1. Data Loading and Overview*

```
import seaborn as sns
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python

# For example, here's several helpful packages to load

import numpy as np # linear algebra import pandas as pd # data
processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory # For example, running this (by clicking
run or pressing Shift+Enter) will list all files under the input directory

import os for dirname, _, filenames in
os.walk('/kaggle/input'):

    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version us
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session


df=pd.read_csv('/kaggle/input/loan-dataset/loan-train.csv')
df2=pd.read_csv('/kaggle/input/loan-dataset/loan-test.csv')
```

2. Data Exploration

```
df.shape
```

```
(614, 13)
```

```
df2.shape
```

```
(367, 12)
```

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

```
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
```

Double-click (or enter) to edit

```
df.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
	614.000000	614.000000	592.000000	600.00000	564.000000
	5403.459283	1621.245798	146.412162	342.00000	0.842199
count mean	6109.041673	2926.248369	85.587325	65.12041	0.364878
std 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

```
new_df=df.dropna()
new_df.shape
df.shape
```

```
(614, 13)
```

```
pd.crosstab(df['Credit_History'],df['Loan_Status'], margins=True)
```

Loan_Status	N	Y	All
Credit_History			
0.0	82	7	89
1.0	97	378	475
All	179	385	564

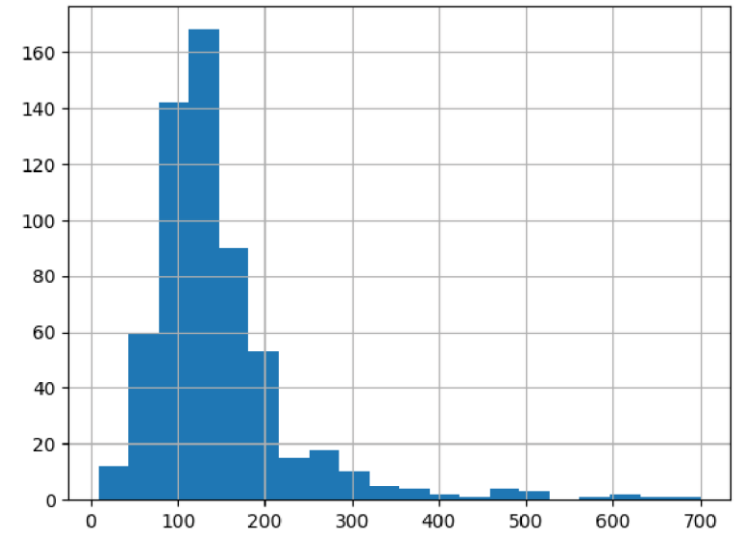
```
pd.crosstab(df['Property_Area'],df['Loan_Status'], margins=True)
```

Loan_Status	N	Y	All
Property_Area			
Rural	69	110	179
Semiurban	54	179	233
Urban	69	133	202
All	192	422	614

Data Visulaizing /(EDA)

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

```
<Axes: >
```




```
df['LoanAmount_log']=np.log(df['LoanAmount'])
df.sample(5)
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amou
263	LP001871	Female	No	0	Graduate	No	7200	0.0	120.0	
308	LP001996	Male	No	0	Graduate	No	20233	0.0	480.0	
184	LP001639	Female	Yes	0	Graduate	No	3625	0.0	108.0	
552	LP002785	Male	Yes	1	Graduate	No	3333	3250.0	158.0	
371	LP002197	Male	Yes	2	Graduate	No	5185	0.0	155.0	

```
Start coding or generate with AI.
```

3. Data Cleaning (Handling Missing Values)

```
import warnings
warnings.filterwarnings('ignore')
df['Gender'].fillna(df['Gender'].mode()[0],inpla
ce=True) df['Gender'].isnull().sum()
```

```
0
```

```
df['Married'].fillna(df['Married'].mode()[0],inplace=True)
df['Married'].isnull().sum()
```

```
0
```

```
df['Dependents'].fillna(df['Dependents'].mode()[0],inplace=True)
df['Dependents'].isnull().sum()
```

```
0
```

```
df.sample(10)
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amou

350	LP002139	Male	Yes	0	Graduate	No	9083	0.0	228.0
270	LP001888	Female	No	0	Graduate	No	3237	0.0	30.0
299	LP001963	Male	Yes	1	Graduate	No	2014	2925.0	113.0
582	LP002894	Female	Yes	0	Graduate	No	3166	0.0	36.0
580	LP002892	Male	Yes	2	Graduate	No	6540	0.0	205.0
365	LP002181	Male	No	0	Not Graduate	No	6216	0.0	133.0
584	LP002911	Male	Yes	1	Graduate	No	2787	1917.0	146.0
506	LP002624	Male	Yes	0	Graduate	No	20833	6667.0	480.0
377	LP002223	Male	Yes	0	Graduate	No	4310	0.0	130.0

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

```
Loan_ID               0
Gender                0
Married               0
Dependents            0
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
```

```
df['Self_Employed'].fillna(df['Self_Employed'].mode()[0],inplace=True)
```

Double-click (or enter) to edit

```
df['LoanAmount'].fillna(df['LoanAmount'].mean(),inplace=True)
```

```
df['LoanAmount_log'].fillna(df['LoanAmount_log'].mean(),inplace=True)
```

Double-click (or enter) to edit

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

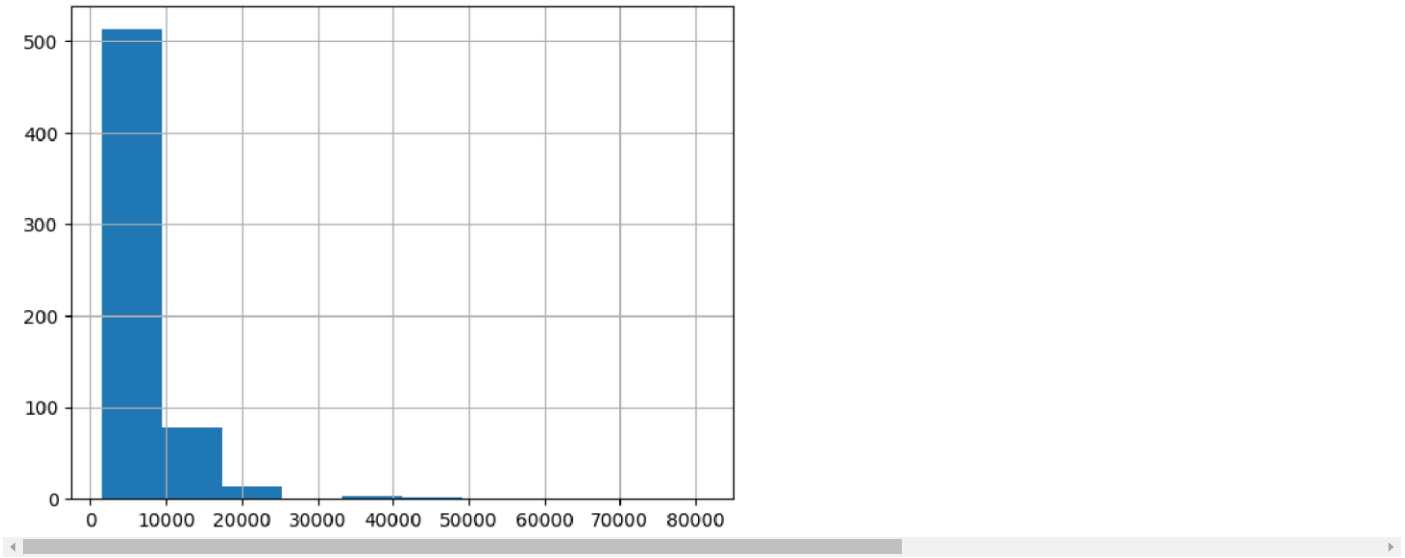
```
df['Credit_History'].fillna(df['Credit_History'].mode()[0],inplace=True)
df.isnull().sum()
```

```
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
```

4. Feature Engineering

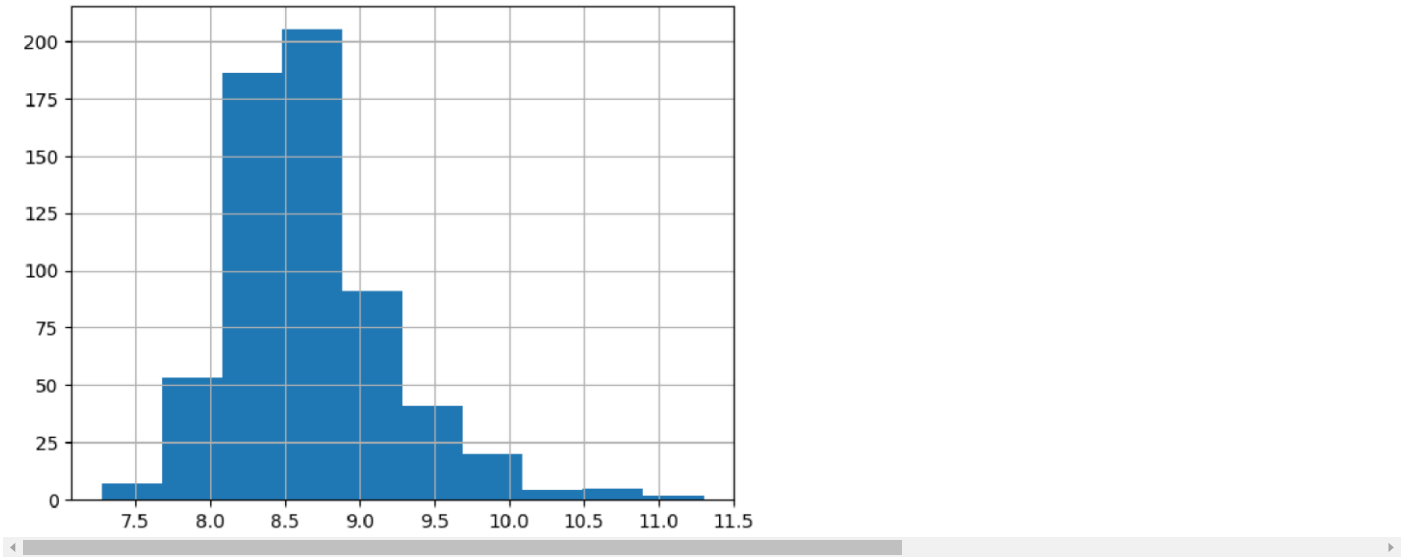
```
df['Total_Income']=df['ApplicantIncome']+df['CoapplicantIncome']
df['Total_Income'].hist()
```

```
<Axes: >
```




```
df['Total_Income_Log']=np.log(df['Total_Income'])
df['Total_Income_Log'].hist()
```

```
<Axes: >
```




```
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
Total_Income         0
Total_Income_Log     0
dtype: int64
```

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

```
X
```

```
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]],
       dtype=object)
```

```
Y
```

```
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', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y',
       'N', 'Y', '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', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y',
       'N', 'Y', 'Y', 'N', 'Y', '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', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', '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', 'Y', 'N', 'Y', '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', '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', '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', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N',
       'N', 'Y', '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', '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)
```

```
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)
```

5. Data Preprocessing (Encoding Categorical Variables)

```
from sklearn.preprocessing import LabelEncoder

labelencoder_X=LabelEncoder() Start coding or

generate with AI.


for i in range(5):
    X_train[:,i]= labelencoder_X.fit_transform(X_train[:,i])
    i


X_train [:,7] = labelencoder_X.fit_transform(X_train[:,7])


for i in range(5):
    X_test[:,i]= labelencoder_X.fit_transform(X_test[:,i])


X_test [:,7] = labelencoder_X.fit_transform(X_test[:,7])


labelencoder_y=LabelEncoder()
y_train=labelencoder_y.fit_transform(y_train)


labelencoder_y=LabelEncoder()
y_test=labelencoder_y.fit_transform(y_test)
```

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
X_train=ss.fit_transform(X_train)
X_test=ss.fit_transform(X_test)
```

Model Training (Decision Tree)

```
from sklearn.tree import DecisionTreeClassifier
DTClassifier= DecisionTreeClassifier (criterion='entropy', random_state=0)
DTClassifier.fit(X_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

```
y_pred= DTClassifier.predict(X_test)


from sklearn import metrics
print('The accuracy of decision tree is:', metrics.accuracy_score(y_pred,y_test))
```

```
The accuracy of decision tree is: 0.7073170731707317
```

Model Training (Naive Bayes)

```
from sklearn.naive_bayes import GaussianNB
NBClassifier= GaussianNB() NBClassifier.fit(X_train,y_train)
```

```
GaussianNB
GaussianNB()
```

```
y_pred= NBClassifier.predict(X_test)


y_pred
```

```
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, 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, 1, 1, 0, 1, 1, 0, 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, 0, 0, 1, 1, 1, 1, 1, 0, 1])
```

Test Data Preprocessing and Predictions*

```
testdata=pd.read_csv('/kaggle/input/loan-dataset/loan-test.csv')
testdata.shape
```

```
(367, 12)
```

```
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)
```

```
testdata.LoanAmount= testdata.LoanAmount.fillna(testdata.LoanAmount.mean())
```

```
testdata['LoanAmount_log']=np.log(testdata['LoanAmount'])
testdata.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
LoanAmount_log       0
dtype: int64
```

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

```
testdata.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	

```
testdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
0   Loan_ID            367 non-null    object
1   Gender             367 non-null    object
2   Married            367 non-null    object
3   Dependents         367 non-null    object
4   Education          367 non-null    object
5   Self_Employed      367 non-null    object
6   ApplicantIncome    367 non-null    int64
7   CoapplicantIncome  367 non-null    int64
8   LoanAmount         367 non-null    float64
9   Loan_Amount_Term   367 non-null    float64
10  Credit_History     367 non-null    float64
11  Property_Area      367 non-null    object
```

```
12  LoanAmount_log     367 non-null    float64
13  TotalIncome        367 non-null    int64   14  TotalIncome_log    367
    non-null    float64 dtypes: float64(5), int64(3), object(7)
memory usage: 43.1+ KB
```

Data Visulaizing

```
testdata.iloc[:,[1,2,3,4,9,10,14]]
# sns.histplot(testdata['Loan_Amount_Term'],)
```

	Gender	Married	Dependents	Education	Loan_Amount_Term	Credit_History	TotalIncome_log
0	Male	Yes	0	Graduate	360.0	1.0	8.651724
1	Male	Yes	1	Graduate	360.0	1.0	8.428581
2	Male	Yes	2	Graduate	360.0	1.0	8.824678
3	Male	Yes	2	Graduate	360.0	1.0	8.494129
4	Male	No	0	Not Graduate	360.0	1.0	8.094378
...	...	...	...	...	...	...	...
362	Male	Yes	3+	Not Graduate	360.0	1.0	8.663196
363	Male	Yes	0	Graduate	360.0	1.0	8.490233
364	Male	No	0	Graduate	360.0	1.0	8.564649
365	Male	Yes	0	Graduate	360.0	1.0	8.908289
366	Male	No	0	Graduate	180.0	1.0	9.126959

367 rows × 7 columns

10. Predictions on Test Data

	Gender	Married	Dependents	Education	Loan_Amount_Term	Credit_History	TotalIncome	TotalIncome_log
0	Male	Yes	0	Graduate	360.0	1.0	5720	8.651724
1	Male	Yes	1	Graduate	360.0	1.0	4576	8.428581
2	Male	Yes	2	Graduate	360.0	1.0	6800	8.824678
3	Male	Yes	2	Graduate	360.0	1.0	4886	8.494129
4	Male	No	0	Not Graduate	360.0	1.0	3276	8.094378

```
test=testdata.iloc[:,np.r_[1:5,9:11, 13:15]]
test. head()
```

```
for i in range(0,5):
  #  print(i)
    test.iloc[:,i]=labelencoder_X.fit_transform(test.iloc[:,i])
```

```
test. head()
```

	Gender	Married	Dependents	Education	Loan_Amount_Term	Credit_History	TotalIncome	TotalIncome_log
0	1	1	0	0	10.0	1.0	5720	8.651724
1	1	1	1	0	10.0	1.0	4576	8.428581
2	1	1	2	0	10.0	1.0	6800	8.824678
3	1	1	2	0	10.0	1.0	4886	8.494129
4	1	0	0	1	10.0	1.0	3276	8.094378

```
test.iloc[:,7]=labelencoder_X.fit_transform(test.iloc[:,7])


test= ss.fit_transform(test)
test[0]
```

```
array([ 0.48547939,  0.75835829, -0.75822199, -0.5448117 ,  0.30677633,
        0.4376739 , -0.12618159,  0.34823304])
```

```
NBClassifier.predict(test)
```

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

```
var=NBClassifier.predict([[ 0.48547939, 0.75835829, -0.75822199, -0.5448117, 0.30677633, 0.4376739,-0.12618159, 0.34823304]])
if var==[1]:
    print('Yes you\'re eligible for the loan')
else:
    print('Sorry you\'re not eligible for the loan')
```

Final Output and Results

```
print('The accuracy of Naive Bayes is: ', metrics.accuracy_score(y_pred,y_test))
```

```
The accuracy of Naive Bayes is:  0.8292682926829268
```

```
Yes you're eligible for the loan
```