IMPORTING THE DEPENDENCIES

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

DATA COLLECTION AND PROCESSING

In [3]:

```
#LOAD THE DATA FROM CSV FILE TO PANDAS DATAFRAME
titanic_data=pd.read_csv('tested.csv')
```

In [9]:

```
#printing the 5 rows of data frame
titanic_data.head()
```

Out[9]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN

In [11]:

```
#number of rows and column
titanic_data.shape
```

Out[11]:

```
(418, 12)
```

In [12]:

```
#getting some information about the data
titanic_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Survived     418 non-null    int64  
 2   Pclass       418 non-null    int64  
 3   Name         418 non-null    object 
 4   Sex          418 non-null    object 
 5   Age          332 non-null    float64
 6   SibSp        418 non-null    int64  
 7   Parch        418 non-null    int64  
 8   Ticket       418 non-null    object 
 9   Fare         417 non-null    float64
 10  Cabin        91 non-null     object 
 11  Embarked     418 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 39.3+ KB
```

In [13]:

```
#check the number of missing values in each column
titanic_data.isnull().sum()
```

Out[13]:

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

HANDLING THE MISSING VALUES

In [17]:

```
#DROP THE "CABIN" COLUMN FROM THE DATASET
titanic_data=titanic_data.drop(columns='Cabin',axis=1)
```

In [20]:

```
#replacing the missing values in 'age 'column with mean
titanic_data['Age'].fillna(titanic_data['Age'].mean(),inplace=True)
```

In [21]:

```
#finding the mode value of 'fare'
print(titanic_data['Fare'].mode())
```

```
0    7.75
Name: Fare, dtype: float64
```

In [23]:

```
print(titanic_data['Fare'].mode()[0])
```

```
7.75
```

In [28]:

```
#replacing the missing value in 'Fare' column with mode value
titanic_data['Fare'].fillna(titanic_data['Fare'].mode()[0],inplace=True)
```

In [30]:

```
titanic_data.isnull().sum()
```

Out[30]:

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

DATA ANALYSIS

In [31]:

```
#getting some statistocal measures about the data
titanic_data.describe()
```

Out[31]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	418.000000	418.000000	418.000000	418.000000	418.000000
mean	1100.500000	0.363636	2.265550	30.272590	0.447368	0.392344	35.560497
std	120.810458	0.481622	0.841838	12.634534	0.896760	0.981429	55.857145
min	892.000000	0.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	996.250000	0.000000	1.000000	23.000000	0.000000	0.000000	7.895800
50%	1100.500000	0.000000	3.000000	30.272590	0.000000	0.000000	14.454200
75%	1204.750000	1.000000	3.000000	35.750000	1.000000	0.000000	31.471875
max	1309.000000	1.000000	3.000000	76.000000	8.000000	9.000000	512.329200

In [33]:

```
#finding the number of peaple survived or not survived
titanic_data['Survived'].value_counts()
```

Out[33]:

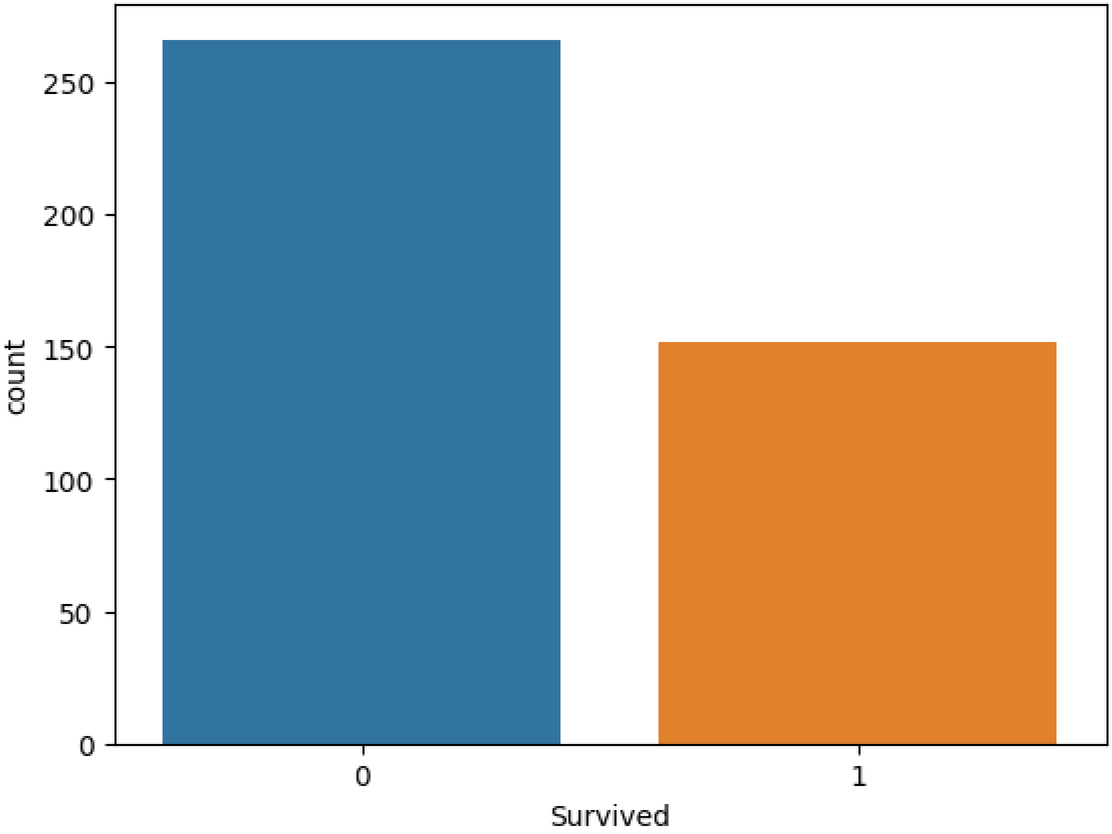
```
0    266
1    152
Name: Survived, dtype: int64
```

DATA VISUALIZATION

In [55]:

```
#MAKING A COUNT PLOT FOR 'SURVIVED' COLUMN
sns.countplot(x="Survived",data=titanic_data)
```

```
Out[55]: <Axes: xlabel='Survived', ylabel='count'>
```

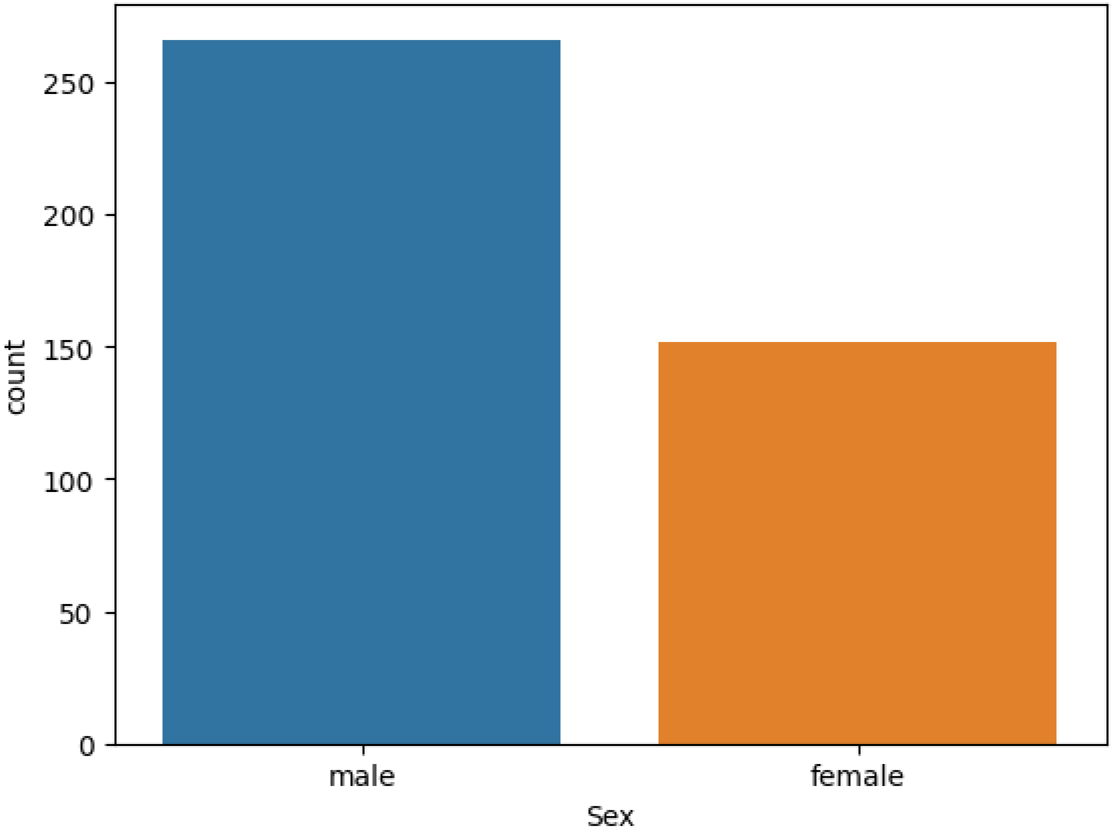


```
In [56]: titanic_data['Sex'].value_counts()
```

```
Out[56]: male      266
         female    152
         Name: Sex, dtype: int64
```

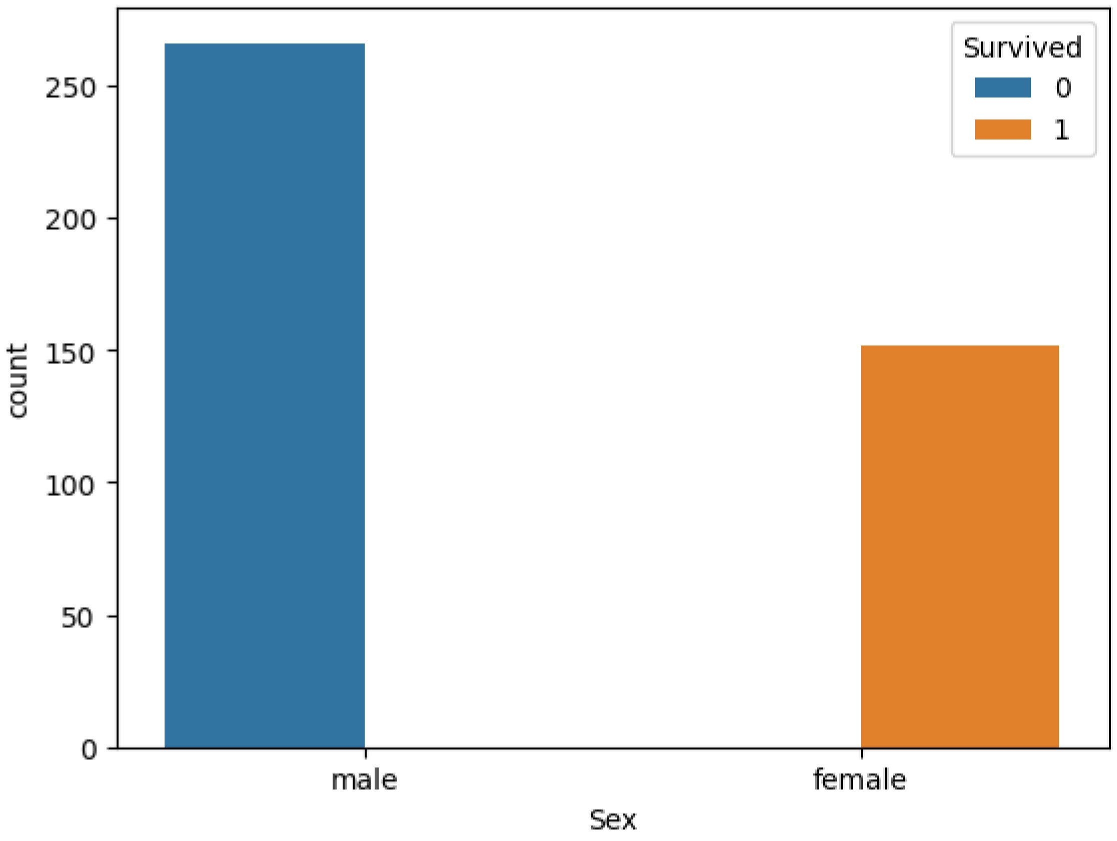
```
In [52]: ##MAKING A COUNT PLOT FOR 'SEX' COLUMN
         sns.countplot(x="Sex",data=titanic_data)
```

```
Out[52]: <Axes: xlabel='Sex', ylabel='count'>
```



```
In [72]: #number of survivers genderwise
         sns.countplot(x='Sex',hue='Survived',data=titanic_data)
```

```
Out[72]: <Axes: xlabel='Sex', ylabel='count'>
```

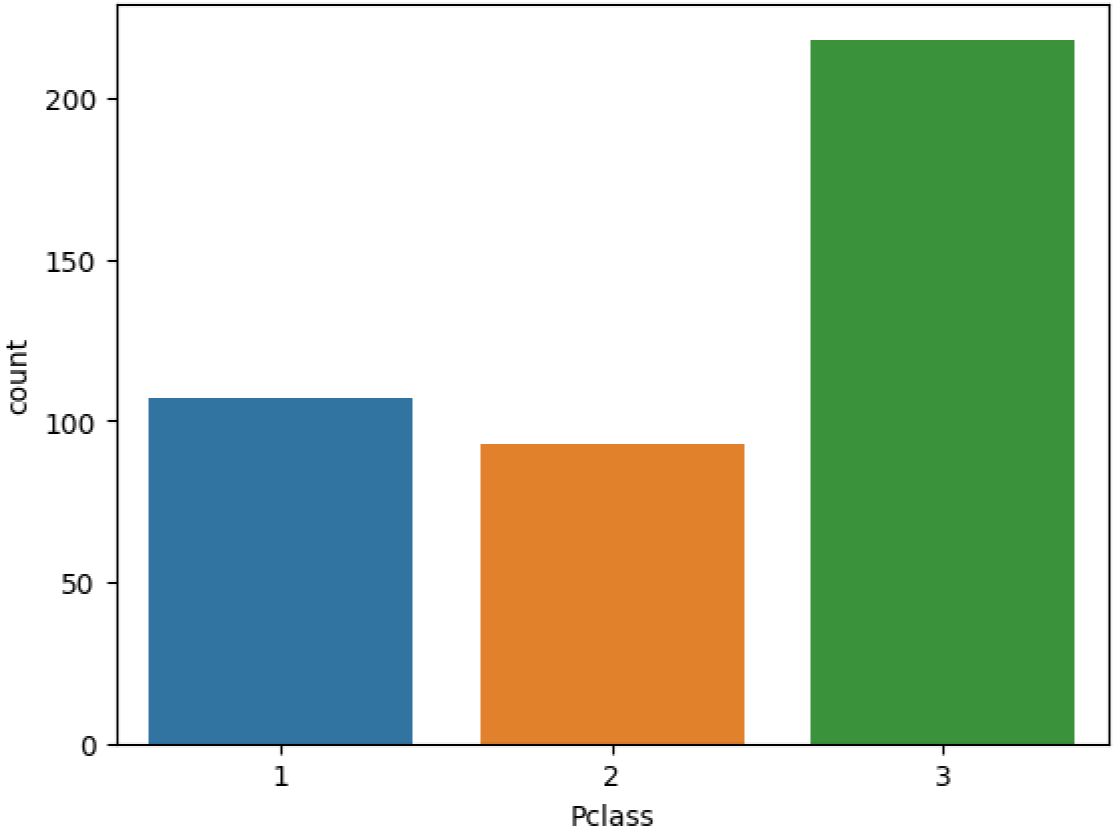


```
In [67]: titanic_data['Survived'].value_counts()
```

```
Out[67]: 0    266
         1    152
         Name: Survived, dtype: int64
```

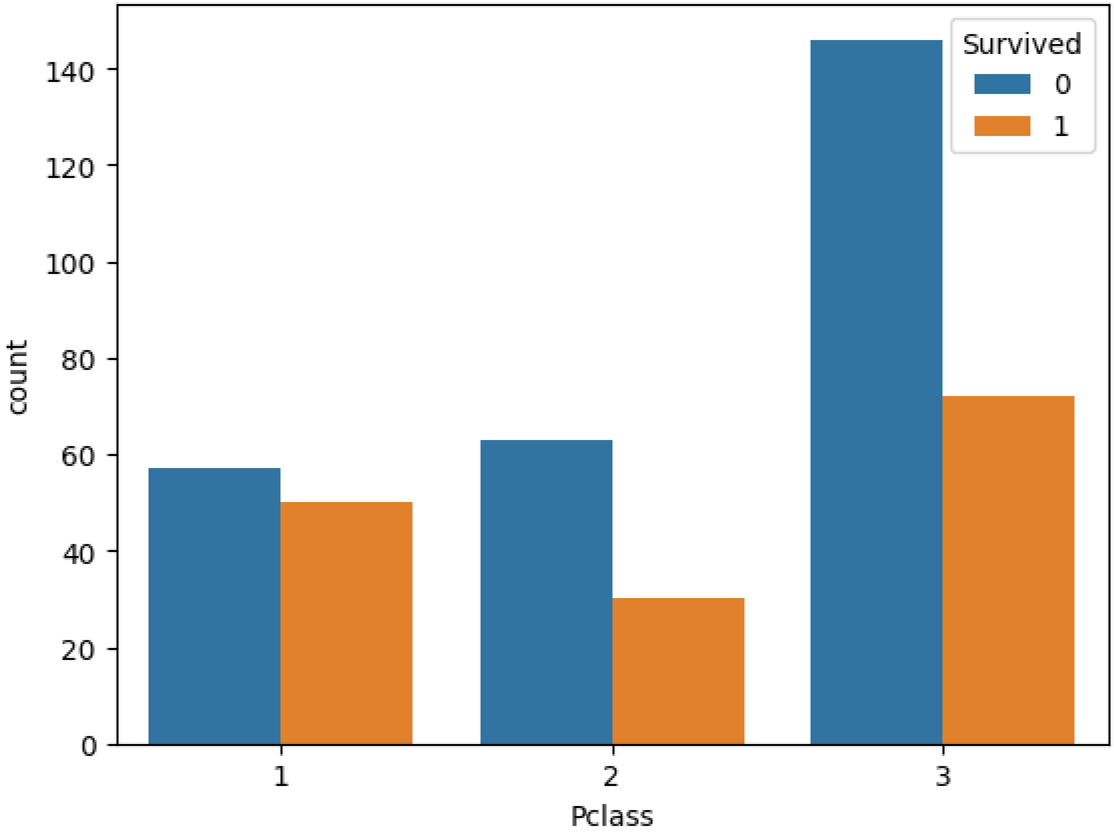
```
In [74]: #number of survivers 'pclass' column
         sns.countplot(x='Pclass',data=titanic_data)
```

```
Out[74]: <Axes: xlabel='Pclass', ylabel='count'>
```



```
In [75]: #number of survivers genderwise
         sns.countplot(x='Pclass',hue='Survived',data=titanic_data)
```

```
Out[75]: <Axes: xlabel='Pclass', ylabel='count'>
```



# encoding the categorical column

```
In [96]: #converting categorical columns
titanic_data['Sex'].replace('female',0,inplace=True)
titanic_data['Sex'].replace('male',1,inplace=True)
titanic_data['Embarked'].replace('S',0,inplace=True)
titanic_data['Embarked'].replace('C',1,inplace=True)
titanic_data['Embarked'].replace('Q',2,inplace=True)
```

```
In [97]: titanic_data.sample(5)
```

Out[97]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Er
416	1308	0	3	Ware, Mr. Frederick	1	30.27259	0	0	359309	8.0500	
272	1164	1	1	Clark, Mrs. Walter Miller (Virginia McDowell)	0	26.00000	1	0	13508	136.7792	
370	1262	0	2	Giles, Mr. Edgar	1	21.00000	1	0	28133	11.5000	
186	1078	1	2	Phillips, Miss. Alice Frances Louisa	0	21.00000	0	1	S.O./P.P. 2	21.0000	
180	1072	0	2	McCrie, Mr. James Matthew	1	30.00000	0	0	233478	13.0000	

SEPERATING FEATURES AND TARGET

In [101…

```
x=titanic_data.drop(columns=['Survived','PassengerId','Name','Ticket'],axis=1)
y=titanic_data['Survived']
```

In [103…

```
x
```

Out[103]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	1	34.50000	0	0	7.8292	2
1	3	0	47.00000	1	0	7.0000	0
2	2	1	62.00000	0	0	9.6875	2
3	3	1	27.00000	0	0	8.6625	0
4	3	0	22.00000	1	1	12.2875	0
...	...	...	...	...	...	...	...
413	3	1	30.27259	0	0	8.0500	0
414	1	0	39.00000	0	0	108.9000	1
415	3	1	38.50000	0	0	7.2500	0
416	3	1	30.27259	0	0	8.0500	0
417	3	1	30.27259	1	1	22.3583	1

418 rows × 7 columns

In [104…

```
y
```

Out[104]:

```
0      0
1      1
2      0
3      0
4      1
      ..
413    0
414    1
415    0
416    0
417    0
Name: Survived, Length: 418, dtype: int64
```

SPLTING DATA INTO TRAINING DATA AND TEST DATA

In [108…

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
```

In [109…

```
print(x.shape,x_train.shape,x_test.shape)
```

```
(418, 7) (334, 7) (84, 7)
```

MODEL TRAINING

In [112…

```
#Logistic Regression
model=LogisticRegression()
```

In [113…

```
#training the logistic regression model with training data
model.fit(x_tarin,y_train)
```

```
C:\Users\pcc\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\lin
ear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=
1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[113]:

```
▾ LogisticRegression

LogisticRegression()
```

In [115…

```
#model evaluation
#Accuracy score
```

```
#accuracy and training data
x_train_prediction=model.predict(x_train)
```

In [ ]:

```
```

In [116…

```
print(x_train_prediction)
```

```
[1 1 0 0 1 1 0 0 0 1 0 0 1 0 0 0 1 0 1 0 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0
 1 1 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 1 1 1 0 1
 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 1 0 0 0 0 0 0 0 1 0 1 1 1 0 1 0 1 0
 1 1 0 0 0 0 1 1 0 1 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 0 1 1 0 0
 0 0 1 1 1 0 0 1 1 0 1 1 0 0 0 0 0 0 0 1 1 0 0 1 1 1 1 0 1 0 0 0 0 1 0 1 1
 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 0 1 0 0 1 0 0
 1 0 1 0 0 0 0 0 1 0 0 0 1 1 0 0 0 1 1 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1
 0 1 1 1 1 0 0 0 1 1 0 0 1 0 1 1 0 0 0 0 1 0 0 0 0 0 1 0 0 1 1 0 1 1 0 0 0
 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 1 0 0 0 1 1 1
 1]
```

In [119…

```
training_data_accuracy=accuracy_score(y_train,x_train_prediction)
print('accuracy score of training data:',training_data_accuracy)
```

```
accuracy score of training data: 1.0
```

In [120…

```
#accuracy and test data
x_test_prediction=model.predict(x_test)
```

In [121…

```
print(x_test_prediction)
```

```
[0 0 0 1 1 0 1 0 0 1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 1 0 1 0 0 1
 1 0 0 0 0 1 1 0 0 1 0 1 0 0 0 1 1 1 0 0 1 0 0 0 0 0 0 1 0 1 1 1 1 1 1 0 0
 0 1 1 0 1 0 0 0 0 0]
```

In [123…

```
test_data_accuracy=accuracy_score(y_test,x_test_prediction)
print('accuracy score of test data:',test_data_accuracy)
```

```
accuracy score of test data: 1.0
```