

# Loan Approval Prediction

August 29, 2025

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

```
[3]: df = pd.read_excel('Copy of loan.xlsx', engine='openpyxl')
```

```
[4]: df.head()
```

```
[4]:
```

	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

```
[5]: df.info()
```

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

```

```
[6]: df.isnull().sum()
```

```

[6]: 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

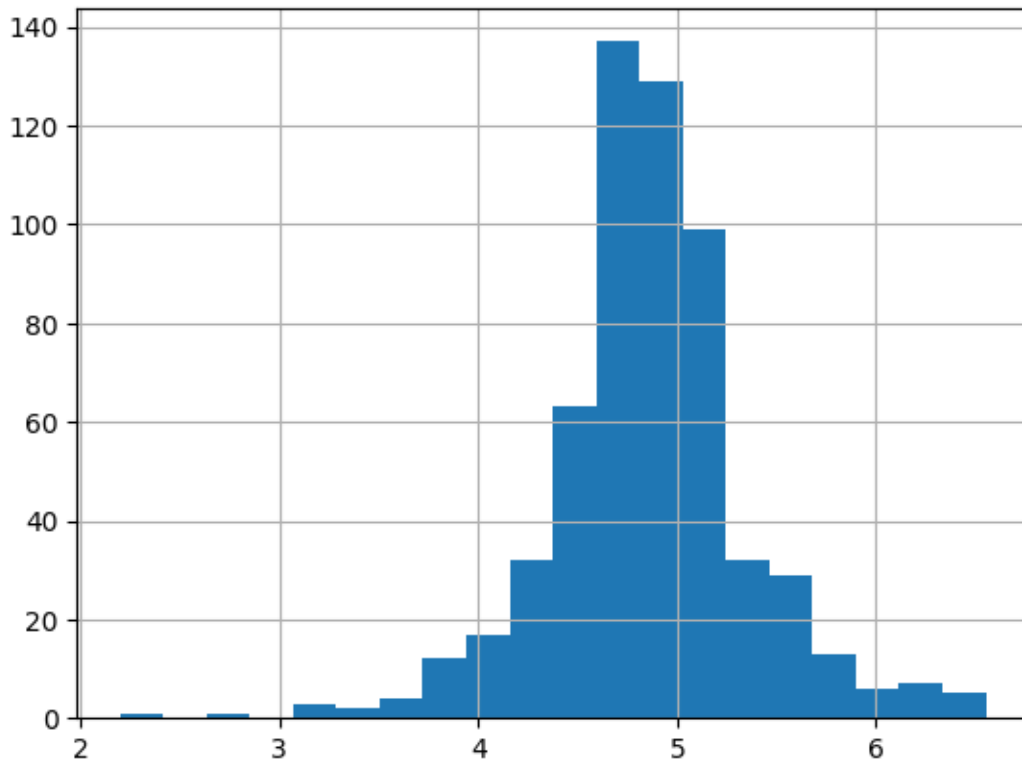
```

```

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

```

```
[7]: <Axes: >
```

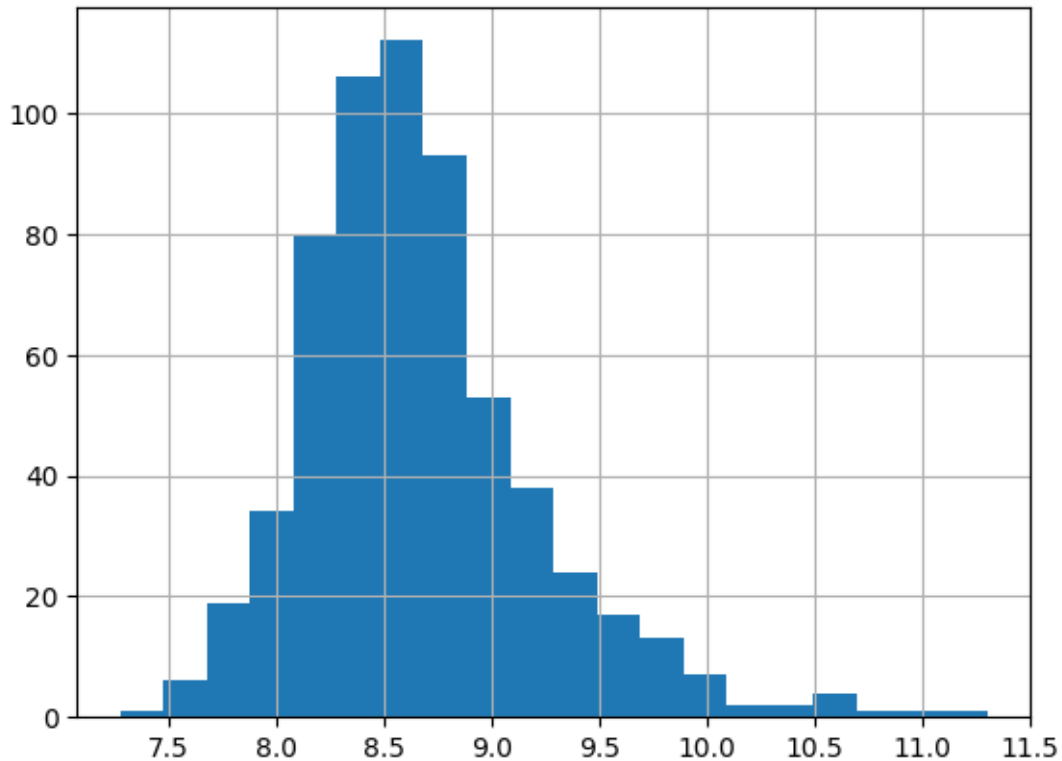


```
[8]: df.isnull().sum()
```

```
[8]: 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
```

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

[9]: <Axes: >



```
[13]: # Fill categorical columns with mode
df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])
df['Married'] = df['Married'].fillna(df['Married'].mode()[0])
df['Self_Employed'] = df['Self_Employed'].fillna(df['Self_Employed'].mode()[0])
df['Dependents'] = df['Dependents'].fillna(df['Dependents'].mode()[0])
df['Loan_Amount_Term'] = df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].
    ↪mode()[0])
df['Credit_History'] = df['Credit_History'].fillna(df['Credit_History'].
    ↪mode()[0])

# Fill numerical columns with mean
df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount'].mean())
df['loanAmount_log'] = df['loanAmount_log'].fillna(df['loanAmount_log'].mean())

# Check for remaining nulls
print(df.isnull().sum())
```

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

```

```

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

      x

```

```

[14]: array([[ 'Male', 'No', 0, ..., 1.0, 4.857444178729352, 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]],
      shape=(614, 8), dtype=object)

```

```

[15]: y

```

```

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

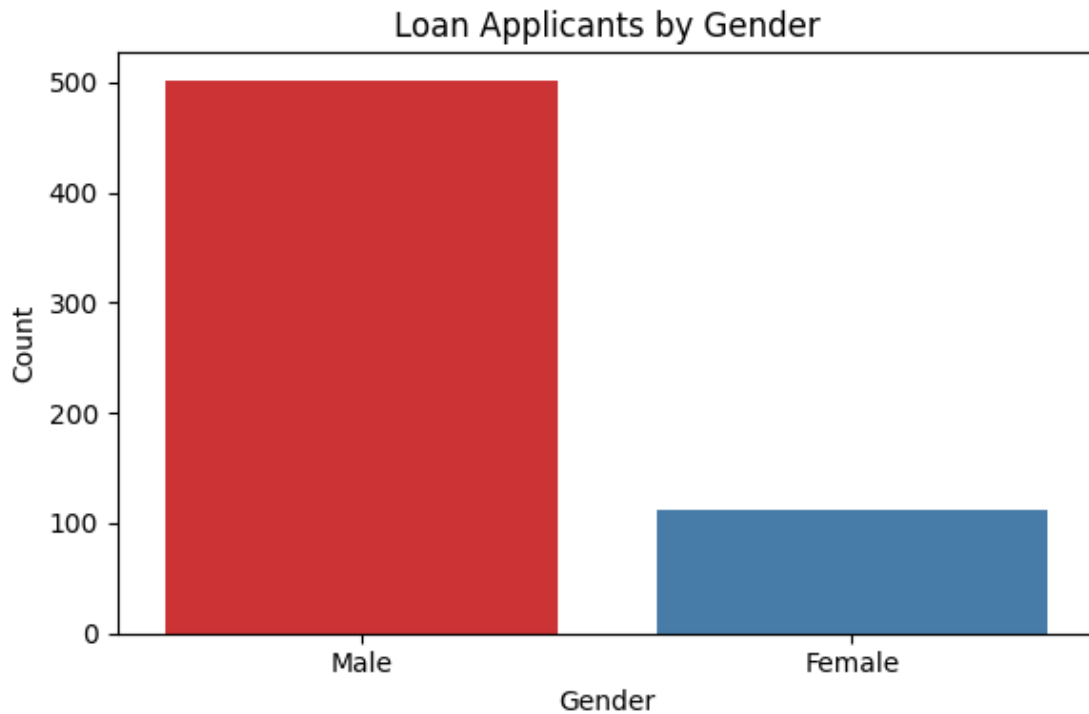
```
[16]: print ("per of missing gender is %2f%%" %((df['Gender'].isnull().sum()/df.
↪shape[0])*100))
```

per of missing gender is 0.000000%

```
[20]: plt.figure(figsize=(6,4))
print(df['Gender'].value_counts())
sns.countplot(x='Gender', hue='Gender', data=df, palette='Set1', legend=False)
plt.title('Loan Applicants by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

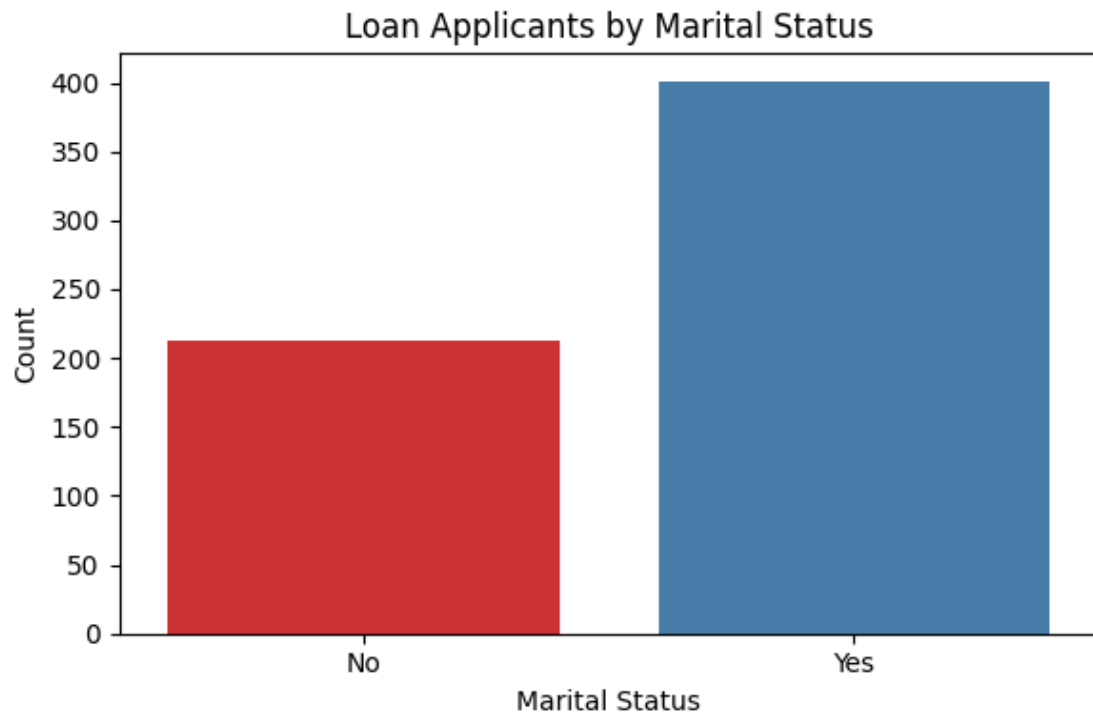
```
Gender
Male      502
```

```
Female    112
Name: count, dtype: int64
```



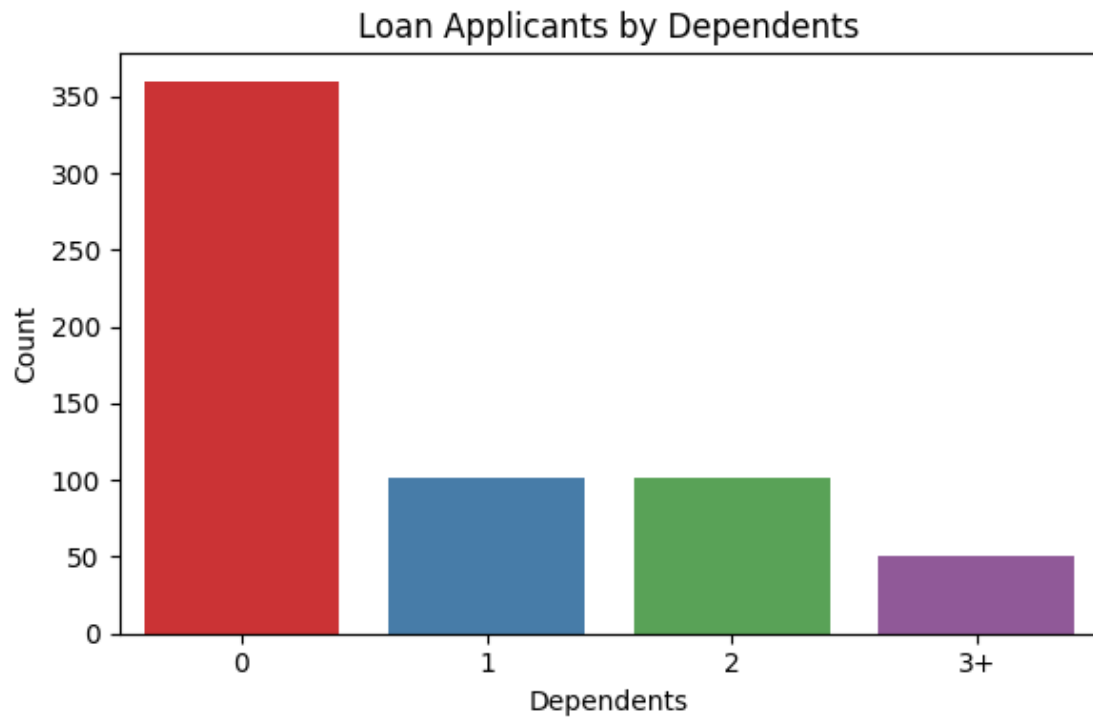
```
[23]: plt.figure(figsize=(6,4))
print(df['Married'].value_counts())
sns.countplot(x='Married', hue='Married', data=df, palette='Set1', legend=False)
plt.title('Loan Applicants by Marital Status')
plt.xlabel('Marital Status')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

```
Married
Yes     401
No      213
Name: count, dtype: int64
```



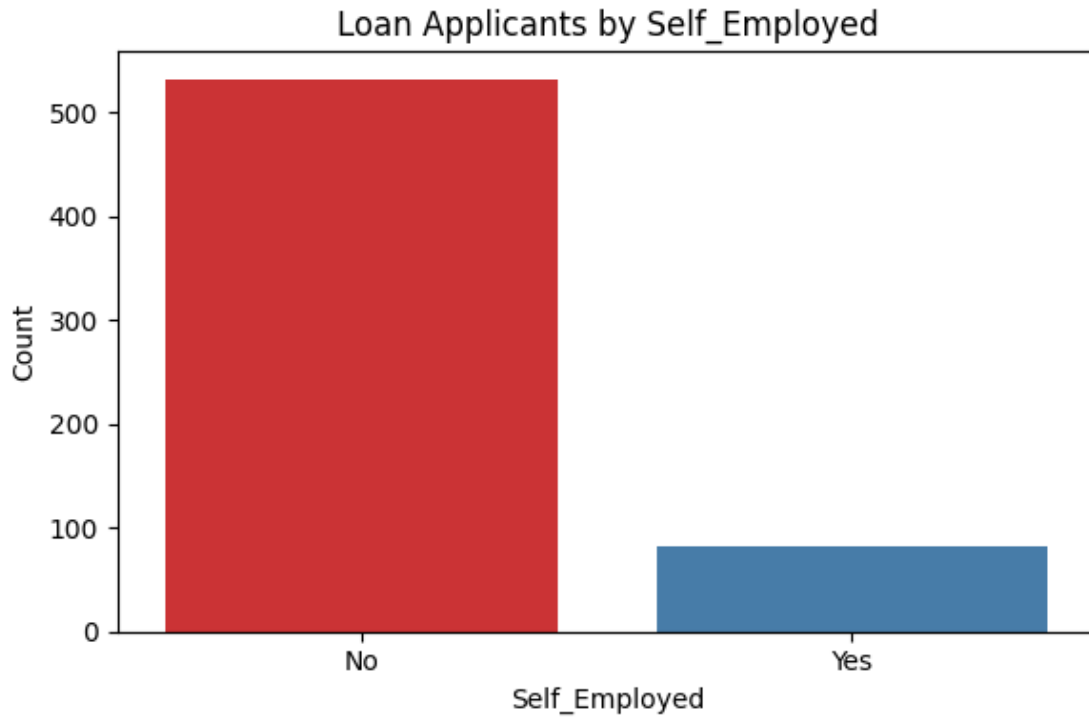
```
[24]: plt.figure(figsize=(6,4))
print(df['Dependents'].value_counts())
sns.countplot(x='Dependents', hue='Dependents', data=df, palette='Set1',
              legend=False)
plt.title('Loan Applicants by Dependents')
plt.xlabel('Dependents')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

```
Dependents
0      360
1      102
2      101
3+       51
Name: count, dtype: int64
```



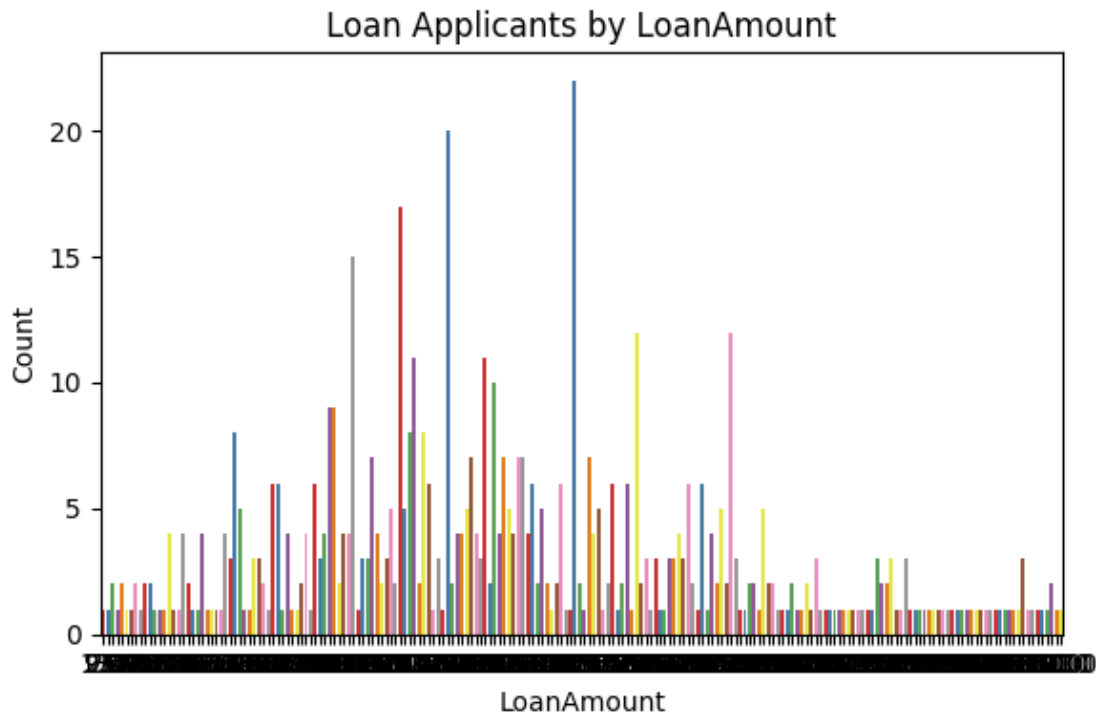
```
[27]: plt.figure(figsize=(6,4))
print(df['Self_Employed'].value_counts())
sns.countplot(x='Self_Employed', hue='Self_Employed', data=df, palette='Set1',
              legend=False)
plt.title('Loan Applicants by Self_Employed')
plt.xlabel('Self_Employed')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

```
Self_Employed
No      532
Yes      82
Name: count, dtype: int64
```



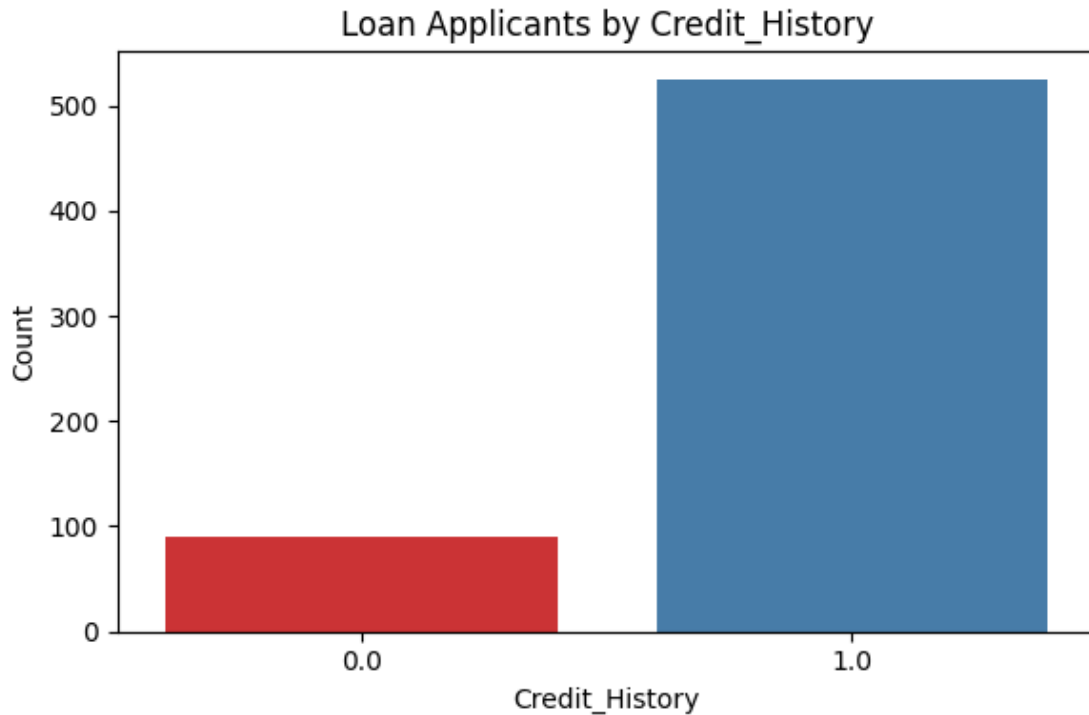
```
[28]: plt.figure(figsize=(6,4))
print(df['LoanAmount'].value_counts())
sns.countplot(x='LoanAmount', hue='LoanAmount', data=df, palette='Set1',
             legend=False)
plt.title('Loan Applicants by LoanAmount')
plt.xlabel('LoanAmount')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

```
LoanAmount
146.412162    22
120.000000    20
110.000000    17
100.000000    15
187.000000    12
..
292.000000     1
142.000000     1
350.000000     1
496.000000     1
253.000000     1
Name: count, Length: 204, dtype: int64
```



```
[29]: plt.figure(figsize=(6,4))
print(df['Credit_History'].value_counts())
sns.countplot(x='Credit_History', hue='Credit_History', data=df,
              palette='Set1', legend=False)
plt.title('Loan Applicants by Credit_History')
plt.xlabel('Credit_History')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

```
Credit_History
1.0      525
0.0       89
Name: count, dtype: int64
```



```
[34]: 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()
```

```
[40]: for i in range(0, 5):
        x_train[:, i] = labelencoder_x.fit_transform(x_train[:, i].astype(str))
```

```
# Encode column 7
```

```
x_train[:, 7] = labelencoder_x.fit_transform(x_train[:, 7].astype(str))
```

```
x_train
```

```
[40]: array([[1, 1, 0, ..., 1.0, 4.875197323201151, 98],
        [1, 0, 1, ..., 1.0, 5.278114659230517, 272],
        [1, 1, 0, ..., 0.0, 5.003946305945459, 76],
        ...,
        [1, 1, 3, ..., 1.0, 5.298317366548036, 217],
        [1, 1, 0, ..., 1.0, 5.075173815233827, 106],
        [0, 1, 0, ..., 1.0, 5.204006687076795, 142]],
        shape=(491, 8), dtype=object)
```

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

y_train
```

```
[41]: 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, 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, 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, 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, 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, 0,
1, 1, 0, 0, 1, 0, 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]])
```

```
[43]: for i in range(0, 5):
x_test[:, i] = labelencoder_x.fit_transform(x_test[:, i].astype(str))

# Encode column 7
x_test[:, 7] = labelencoder_x.fit_transform(x_test[:, 7].astype(str))

x_test
```

```
[43]: array([[1, 0, 0, 0, 4, 1.0, 4.430816798843313, 105],
[0, 0, 0, 0, 4, 1.0, 4.718498871295094, 42],
[1, 1, 0, 0, 4, 1.0, 5.780743515792329, 7],
[1, 1, 0, 0, 4, 1.0, 4.700480365792417, 100],
[1, 1, 2, 0, 4, 1.0, 4.574710978503383, 36],
[1, 1, 0, 1, 2, 0.0, 5.10594547390058, 89],
[1, 1, 3, 0, 2, 1.0, 5.056245805348308, 96],
[1, 0, 0, 0, 4, 1.0, 6.003887067106539, 18],
[1, 0, 0, 0, 4, 0.0, 4.820281565605037, 70],
[1, 1, 0, 0, 4, 1.0, 4.852030263919617, 72],
```

[0, 0, 0, 0, 4, 1.0, 4.430816798843313, 55],  
 [1, 1, 1, 0, 4, 1.0, 4.553876891600541, 33],  
 [0, 0, 0, 0, 4, 1.0, 5.634789603169249, 117],  
 [1, 1, 2, 0, 4, 1.0, 5.4638318050256105, 118],  
 [1, 1, 0, 0, 4, 1.0, 4.564348191467836, 21],  
 [1, 1, 1, 0, 4, 1.0, 4.204692619390966, 36],  
 [1, 0, 1, 1, 4, 1.0, 5.247024072160486, 47],  
 [1, 0, 0, 1, 4, 1.0, 4.882801922586371, 39],  
 [0, 0, 0, 0, 4, 1.0, 4.532599493153256, 1],  
 [1, 1, 0, 1, 4, 0.0, 5.198497031265826, 60],  
 [0, 1, 0, 0, 4, 0.0, 4.787491742782046, 90],  
 [1, 1, 0, 0, 4, 1.0, 4.962844630259907, 59],  
 [1, 1, 2, 0, 4, 1.0, 4.68213122712422, 112],  
 [1, 1, 2, 0, 4, 1.0, 5.10594547390058, 15],  
 [1, 1, 0, 0, 4, 1.0, 4.060443010546419, 50],  
 [1, 1, 1, 0, 4, 1.0, 5.521460917862246, 115],  
 [1, 0, 0, 0, 4, 1.0, 5.231108616854587, 119],  
 [1, 1, 0, 0, 4, 1.0, 5.231108616854587, 14],  
 [1, 1, 3, 0, 4, 0.0, 4.852030263919617, 57],  
 [0, 0, 0, 0, 4, 0.0, 4.634728988229636, 67],  
 [1, 1, 0, 0, 4, 1.0, 5.429345628954441, 120],  
 [1, 0, 0, 1, 4, 1.0, 3.871201010907891, 62],  
 [1, 1, 1, 1, 4, 1.0, 4.499809670330265, 69],  
 [1, 1, 0, 0, 4, 1.0, 5.19295685089021, 5],  
 [1, 1, 0, 0, 4, 1.0, 4.857444178729352, 116],  
 [0, 1, 0, 1, 4, 0.0, 5.181783550292085, 74],  
 [1, 1, 0, 0, 4, 1.0, 5.147494476813453, 83],  
 [1, 0, 0, 1, 4, 1.0, 4.836281906951478, 54],  
 [1, 1, 0, 0, 4, 1.0, 4.852030263919617, 94],  
 [1, 1, 2, 1, 4, 1.0, 4.68213122712422, 38],  
 [0, 0, 0, 0, 4, 1.0, 4.382026634673881, 110],  
 [1, 1, 3, 0, 4, 0.0, 4.812184355372417, 86],  
 [1, 1, 2, 0, 1, 1.0, 2.833213344056216, 0],  
 [1, 1, 1, 1, 4, 1.0, 5.062595033026967, 85],  
 [1, 0, 0, 0, 4, 1.0, 4.330733340286331, 35],  
 [1, 0, 0, 0, 4, 1.0, 5.231108616854587, 17],  
 [1, 1, 1, 0, 4, 1.0, 4.7535901911063645, 31],  
 [0, 0, 0, 0, 4, 1.0, 4.74493212836325, 52],  
 [1, 1, 1, 0, 4, 1.0, 4.852030263919617, 91],  
 [1, 0, 0, 0, 4, 1.0, 4.941642422609304, 97],  
 [1, 1, 3, 1, 4, 1.0, 4.30406509320417, 99],  
 [1, 1, 0, 0, 4, 1.0, 4.867534450455582, 104],  
 [1, 1, 0, 1, 4, 1.0, 4.672828834461906, 46],  
 [1, 0, 0, 0, 4, 1.0, 4.857444178729352, 79],  
 [1, 1, 0, 0, 4, 1.0, 4.718498871295094, 32],  
 [1, 1, 0, 0, 4, 1.0, 5.556828061699537, 10],  
 [1, 1, 0, 0, 4, 1.0, 4.553876891600541, 49],

[1, 0, 0, 1, 4, 1.0, 4.890349128221754, 93],  
 [1, 1, 2, 0, 4, 1.0, 5.123963979403259, 80],  
 [1, 0, 0, 0, 4, 1.0, 4.787491742782046, 41],  
 [0, 0, 0, 0, 4, 0.0, 4.919980925828125, 11],  
 [0, 0, 0, 0, 4, 1.0, 5.365976015021851, 6],  
 [1, 1, 0, 1, 4, 1.0, 4.74493212836325, 53],  
 [0, 0, 0, 0, 4, 0.0, 4.330733340286331, 26],  
 [1, 1, 2, 0, 4, 1.0, 4.890349128221754, 87],  
 [1, 1, 1, 0, 4, 1.0, 5.752572638825633, 16],  
 [1, 1, 0, 0, 4, 1.0, 5.075173815233827, 92],  
 [1, 0, 0, 0, 4, 1.0, 4.912654885736052, 63],  
 [1, 1, 0, 0, 4, 1.0, 5.204006687076795, 101],  
 [1, 0, 0, 1, 4, 1.0, 4.564348191467836, 78],  
 [1, 0, 0, 0, 4, 1.0, 4.204692619390966, 103],  
 [0, 1, 0, 0, 4, 1.0, 4.867534450455582, 66],  
 [1, 1, 2, 1, 4, 1.0, 5.056245805348308, 75],  
 [1, 1, 1, 1, 2, 1.0, 4.919980925828125, 98],  
 [0, 1, 0, 0, 4, 1.0, 4.969813299576001, 71],  
 [1, 1, 0, 1, 3, 1.0, 4.820281565605037, 73],  
 [1, 0, 0, 0, 4, 1.0, 4.499809670330265, 25],  
 [1, 0, 3, 0, 4, 1.0, 5.768320995793772, 22],  
 [1, 1, 2, 0, 4, 1.0, 4.718498871295094, 4],  
 [0, 0, 0, 0, 4, 0.0, 4.7535901911063645, 40],  
 [0, 0, 0, 0, 5, 1.0, 4.727387818712341, 48],  
 [1, 1, 1, 0, 4, 1.0, 6.214608098422191, 23],  
 [0, 0, 0, 0, 4, 1.0, 5.267858159063328, 109],  
 [1, 1, 2, 0, 4, 1.0, 5.231108616854587, 113],  
 [1, 0, 0, 0, 5, 1.0, 4.2626798770413155, 77],  
 [1, 1, 0, 0, 0, 1.0, 4.709530201312334, 111],  
 [1, 1, 0, 0, 4, 1.0, 4.700480365792417, 61],  
 [1, 1, 2, 0, 4, 1.0, 5.298317366548036, 12],  
 [1, 0, 1, 0, 2, 1.0, 4.727387818712341, 30],  
 [1, 1, 1, 0, 4, 1.0, 4.6443908991413725, 51],  
 [0, 1, 0, 1, 4, 1.0, 4.605170185988092, 29],  
 [1, 0, 0, 0, 4, 1.0, 4.30406509320417, 88],  
 [1, 1, 1, 0, 6, 1.0, 5.147494476813453, 108],  
 [1, 1, 3, 0, 3, 0.0, 5.19295685089021, 107],  
 [0, 0, 0, 0, 4, 1.0, 4.2626798770413155, 44],  
 [1, 0, 0, 1, 2, 0.0, 4.836281906951478, 76],  
 [1, 0, 0, 0, 2, 1.0, 5.1647859739235145, 102],  
 [1, 0, 0, 0, 4, 1.0, 4.969813299576001, 84],  
 [1, 1, 2, 1, 4, 1.0, 4.394449154672439, 68],  
 [1, 1, 1, 0, 4, 1.0, 5.231108616854587, 3],  
 [1, 1, 0, 0, 4, 1.0, 5.351858133476067, 114],  
 [1, 1, 0, 0, 4, 1.0, 4.605170185988092, 28],  
 [1, 1, 2, 0, 4, 1.0, 4.787491742782046, 9],  
 [1, 0, 0, 0, 2, 1.0, 4.787491742782046, 8],

```
[1, 1, 3, 0, 4, 1.0, 4.852030263919617, 82],
[1, 0, 0, 0, 4, 1.0, 4.8283137373023015, 65],
[1, 0, 0, 1, 4, 1.0, 4.6443908991413725, 58],
[0, 0, 0, 0, 4, 1.0, 4.477336814478207, 2],
[1, 1, 0, 1, 4, 1.0, 4.553876891600541, 34],
[1, 1, 3, 1, 2, 1.0, 4.394449154672439, 27],
[1, 0, 0, 0, 4, 1.0, 5.298317366548036, 95],
[0, 0, 0, 0, 4, 1.0, 4.90527477843843, 13],
[1, 0, 0, 0, 5, 1.0, 4.727387818712341, 31],
[1, 1, 2, 0, 4, 1.0, 4.248495242049359, 37],
[1, 1, 0, 1, 4, 0.0, 5.303304908059076, 81],
[1, 1, 0, 0, 2, 0.0, 4.499809670330265, 64],
[0, 0, 0, 0, 4, 1.0, 4.430816798843313, 45],
[1, 0, 0, 0, 4, 1.0, 4.897839799950911, 43],
[1, 1, 2, 0, 4, 1.0, 5.170483995038151, 106],
[1, 1, 3, 0, 4, 1.0, 4.867534450455582, 19],
[1, 1, 0, 0, 4, 1.0, 6.077642243349034, 20],
[1, 1, 3, 1, 2, 0.0, 4.248495242049359, 56],
[1, 1, 1, 0, 4, 1.0, 4.564348191467836, 24]], dtype=object)
```

```
[45]: Labelencoder_y = LabelEncoder()

y_test = Labelencoder_y.fit_transform(y_test)

y_test
```

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

```
[47]: from sklearn.preprocessing import StandardScaler

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

```
[48]: from sklearn.ensemble import RandomForestClassifier

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

```
[48]: RandomForestClassifier()
```

```
[50]: 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 0.7317073170731707

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

```
[51]: from sklearn.naive_bayes import GaussianNB
nb_clf = GaussianNB()
nb_clf.fit(x_train, y_train)
```

```
[51]: GaussianNB()
```

```
[53]: y_pred = nb_clf.predict(x_test)
print(" acc of gaussianNb is %.", metrics.accuracy_score(y_pred, y_test))
```

acc of gaussianNb is %. 0.8292682926829268

```
[54]: y_pred
```

```
[54]: 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, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
          1, 1, 1, 1, 1, 1, 1, 1, 1, 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, 0, 1, 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, 0, 1])
```

```
[55]: from sklearn.tree import DecisionTreeClassifier
dt_clf = DecisionTreeClassifier()
dt_clf.fit(x_train, y_train)
```

```
[55]: DecisionTreeClassifier()
```

```
[56]: y_pred = dt_clf.predict(x_test)
print ("acc of DT is", metrics.accuracy_score(y_pred, y_test))
```

acc of DT is 0.6178861788617886

```
[57]: y_pred
```

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

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: