```
import numpy as np
import pandas as pd
df=pd.read_csv('/content/LoanApprovalPrediction.csv')
df
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Te
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360
...	...	...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360

614 rows × 13 columns

```
df.drop(['Loan_ID'],axis=1,inplace=True)
df
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_
0	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	
...	...	...	...	...	...	...	...	...	...	
609	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	
610	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	
611	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	
612	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	
613	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	

614 rows × 12 columns

```
df.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_H
0	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	

```
df.tail()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_
609	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	
610	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	
611	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	
612	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	
613	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	

```
df.dtypes
```

```
Gender             object
Married            object
Dependents         object
Education          object
Self_Employed      object
ApplicantIncome     int64
CoapplicantIncome  float64
LoanAmount         float64
Loan_Amount_Term   float64
Credit_History     float64
Property_Area       object
Loan_Status         object
dtype: object
```

```
df.isna().sum()
```

```
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
```

```
df['Gender']=df['Gender'].fillna(df['Gender'].mode()[0])#Single line filling
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])
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())
df.isna().sum()
```

```
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
```

```
from sklearn.preprocessing import LabelEncoder
encode=LabelEncoder()
df['Gender']=encode.fit_transform(df['Gender'])#since its a dataframe colomn namedf['Gender]


df['Married']=encode.fit_transform(df['Married'])


df['Dependents']=encode.fit_transform(df['Dependents'])


df['Education']=encode.fit_transform(df['Education'])
```

```python
df['Self_Employed']=encode.fit_transform(df['Self_Employed'])
```

```python
df['Property_Area']=encode.fit_transform(df['Property_Area'])
```

```python
df['Loan_Status']=encode.fit_transform(df['Loan_Status'])
df.dtypes
```

```
Gender                int64
Married               int64
Dependents            int64
Education             int64
Self_Employed         int64
ApplicantIncome       int64
CoapplicantIncome     float64
LoanAmount            float64
Loan_Amount_Term      float64
Credit_History        float64
Property_Area         int64
Loan_Status           int64
dtype: object
```

```python
x=df.iloc[:,:-1].values
x
```

```
array([[  1.,    0.,    0., ...,  360.,    1.,    2.],
       [  1.,    1.,    1., ...,  360.,    1.,    0.],
       [  1.,    1.,    0., ...,  360.,    1.,    2.],
       ...,
       [  1.,    1.,    1., ...,  360.,    1.,    2.],
       [  1.,    1.,    2., ...,  360.,    1.,    2.],
       [  0.,    0.,    0., ...,  360.,    0.,    1.]])
```

```python
y=df.iloc[:,-1].values
y
```

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

```python
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=0)
x_train
```

```
array([[  1.,    0.,    0., ...,  360.,    1.,    0.],
       [  1.,    1.,    0., ...,  180.,    1.,    0.],
       [  1.,    1.,    0., ...,  360.,    1.,    1.],
       ...,
       [  1.,    1.,    4., ...,  360.,    1.,    1.],
       [  1.,    1.,    0., ...,  360.,    1.,    2.],
       [  0.,    1.,    0., ...,  360.,    1.,    1.]])
```

```python
y_train
```

```
array([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, 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])
```

```python
from sklearn.preprocessing import StandardScaler
scale=StandardScaler()
scale.fit(x_train)
x_train=scale.transform(x_train)
x_test=scale.transform(x_test)
```

```python
from sklearn.naive_bayes import BernoulliNB
base=BernoulliNB()
base.fit(x_train,y_train)
y_pred=base.predict(x_test)
y_pred
```

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

```python
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[ 21  30]
 [  2 132]]
```

```python
score=accuracy_score(y_test,y_pred)
score
```

```
0.827027027027027
```