Importing the libraries

In [21]:

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import psycopg2 as ps
```

Reading the data

In [22]:

```python
df = pd.read_csv("C:\\Users\\katar\\Desktop\\Machine Learning\\Loan Prediction Analysis.csv")
df.head()
```

Out[22]:

	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

Connecting to the database

```python
connection=ps.connect(host="localhost",database="Python",user="postgres",password="0000",port=5432)
```

In [12]:

```python
cursor=connection.cursor()
cursor.execute("DROP TABLE IF EXISTS Loan")
cursor.execute("CREATE TABLE Loan (Loan_ID text,Gender text,Married text,Dependents text,Education text,Self_Employed text,ApplicantIncome numeric,CoapplicantIncome numeric,LoanAmount text,Loan_Amount_Term numeric, Credit_History numeric,Property_Area text,Loa
connection.commit()
```

In [13]:

```python
for i in df.index:
    vals=[df.at[i,col] for col in list(df.columns)]
    query= "insert into Loan values ('%s','%s','%s','%s','%s','%s',%s,%s,%s,%s,%s,'%s','%s')" %(vals[0],vals[1],vals[2],vals[3],vals[4],vals[5],vals[6],vals[7],vals[8],vals[9],vals[10],vals[11],vals[12])
    cursor.execute(query)
```

In [14]:

```python
cursor.execute("select * from Loan")
connection.commit()
cursor.fetchall()
```

```
  'Male',
  'No',
  '0',
  'Graduate',
  'No',
  Decimal('5849'),
  Decimal('0.0'),
  '146.41216216216216',
  Decimal('360.0'),
  Decimal('1.0'),
  'Urban',
  'Y'),
 ('LP001003',
  'Male',
  'Yes',
  '1',
  'Graduate',
  'No',
  Decimal('4583'),
  Decimal('1508.0'),
```

In [3]:

```python
df.describe()
```

Out[3]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

In [4]:

```python
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]:

```python
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
```

Filling the null values

In [23]:

```python
df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount'].mean())
df['Loan_Amount_Term'] = df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mean())
df['Credit_History'] = df['Credit_History'].fillna(df['Credit_History'].mean())
```

In [24]:

```python
df['Gender'] = df["Gender"].fillna(df['Gender'].mode()[0])
df['Married'] = df["Married"].fillna(df['Married'].mode()[0])
df['Dependents'] = df["Dependents"].fillna(df['Dependents'].mode()[0])
df['Self_Employed'] = df["Self_Employed"].fillna(df['Self_Employed'].mode()[0])
```

In [8]:

```python
df
```

Out[8]:

	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	146.412162	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.000000	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.000000	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.000000	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.000000	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.000000	360.0	1.0	Rural	Y
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.000000	180.0	1.0	Rural	Y
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.000000	360.0	1.0	Urban	Y
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.000000	360.0	1.0	Urban	Y
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.000000	360.0	0.0	Semiurban	N

614 rows × 13 columns

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
dtype: int64
```

Exploratory Data Analysis

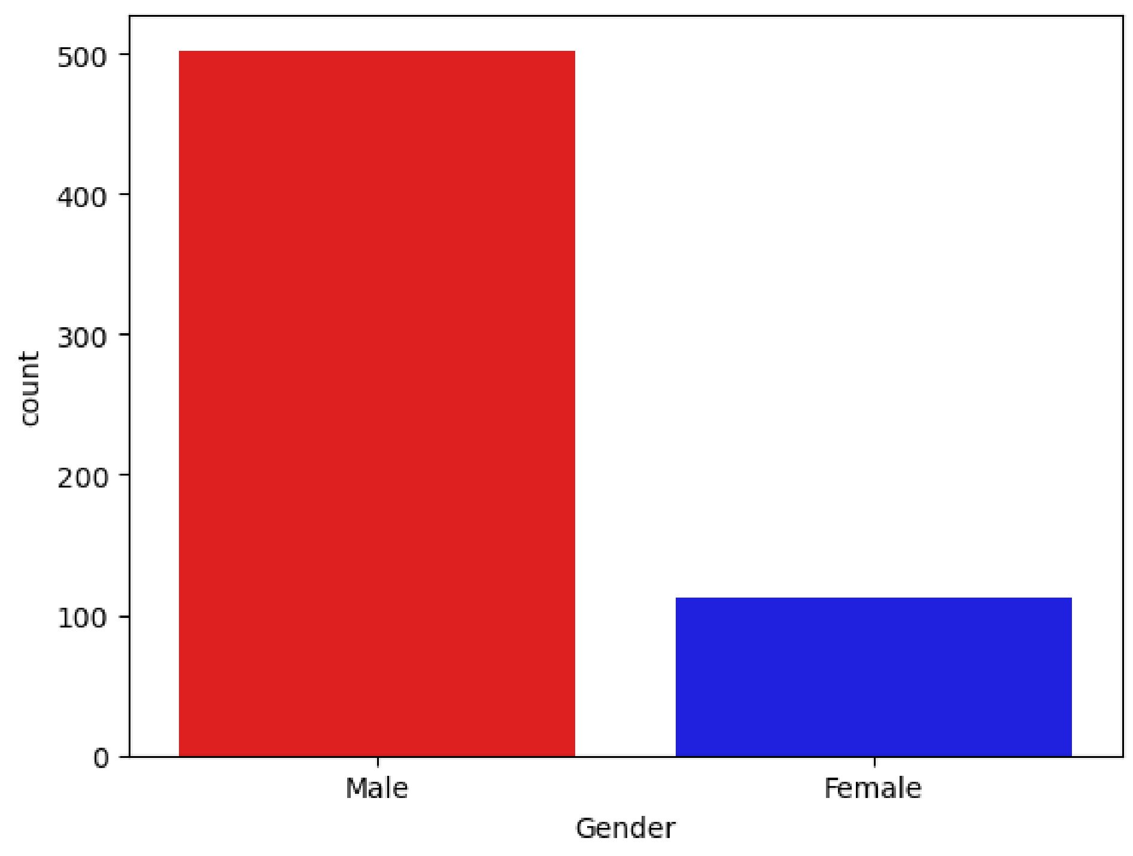
In [9]:

```
sns.countplot(df['Gender'],palette=["red","blue"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword wi
ll result in an error or misinterpretation.
  warnings.warn(
```

Out[9]:

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



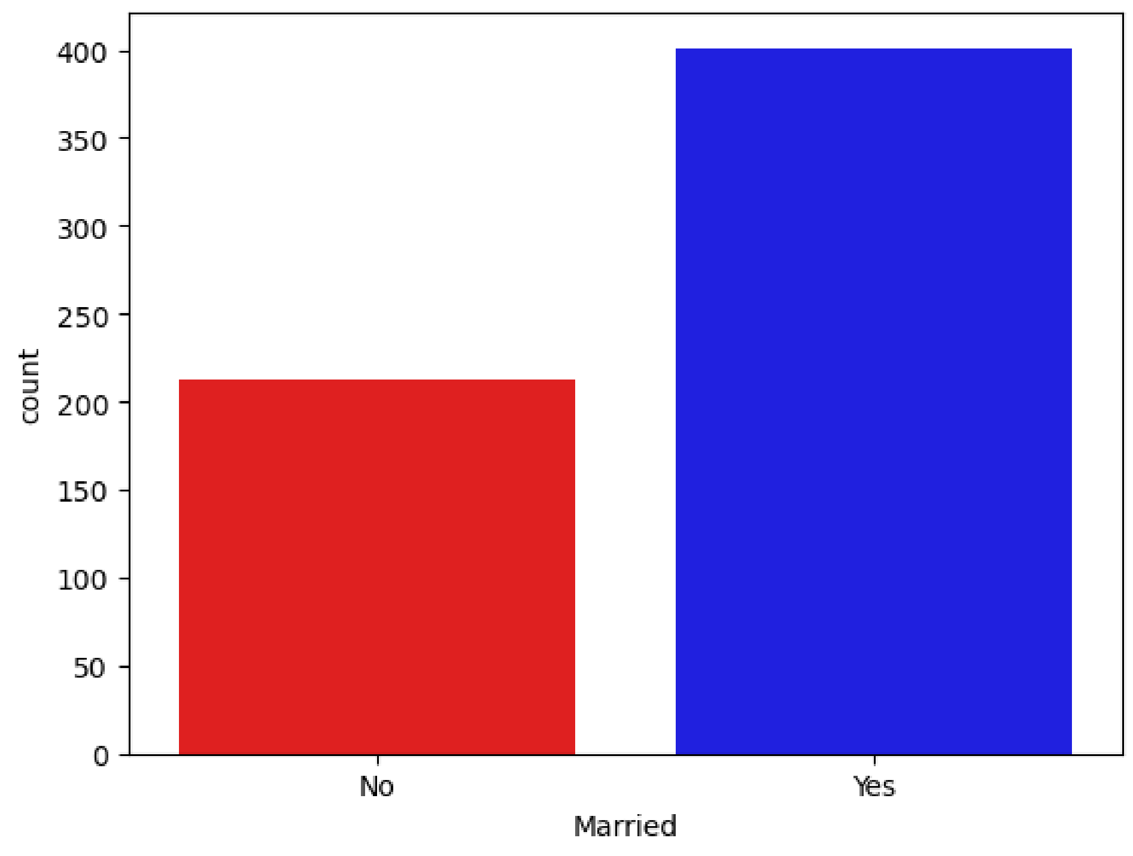
In [10]:

```
sns.countplot(df['Married'],palette=["red","blue"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword wi
ll result in an error or misinterpretation.
  warnings.warn(
```

Out[10]:

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



In [11]:

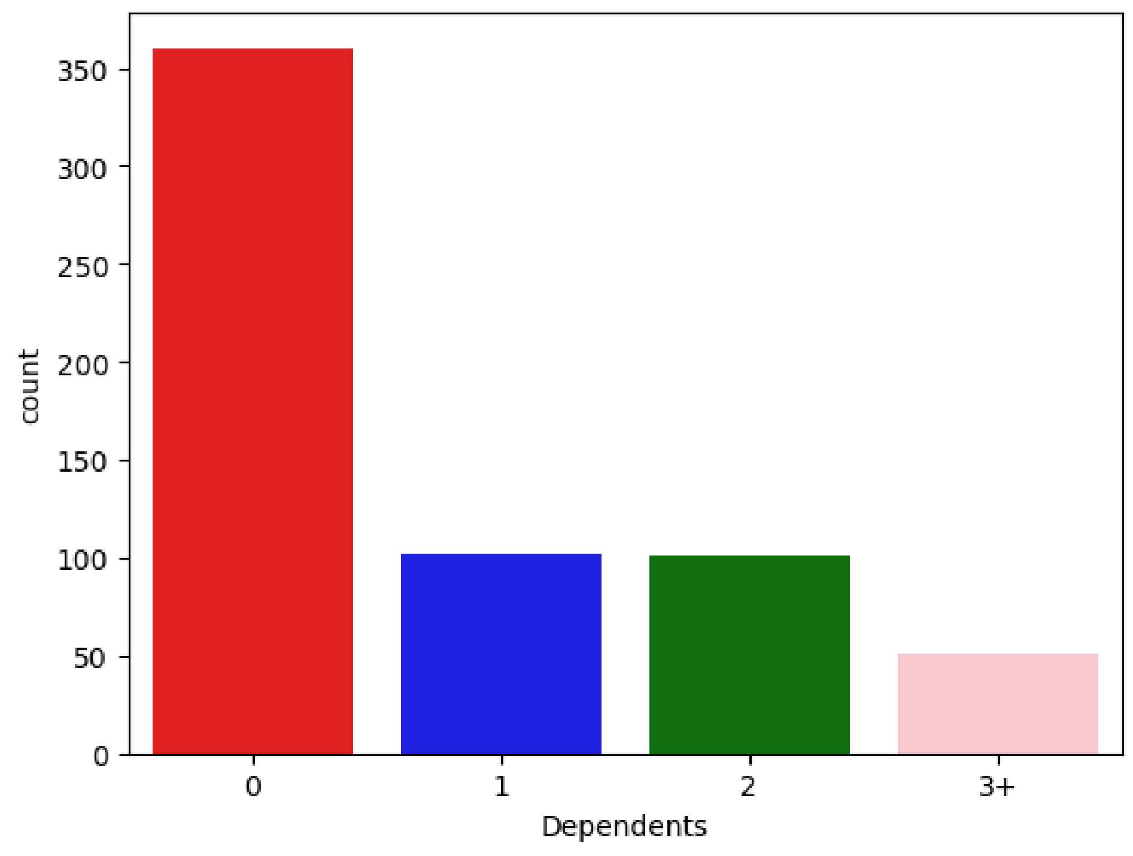
```
sns.countplot(df['Dependents'],palette=["red","blue","green","pink"])

```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword wi
ll result in an error or misinterpretation.
  warnings.warn(
```

Out[11]:

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



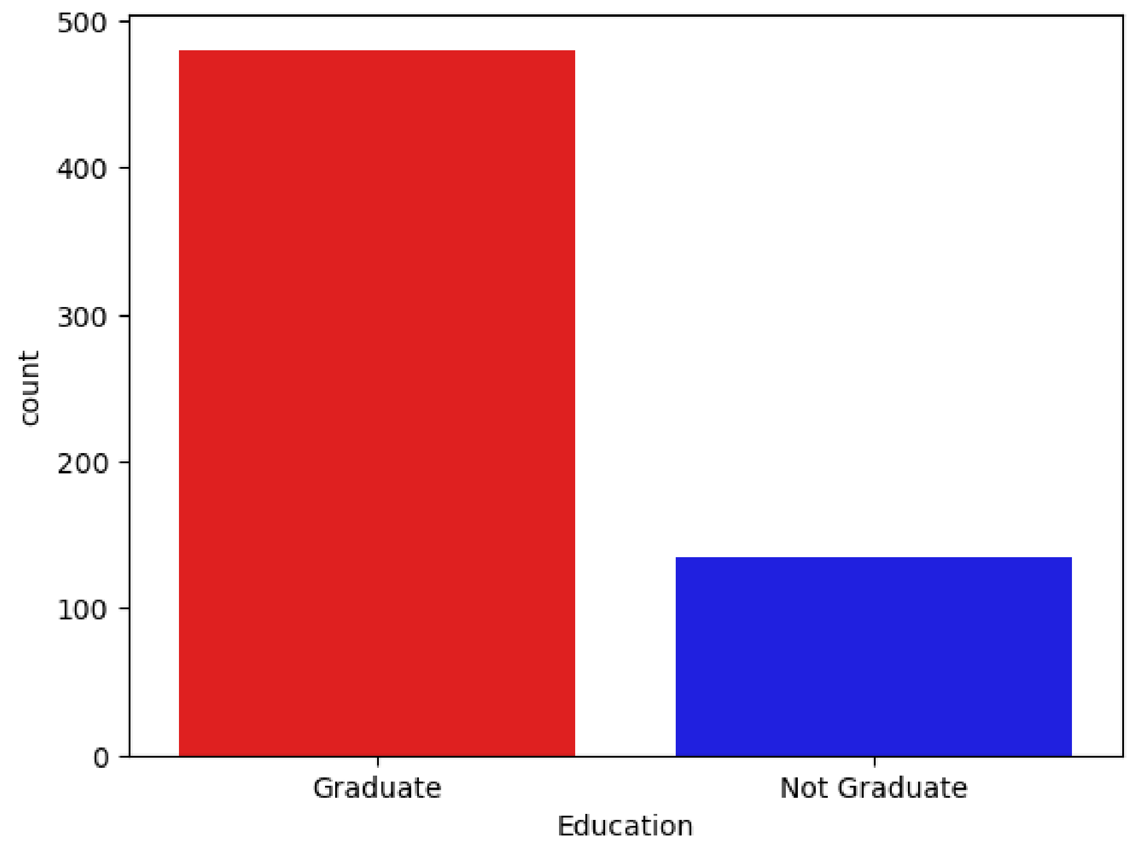
In [12]:

```
sns.countplot(df['Education'],palette=["red","blue"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword wi
ll result in an error or misinterpretation.
  warnings.warn(
```

Out[12]:

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



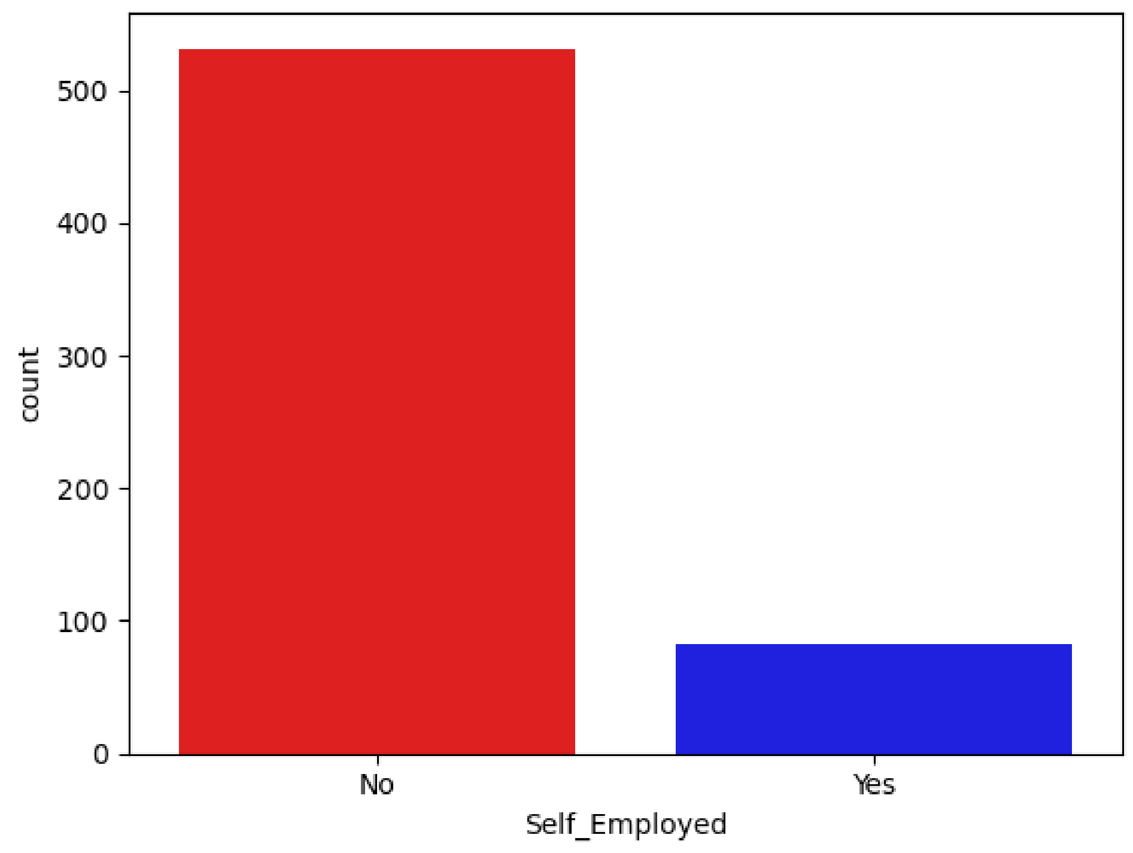
In [13]:

```
sns.countplot(df['Self_Employed'],palette=["red","blue"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword wi
ll result in an error or misinterpretation.
  warnings.warn(
```

Out[13]:

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



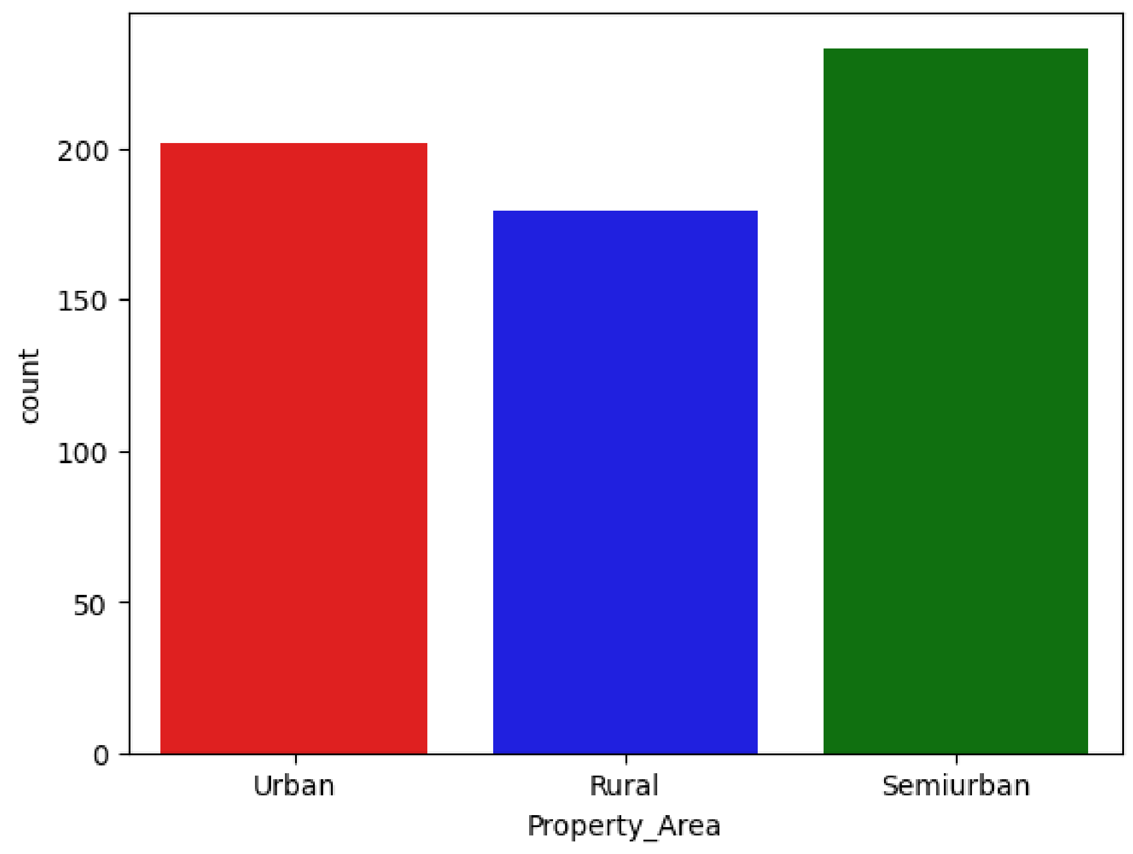
In [14]:

```
sns.countplot(df['Property_Area'],palette=["red","blue","green"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword wi
ll result in an error or misinterpretation.
  warnings.warn(
```

Out[14]:

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



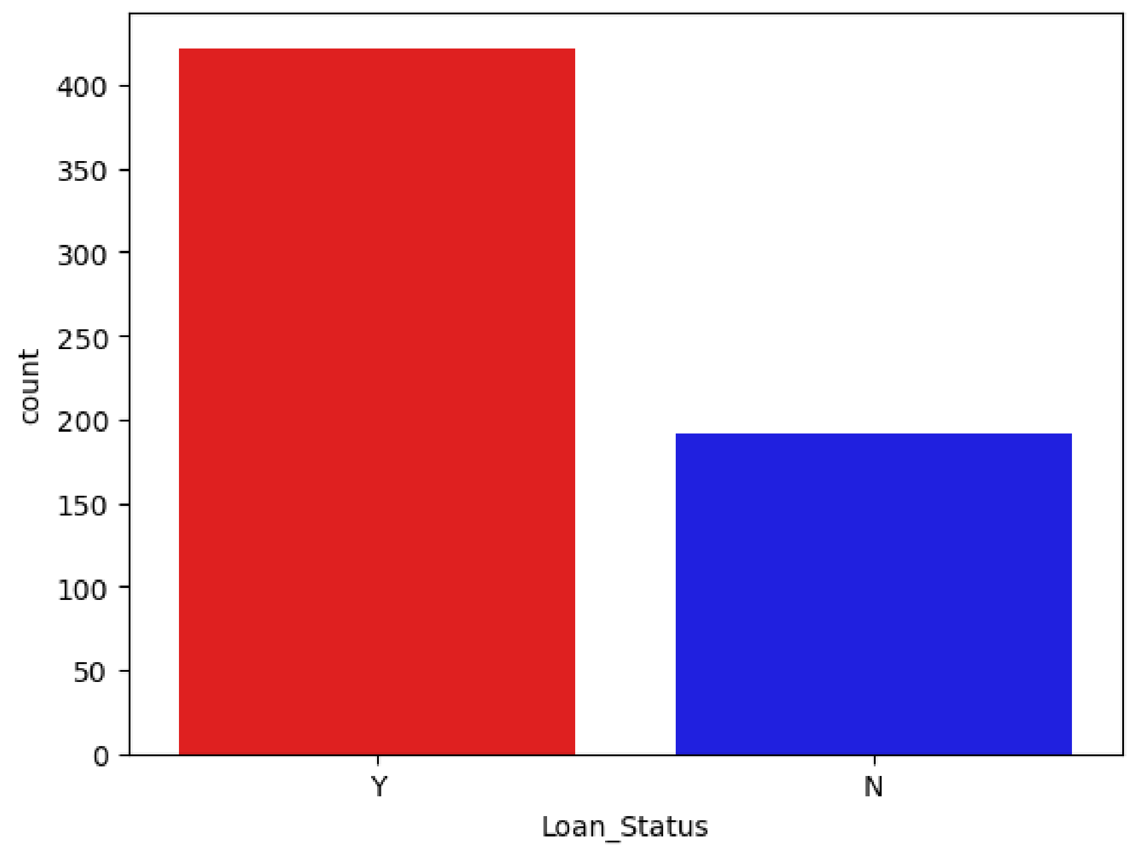
In [15]:

```
sns.countplot(df['Loan_Status'],palette=["red","blue"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword wi
ll result in an error or misinterpretation.
  warnings.warn(
```

Out[15]:

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



In [25]:

```
df['Total_Income'] = df['ApplicantIncome'] + df['CoapplicantIncome']
df.head()
```

Out[25]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status	Total_Income
0	LP001002	Male	No	0	Graduate	No	5849	0.0	146.412162	360.0	1.0	Urban	Y	5849.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.000000	360.0	1.0	Rural	N	6091.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.000000	360.0	1.0	Urban	Y	3000.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.000000	360.0	1.0	Urban	Y	4941.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.000000	360.0	1.0	Urban	Y	6000.0

Log Transformation

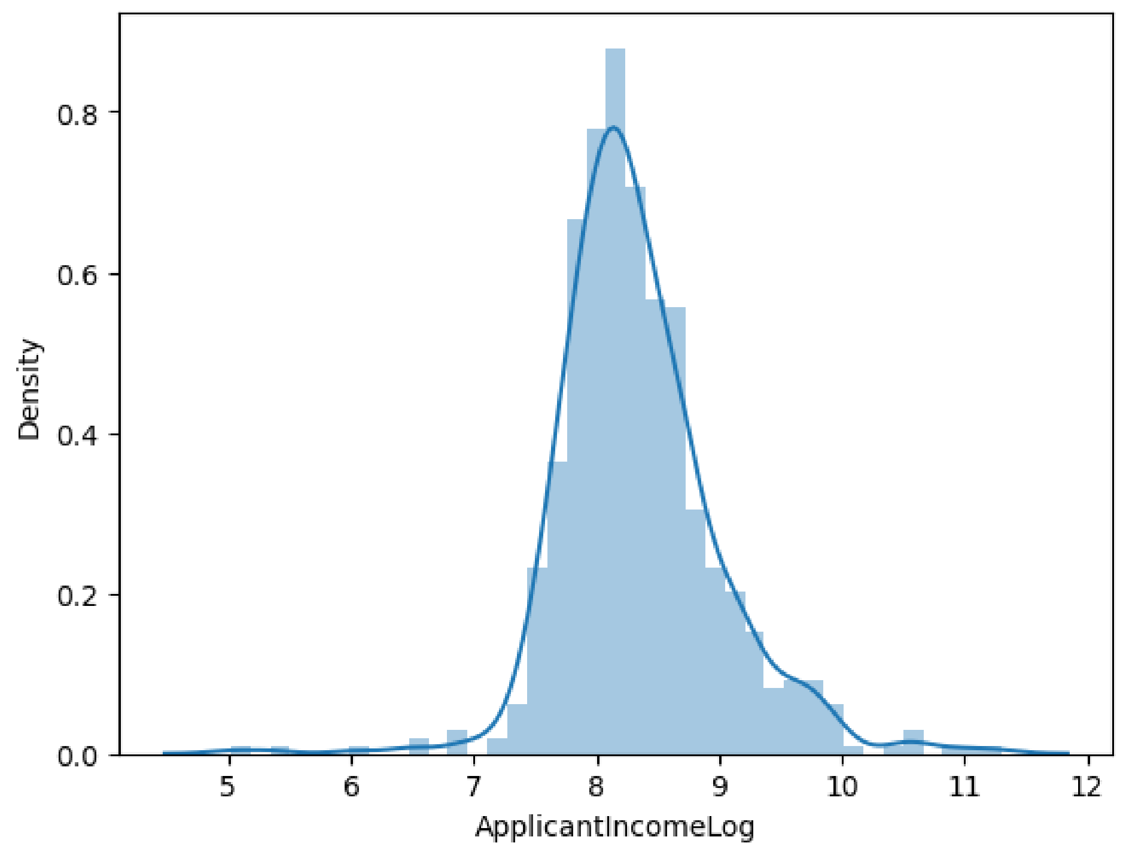
In [26]:

```
df['ApplicantIncomeLog'] = np.log(df['ApplicantIncome']+1)
sns.distplot(df["ApplicantIncomeLog"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibilit
y) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[26]:

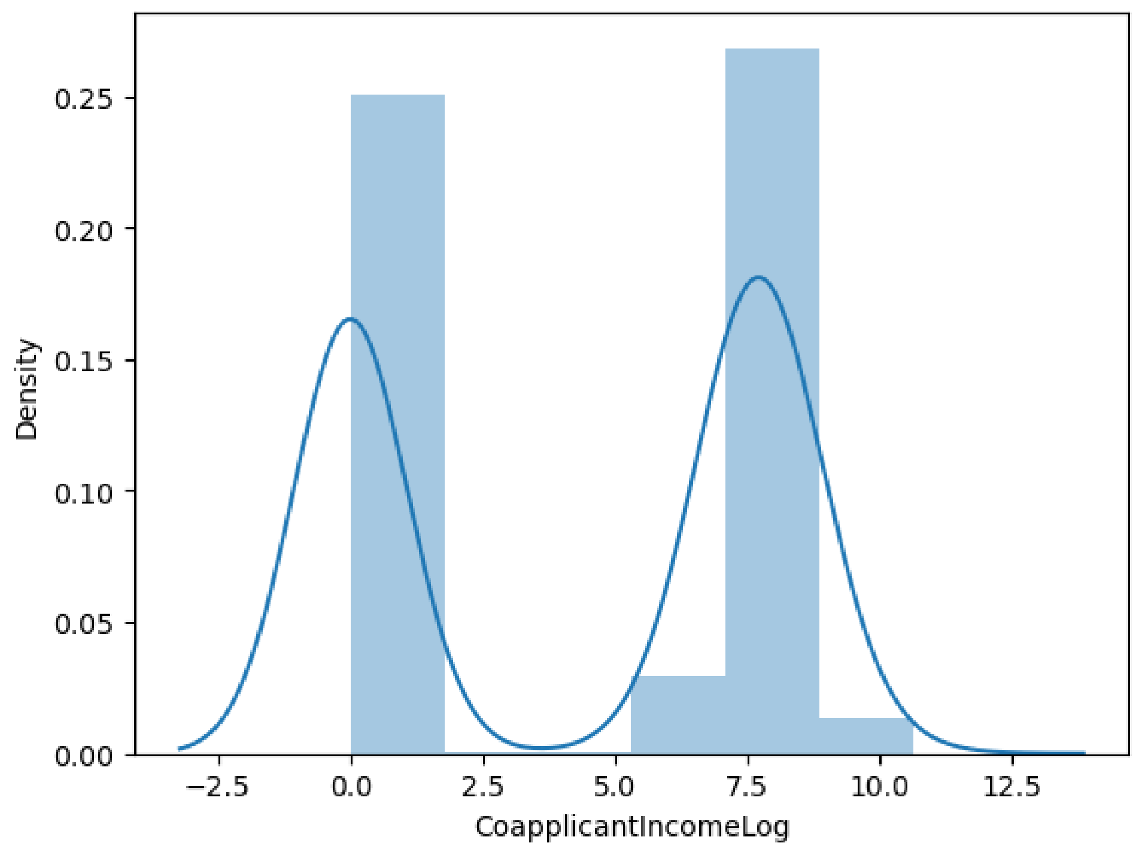
```
<AxesSubplot:xlabel='ApplicantIncomeLog', ylabel='Density'>
```



In [27]:
```
df['CoapplicantIncomeLog'] = np.log(df['CoapplicantIncome']+1)
sns.distplot(df["CoapplicantIncomeLog"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibilit
y) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

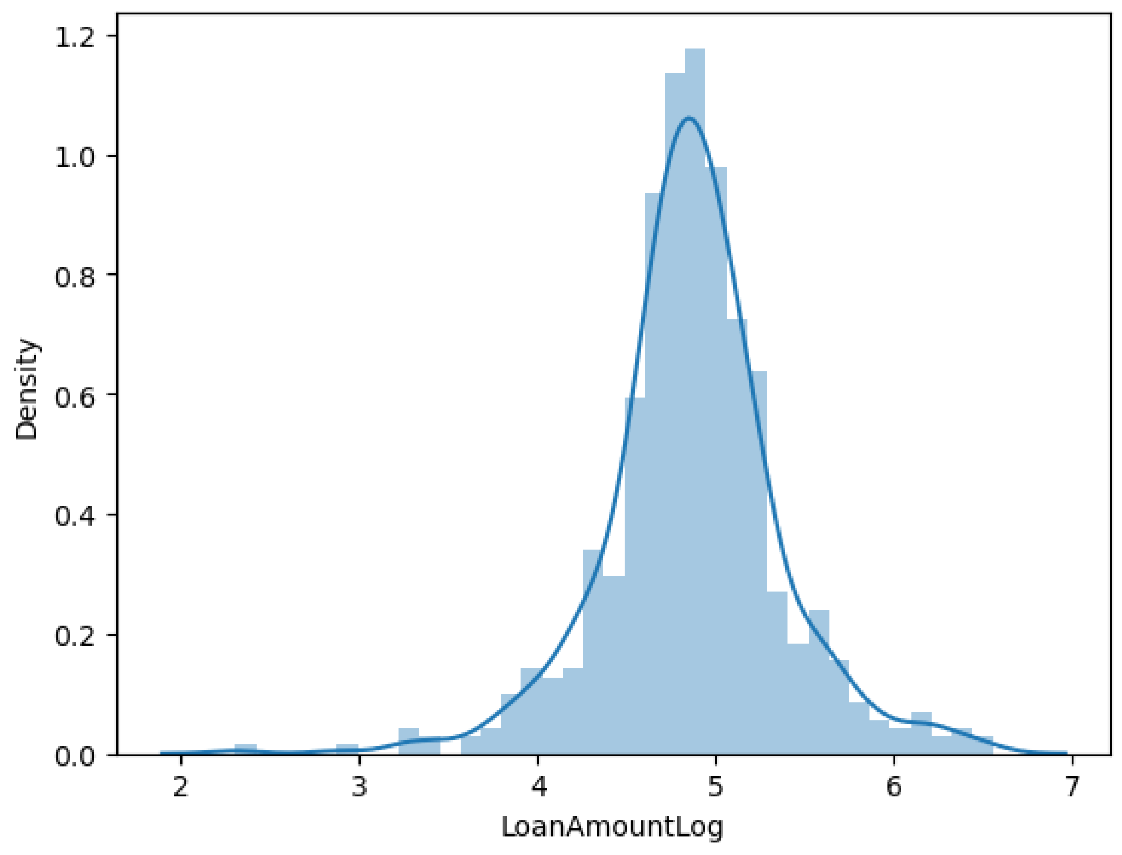
Out[27]:
```
<AxesSubplot:xlabel='CoapplicantIncomeLog', ylabel='Density'>
```



In [28]:
```
df['LoanAmountLog'] = np.log(df['LoanAmount']+1)
sns.distplot(df["LoanAmountLog"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibilit
y) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

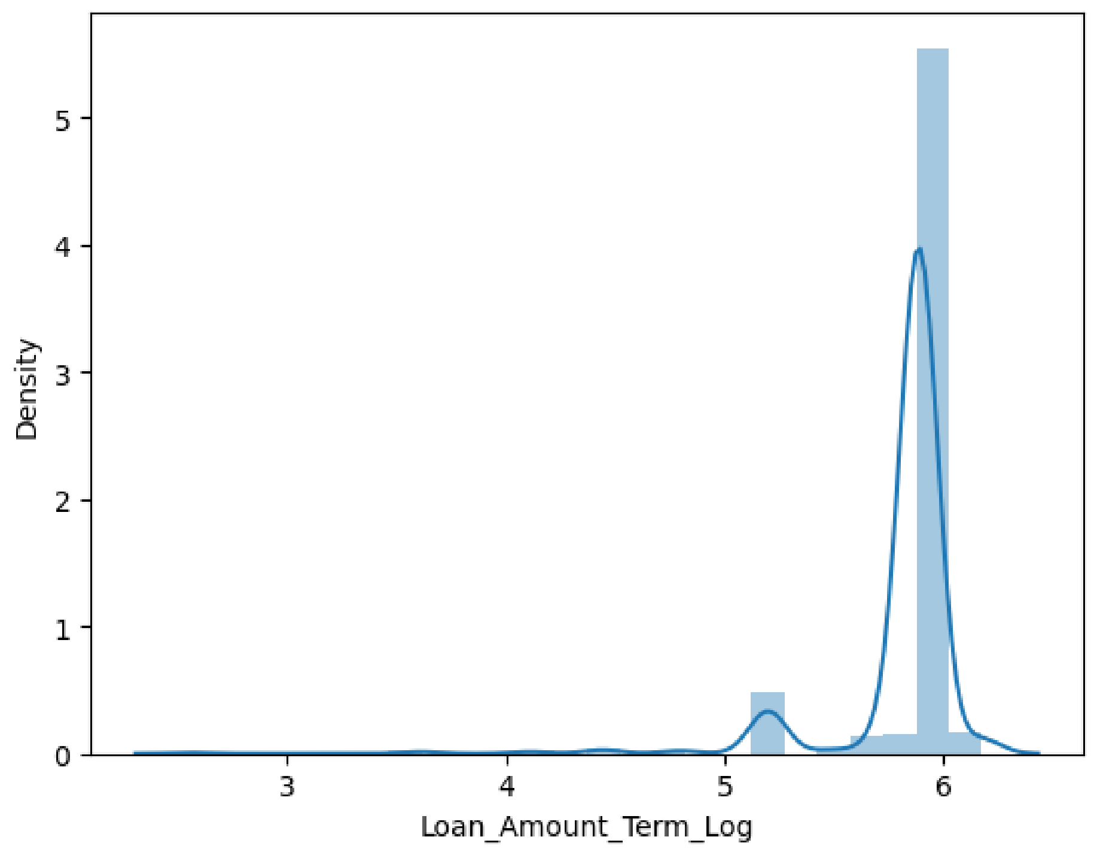
Out[28]:
```
<AxesSubplot:xlabel='LoanAmountLog', ylabel='Density'>
```



In [29]:
```
df['Loan_Amount_Term_Log'] = np.log(df['Loan_Amount_Term']+1)
sns.distplot(df["Loan_Amount_Term_Log"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibilit
y) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

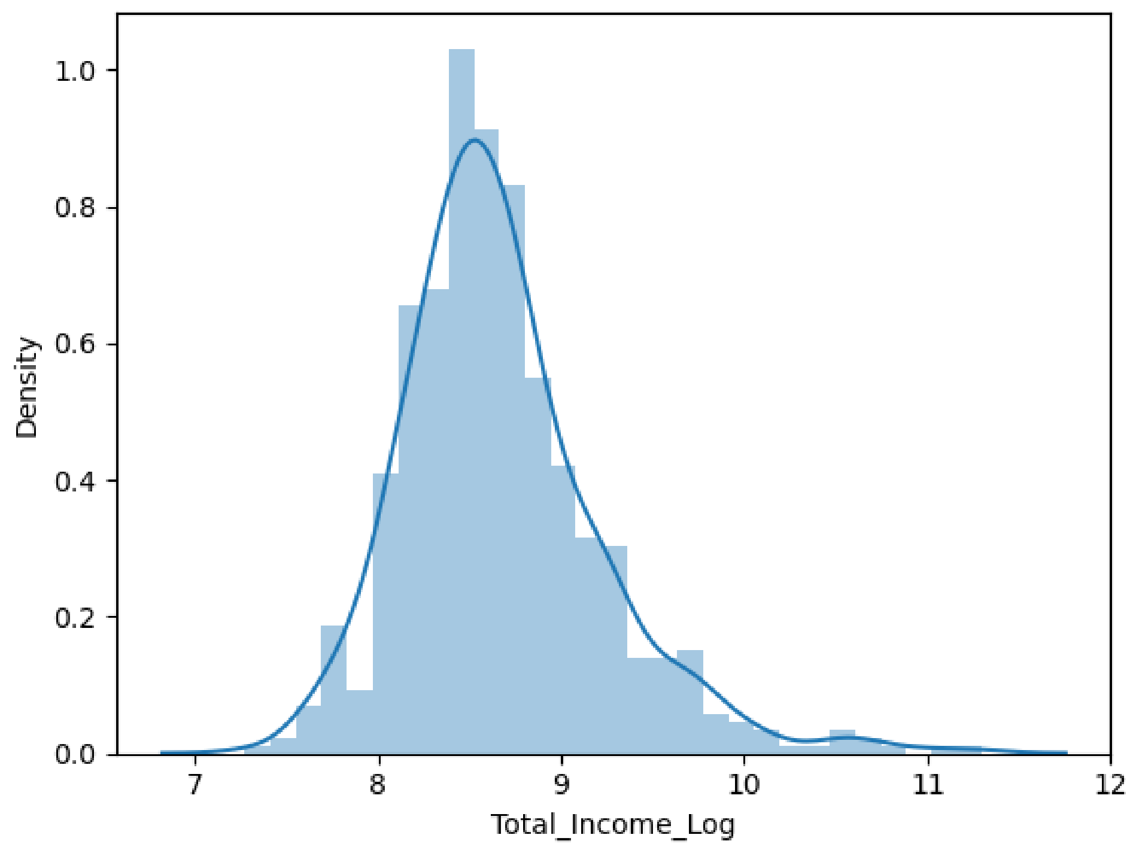
Out[29]:
```
<AxesSubplot:xlabel='Loan_Amount_Term_Log', ylabel='Density'>
```



In [30]:
```
df['Total_Income_Log'] = np.log(df['Total_Income']+1)
sns.distplot(df["Total_Income_Log"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibilit
y) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[30]:
```
<AxesSubplot:xlabel='Total_Income_Log', ylabel='Density'>
```



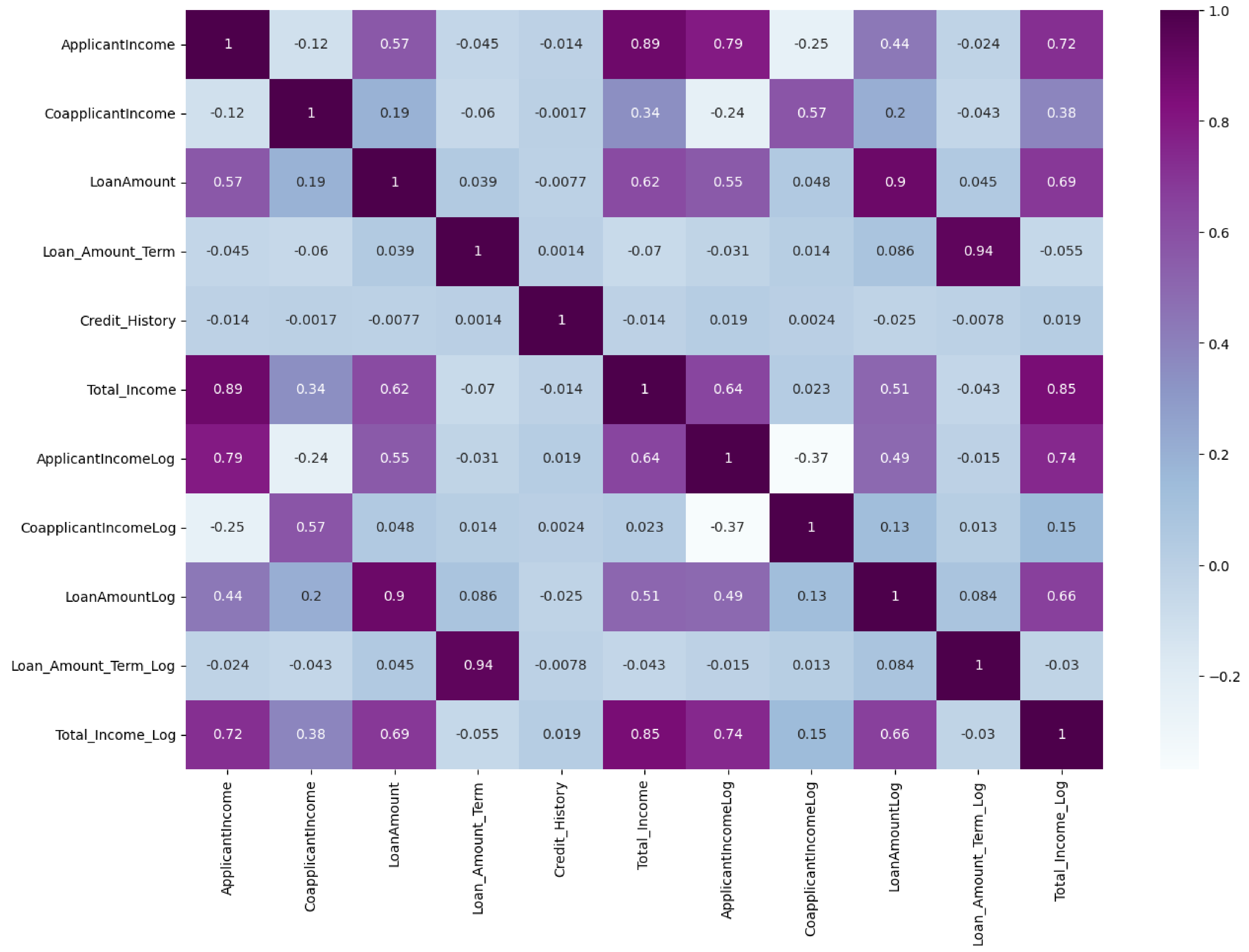
Correlation Matrix

In [27]:

```
corr = df.corr()
plt.figure(figsize=(15,10))
sns.heatmap(corr, annot = True, cmap="BuPu")
```

Out[27]:

```
<AxesSubplot:>
```



In [28]:

```
df.head()
```

Out[28]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status	Total_Income	ApplicantIncomeLog	CoapplicantIncomeLog	LoanAmountLog	Loan_Amount_Term_Log	Total_Income_Log
0	LP001002	Male	No	0	Graduate	No	5849	0.0	146.412162	360.0	1.0	Urban	Y	5849.0	8.674197	0.000000	4.993232	5.888878	8.674197
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.000000	360.0	1.0	Rural	N	6091.0	8.430327	7.319202	4.859812	5.888878	8.714732
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.000000	360.0	1.0	Urban	Y	3000.0	8.006701	0.000000	4.204693	5.888878	8.006701
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.000000	360.0	1.0	Urban	Y	4941.0	7.857094	7.765993	4.795791	5.888878	8.505525
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.000000	360.0	1.0	Urban	Y	6000.0	8.699681	0.000000	4.955827	5.888878	8.699681

In [31]:

```
cols = ['ApplicantIncome', 'CoapplicantIncome', "LoanAmount", "Loan_Amount_Term", "Total_Income", 'Loan_ID', 'CoapplicantIncomeLog']
df = df.drop(columns=cols, axis=1)
df.head()
```

Out[31]:

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Property_Area	Loan_Status	ApplicantIncomeLog	LoanAmountLog	Loan_Amount_Term_Log	Total_Income_Log
0	Male	No	0	Graduate	No	1.0	Urban	Y	8.674197	4.993232	5.888878	8.674197
1	Male	Yes	1	Graduate	No	1.0	Rural	N	8.430327	4.859812	5.888878	8.714732
2	Male	Yes	0	Graduate	Yes	1.0	Urban	Y	8.006701	4.204693	5.888878	8.006701
3	Male	Yes	0	Not Graduate	No	1.0	Urban	Y	7.857094	4.795791	5.888878	8.505525
4	Male	No	0	Graduate	No	1.0	Urban	Y	8.699681	4.955827	5.888878	8.699681

Label Encoding

In [32]:

```
from sklearn.preprocessing import LabelEncoder
cols = ['Gender',"Married","Education",'Self_Employed',"Property_Area","Loan_Status","Dependents"]
le = LabelEncoder()
for col in cols:
    df[col] = le.fit_transform(df[col])
```

In [33]:

```
df.tail()
```

Out[33]:

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Property_Area	Loan_Status	ApplicantIncomeLog	LoanAmountLog	Loan_Amount_Term_Log	Total_Income_Log
609	0	0	0	0	0	1.0	0	1	7.972811	4.276666	5.888878	7.972811
610	1	1	3	0	0	1.0	0	1	8.320448	3.713572	5.198497	8.320448
611	1	1	1	0	0	1.0	2	1	8.996280	5.537334	5.888878	9.025576
612	1	1	2	0	0	1.0	2	1	8.933796	5.236442	5.888878	8.933796
613	0	0	0	0	1	0.0	1	0	8.430327	4.897840	5.888878	8.430327

In [34]:

```
X = df.drop(columns=['Loan_Status'], axis=1)
y = df['Loan_Status']
```

Train Test Split

In [35]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

Model Training

In [34]:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train, y_train)
print("Accuracy is", model.score(x_test, y_test)*100)

```

```
Accuracy is 77.27272727272727
```

In [35]:

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(x_train, y_train)
print("Accuracy is", model.score(x_test, y_test)*100)

```

```
Accuracy is 70.12987012987013
```

In [36]:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(x_train, y_train)
print("Accuracy is", model.score(x_test, y_test)*100)

```

```
Accuracy is 79.22077922077922
```

In [37]:

```
from sklearn import svm
model= svm.SVC()
model.fit(x_train, y_train)
print("Accuracy is", model.score(x_test, y_test)*100)
```

```
Accuracy is 64.93506493506493
```

In [38]:

```
from sklearn.naive_bayes import GaussianNB
model= GaussianNB()
model.fit(x_train, y_train)
print("Accuracy is", model.score(x_test, y_test)*100)
```

```
Accuracy is 77.27272727272727
```

Confusion Matrix

In [37]:

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

Out[37]:

```
RandomForestClassifier()
```

In [40]:

```
from sklearn.metrics import confusion_matrix
y_pred = model.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[40]:

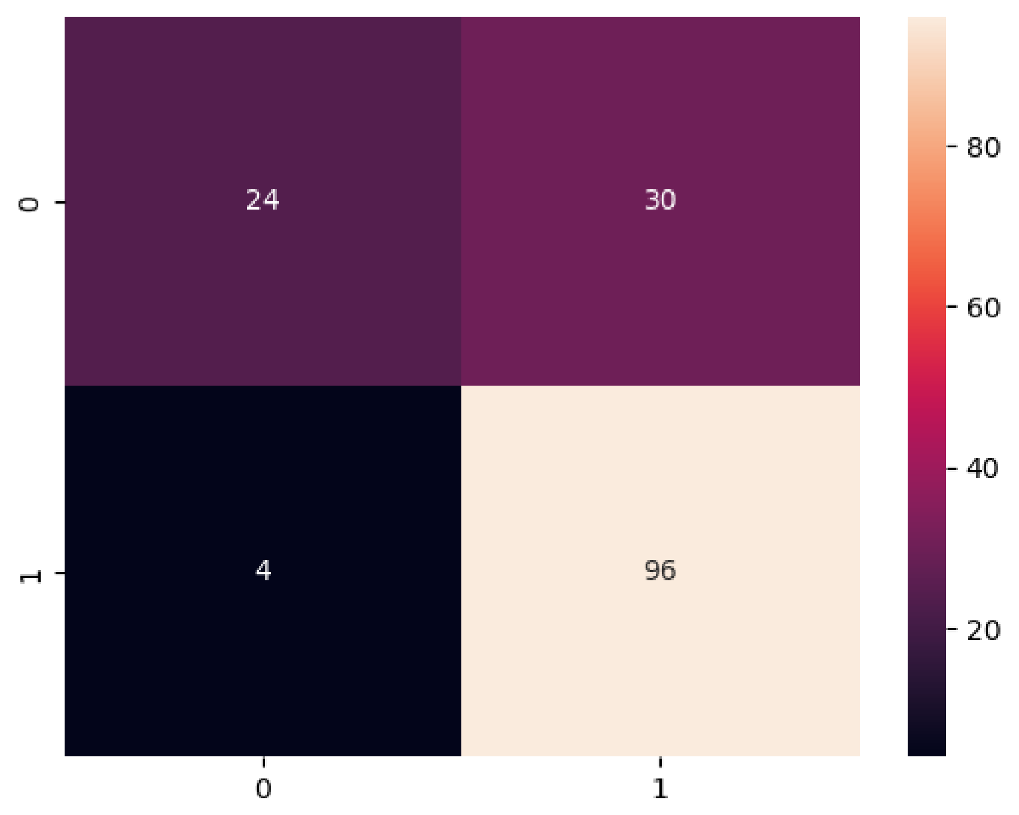
```
array([[24, 30],
       [ 4, 96]], dtype=int64)
```

In [41]:

```
sns.heatmap(cm, annot=True)
```

Out[41]:

```
<AxesSubplot:>
```



Prediction

In [59]:

```
array=np.array([[1,1,0,0,0,1.000000,1,9.114270,5.433722,5.888878,9.114270]])
from sklearn.metrics import confusion_matrix
y_pred = model.predict(array)
y_pred
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

Out[59]:

```
array([1])
```

Model Comparison

In [61]:

```
models=pd.DataFrame({'Model':['Decision Tree','Naive Bayes','SVM','Random Forest','Logistic Regression'],'Accuracy':[70.1298,77.2727,64.9350,79.2207,77.2727]})
models.sort_values(by='Accuracy',ascending=False)
```

Out[61]:

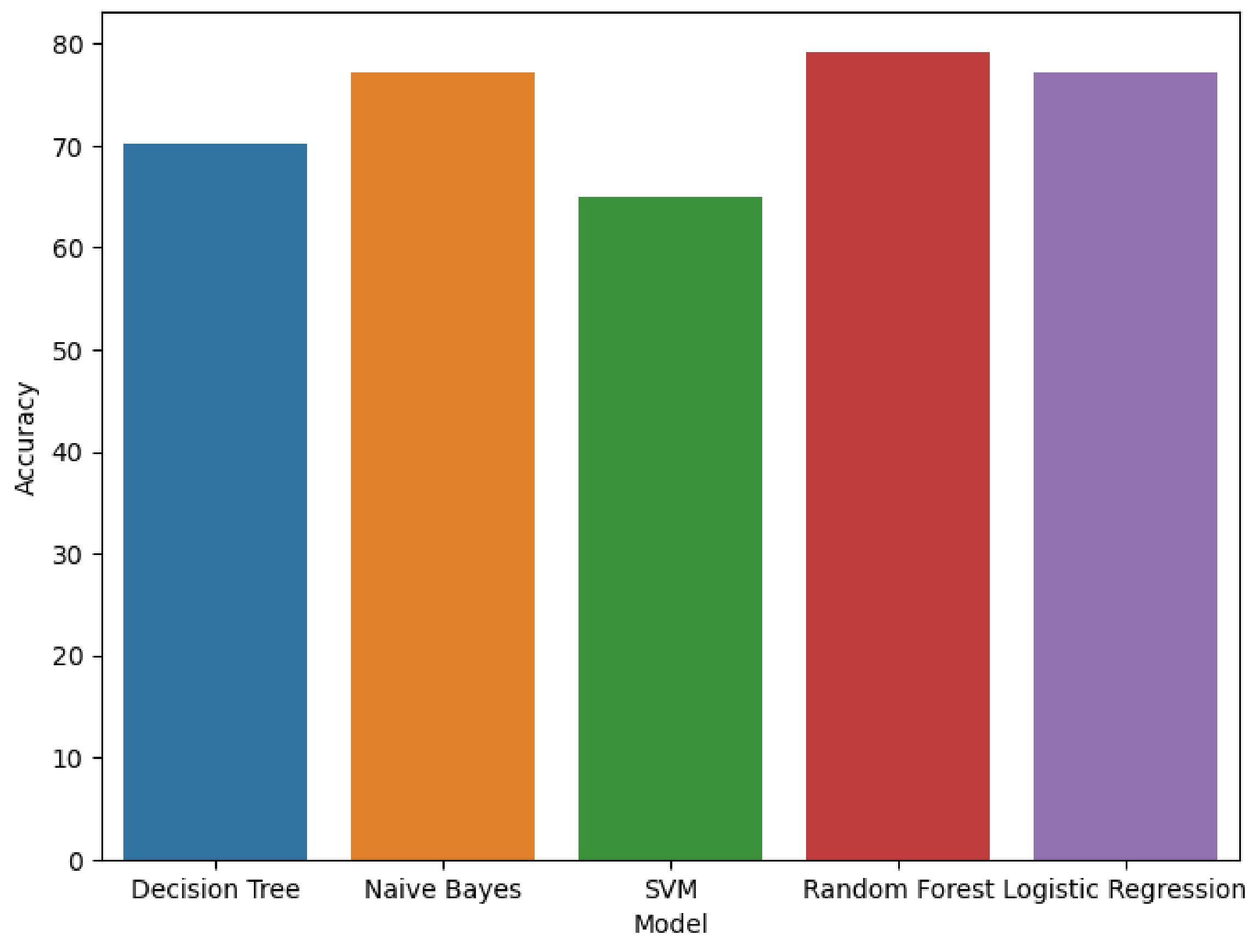
	Model	Accuracy
3	Random Forest	79.2207
1	Naive Bayes	77.2727
4	Logistic Regression	77.2727
0	Decision Tree	70.1298
2	SVM	64.9350

In [65]:

```
plt.figure(figsize=(8,6))
sns.barplot(x='Model',y='Accuracy',data=models)
plt.show
```

Out[65]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [ ]: