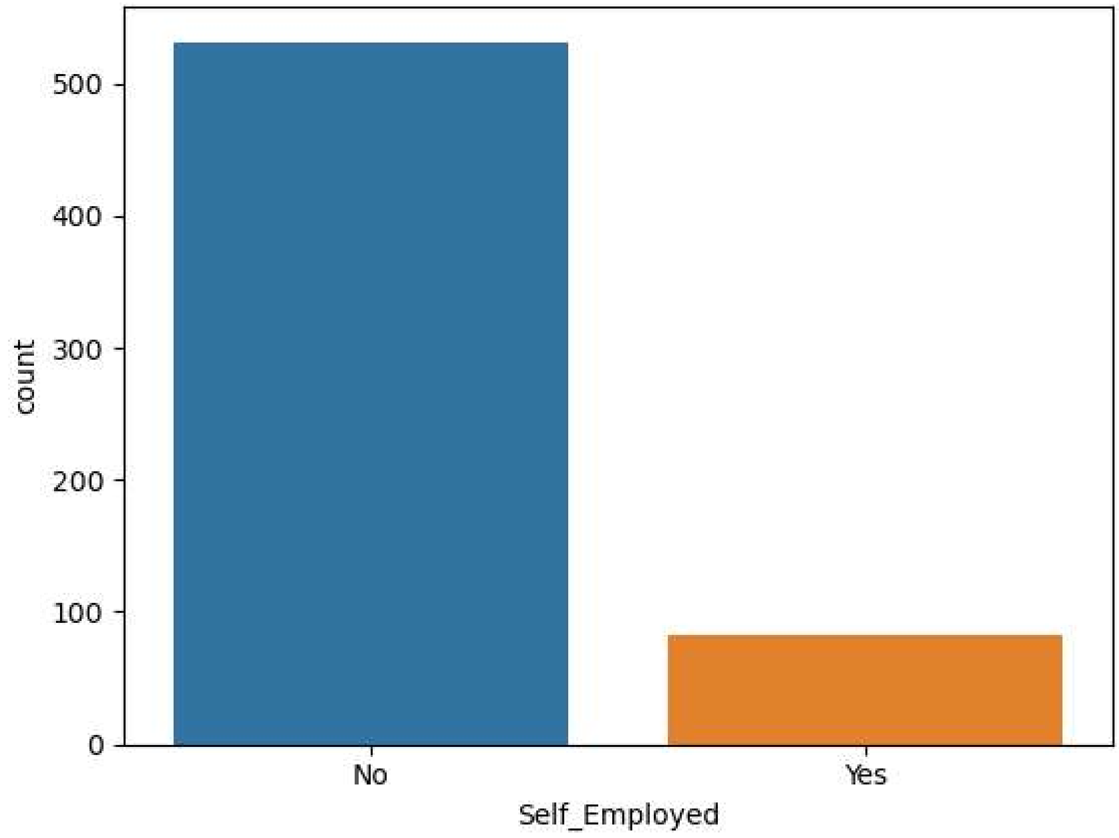
In [15]:

```
print('Number of people who takes loan as group by self employed')
print(df['Self_Employed'].value_counts())
sns.countplot(x='Self_Employed',data=df)
```

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

Out[15]:

```
<AxesSubplot:xlabel='Self_Employed', ylabel='count'>
```



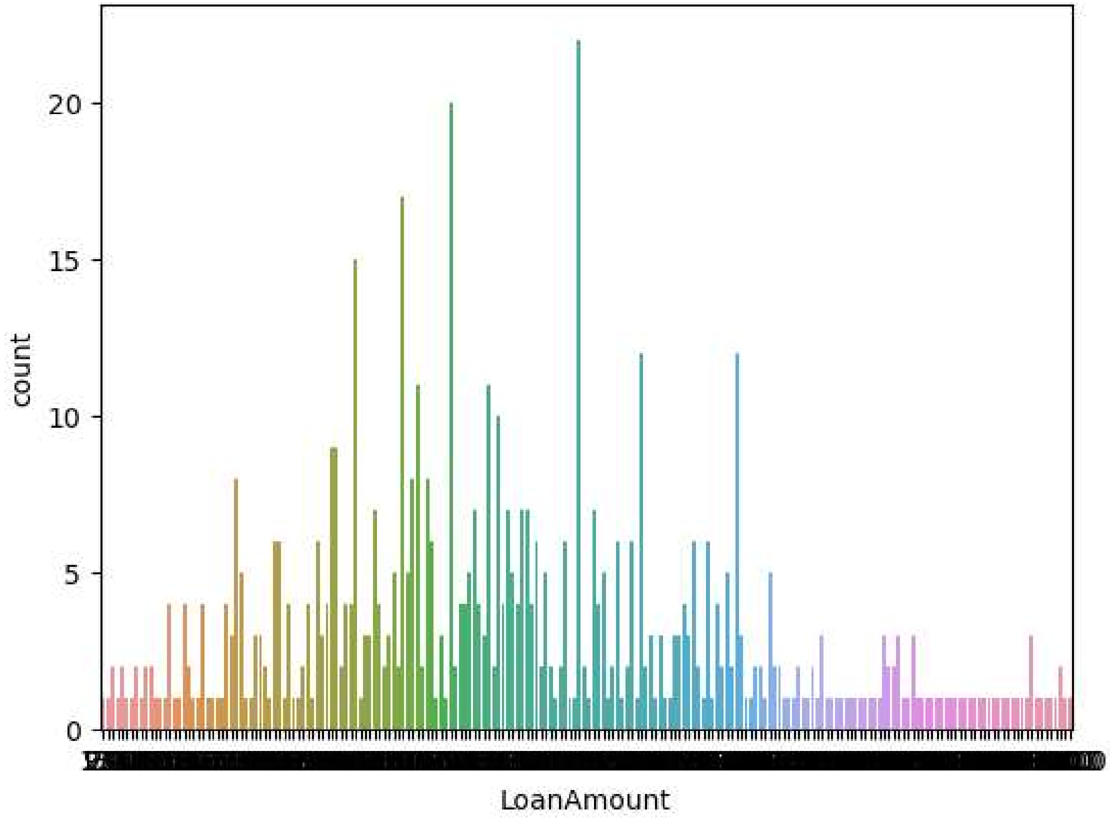
In [16]:

```
print('Number of people who takes loan as group by LoanAmount')
print(df['LoanAmount'].value_counts())
sns.countplot(x='LoanAmount',data=df)
```

```
Number of people who takes 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[16]:

```
<AxesSubplot:xlabel='LoanAmount', ylabel='count'>
```



Splitting dataset into Training and Testing

In [17]:

```
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)
```

Preprocessing Data

In [18]:

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

In [23]:

```
for i in range(0,5):
    x_train[:,i] = labelencoder.fit_transform(x_train[:,i].astype(float))
    x_train[:,7] = labelencoder.fit_transform(x_train[:,7].astype(float))
```

In [24]:

```
x_train
```

Out[24]:

```
array([[1, 1, 0, ..., 1.0, 4.875197323201151, 331],
       [1, 0, 1, ..., 1.0, 5.278114659230517, 8],
       [1, 1, 0, ..., 0.0, 5.003946305945459, 303],
       ...,
       [1, 1, 3, ..., 1.0, 5.298317366548036, 312],
       [1, 1, 0, ..., 1.0, 5.075173815233827, 387],
       [0, 1, 0, ..., 1.0, 5.204006687076795, 338]], dtype=object)
```

In [26]:

```
for i in range(0,5):
    y_train = labelencoder.fit_transform(y_train)
y_train
```

Out[26]:

```
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], dtype=int64)
```

In [32]:

```
for i in range(0,5):
    x_test[:,i]=labelencoder.fit_transform(x_test[:,i].astype(int))
    x_test[:,7]=labelencoder.fit_transform(x_test[:,7].astype(int))
x_test
```

```
       [0, 0, 0, 0, 4, 0.0, 4.634728988229636, 71],
       [1, 1, 0, 0, 4, 1.0, 5.429345628954441, 42],
       [1, 0, 0, 1, 4, 1.0, 3.871201010907891, 22],
       [1, 1, 1, 1, 4, 1.0, 4.499809670330265, 102],
       [1, 1, 0, 0, 4, 1.0, 5.19295685089021, 26],
       [1, 1, 0, 0, 4, 1.0, 4.857444178729353, 96],
       [0, 1, 0, 1, 4, 0.0, 5.181783550292085, 3],
       [1, 1, 0, 0, 4, 1.0, 5.147494476813453, 67],
       [1, 0, 0, 1, 4, 1.0, 4.836281906951478, 94],
       [1, 1, 0, 0, 4, 1.0, 4.852030263919617, 55],
       [1, 1, 2, 1, 4, 1.0, 4.68213122712422, 70],
       [0, 0, 0, 0, 4, 1.0, 4.382026634673881, 31],
       [1, 1, 3, 0, 4, 0.0, 4.812184355372417, 93],
       [1, 1, 2, 0, 1, 1.0, 2.833213344056216, 0],
       [1, 1, 1, 1, 4, 1.0, 5.062595033026967, 90],
       [1, 0, 0, 0, 4, 1.0, 4.330733340286331, 39],
       [1, 0, 0, 0, 4, 1.0, 5.231108616854587, 52],
       [1, 1, 1, 0, 4, 1.0, 4.753590191063645, 97],
       [0, 0, 0, 0, 4, 1.0, 4.74493212836325, 82],
       [1, 1, 1, 0, 4, 1.0, 4.852030263919617, 20],
```

In [33]:

```
y_test = labelencoder.fit_transform(y_test)
y_test
```

Out[33]:

```
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])
```

In [34]:

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

Random Forest Classifier

In [35]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[35]:

```
RandomForestClassifier()
```

In [36]:

```
from sklearn.metrics import accuracy_score
y_pred = rfc.predict(x_test)
print('accuracy of using random forest classifier is ',accuracy_score(y_pred,y_test))
```

```
accuracy of using random forest classifier is  0.7642276422764228
```

Naive Bayes Classifier

In [37]:

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

Out[37]:

```
GaussianNB()
```

In [38]:

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

In [39]:

```
print('accuracy of using naive bayes is ',accuracy_score(y_pred,y_test))
```

```
accuracy of using naive bayes is  0.8292682926829268
```

Decision Tree Classifier

In [40]:

```
from sklearn.tree import DecisionTreeClassifier
dfc = DecisionTreeClassifier()
dfc.fit(x_train,y_train)
```

Out[40]:

```
DecisionTreeClassifier()
```

In [41]:

```
y_pred=dfc.predict(x_test)
print('accuracy of using decision tree classifier is',accuracy_score(y_pred,y_test))
```

```
accuracy of using decision tree classifier is 0.7073170731707317
```

KNN Classifier

In [44]:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(x_train,y_train)
```

Out[44]:

```
KNeighborsClassifier()
```

In [45]:

```
y_pred = knn.predict(x_test)
print('accuracy of using knn classifier is ',accuracy_score(y_pred,y_test))
```

```
accuracy of using knn classifier is  0.7804878048780488
```

```
C:\Users\vasam\anaconda3\lib\site-packages\sklearn\neighbors\_classificat
ion.py:228: FutureWarning: Unlike other reduction functions (e.g. `skew`,
`kurtosis`), the default behavior of `mode` typically preserves the axis
it acts along. In SciPy 1.11.0, this behavior will change: the default va
lue of `keepdims` will become False, the `axis` over which the statistic
is taken will be eliminated, and the value None will no longer be accepte
d. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

Support Vector Machine

In [46]:

```
from sklearn.svm import SVC
svm = SVC()
svm.fit(x_train,y_train)
```

Out[46]:

```
SVC()
```

In [48]:

```
y_pred = svm.predict(x_test)
print('accuracy of using svm is ',accuracy_score(y_pred,y_test))
```

```
accuracy of using svm is  0.8292682926829268
```

Hence, Naive Bayes Classifier and Support Vector Machine is best suited algorithms for this model