IMPORTING THE DEPENDENCIES

In [44]:
```
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 [45]:
```
titanic_data = pd.read_csv('titanic.csv')
```

In [46]:
```
titanic_data.head()
```

Out[46]:

	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 [47]:
```
titanic_data.shape
```

Out[47]:
```
(418, 12)
```

In [48]:
```
titanic_data.info
```

Out[48]:
```
<bound method DataFrame.info of      PassengerId  Survived  Pclass  \
0            892         0       3
1            893         1       3
2            894         0       2
3            895         0       3
4            896         1       3
..           ...       ...     ...
413         1305         0       3
414         1306         1       1
415         1307         0       3
416         1308         0       3
417         1309         0       3

                                          Name     Sex   Age  SibSp  Parch  \
0                              Kelly, Mr. James    male  34.5      0      0
1              Wilkes, Mrs. James (Ellen Needs)  female  47.0      1      0
2                     Myles, Mr. Thomas Francis    male  62.0      0      0
3                              Wirz, Mr. Albert    male  27.0      0      0
4    Hirvonen, Mrs. Alexander (Helga E Lindqvist)  female  22.0      1      1
..                                          ...     ...   ...    ...    ...
413                            Spector, Mr. Woolf    male   NaN      0      0
414                Oliva y Ocana, Dona. Fermina  female  39.0      0      0
415                Saether, Mr. Simon Sivertsen    male  38.5      0      0
416                         Ware, Mr. Frederick    male   NaN      0      0
417                     Peter, Master. Michael J    male   NaN      1      1

                Ticket      Fare Cabin Embarked
0               330911    7.8292   NaN        Q
1               363272    7.0000   NaN        S
2               240276    9.6875   NaN        Q
3               315154    8.6625   NaN        S
4              3101298   12.2875   NaN        S
..                 ...       ...   ...      ...
413          A.5. 3236    8.0500   NaN        S
414           PC 17758  108.9000  C105        C
415  SOTON/O.Q. 3101262    7.2500   NaN        S
416              359309    8.0500   NaN        S
417                2668   22.3583   NaN        C

[418 rows x 12 columns]>
```

In [49]:
```
#HANDLING MISSING VALUES
titanic_data.isnull().sum()
```

Out[49]:
```
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
```

In [50]:
```
titanic_data = titanic_data.drop(columns = 'Cabin',axis = 1)
```

In [51]:
```
titanic_data = titanic_data.drop(columns = 'Fare',axis = 1)
```

In [52]:
```
titanic_data['Age'].fillna(titanic_data['Age'].mean(), inplace=True)
```

In [53]:
```
titanic_data.isnull().sum()
```

Out[53]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Embarked       0
dtype: int64
```

DATA ANALYSIS

In [54]:
```
titanic_data.describe()
```

Out[54]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch
count	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
std	120.810458	0.481622	0.841838	12.634534	0.896760	0.981429
min	892.000000	0.000000	1.000000	0.170000	0.000000	0.000000
25%	996.250000	0.000000	1.000000	23.000000	0.000000	0.000000
50%	1100.500000	0.000000	3.000000	30.272590	0.000000	0.000000
75%	1204.750000	1.000000	3.000000	35.750000	1.000000	0.000000
max	1309.000000	1.000000	3.000000	76.000000	8.000000	9.000000

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

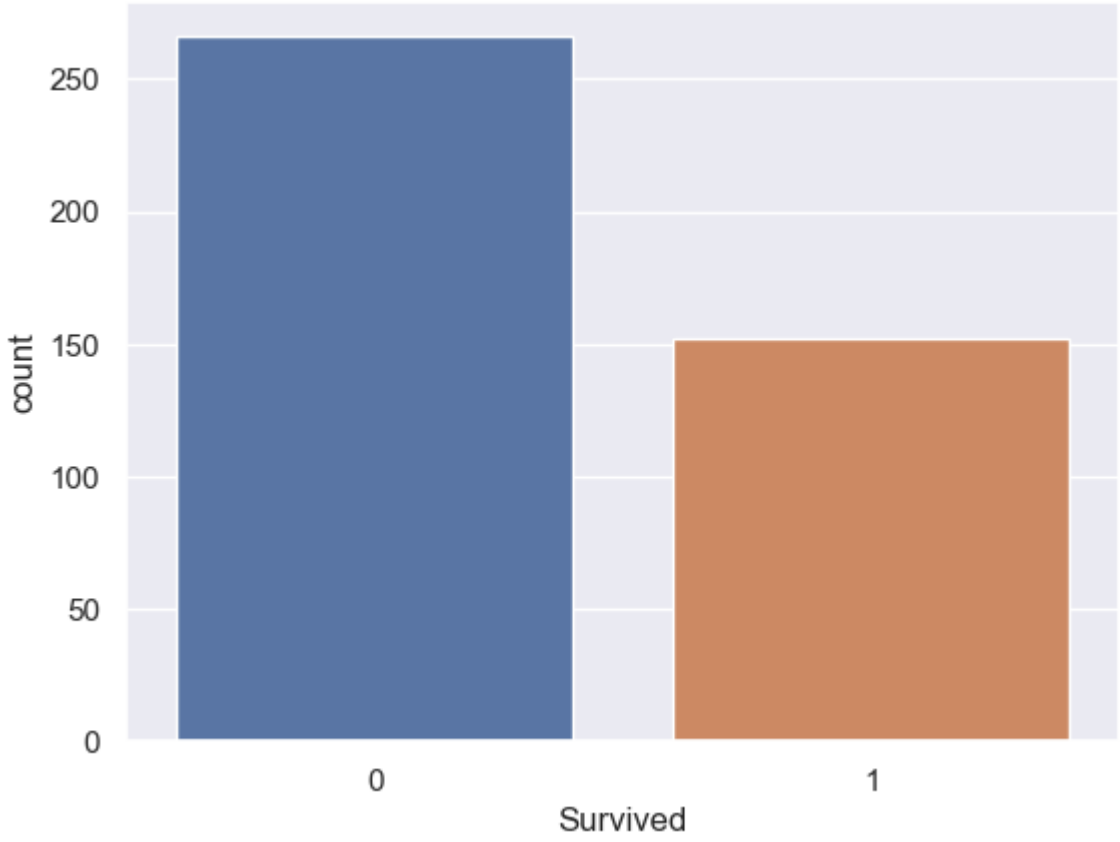
Out[56]:
```
0    266
1    152
Name: Survived, dtype: int64
```

DATA VISUALIZATION

In [57]:
```
sns.set()
```

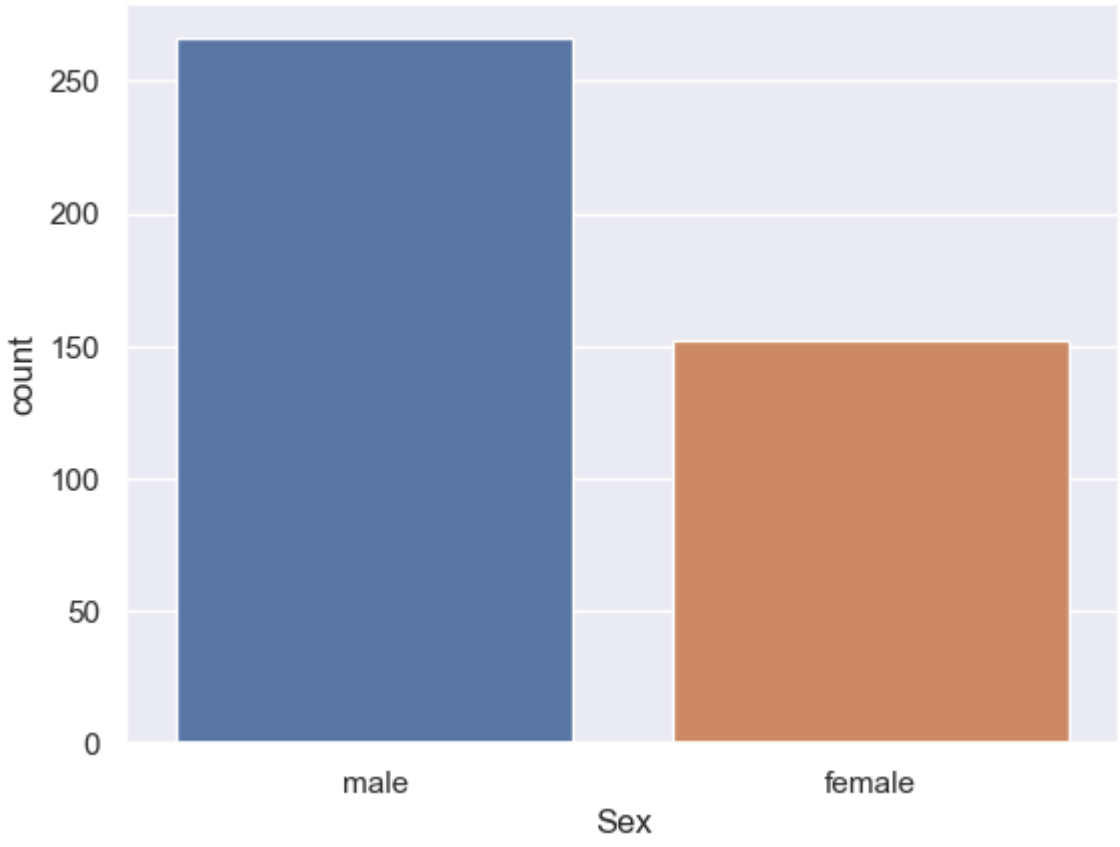
In [60]:
```
sns.countplot('Survived', data = titanic_data)
```

Out[60]:
```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```



In [62]:
```
sns.countplot('Sex', data = titanic_data)
```

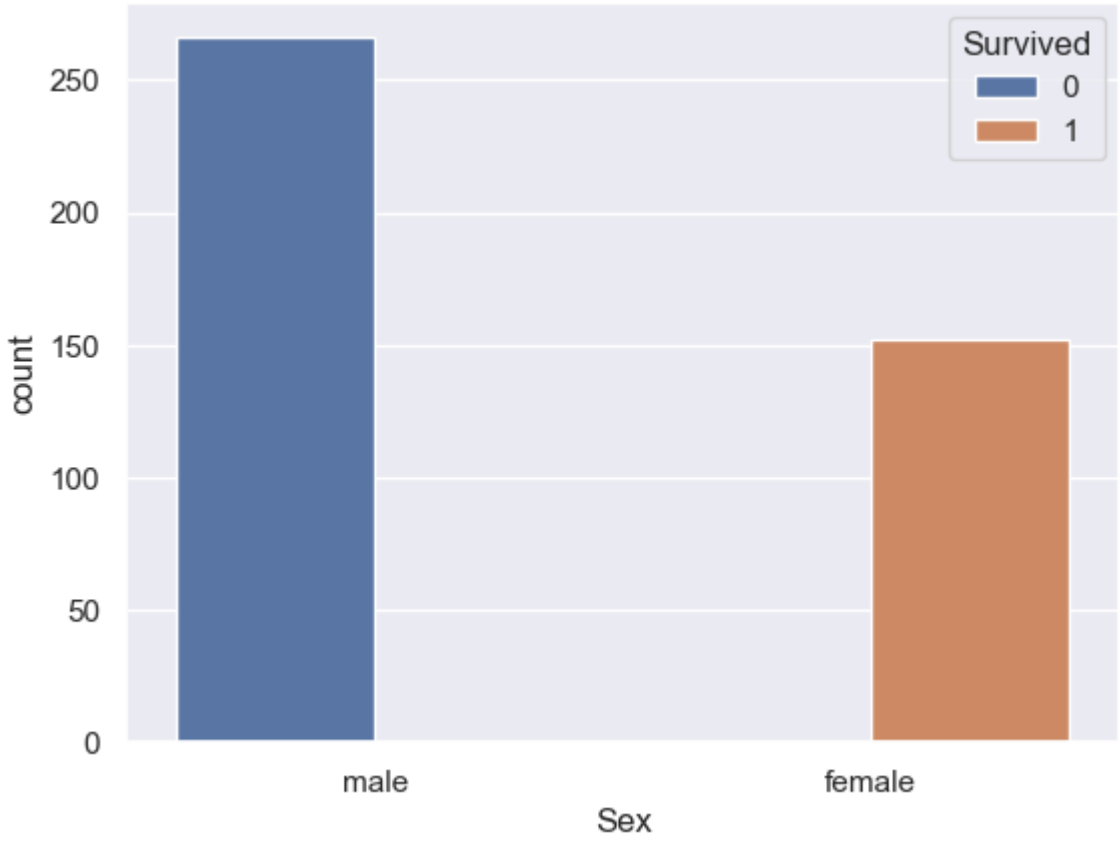
Out[62]:
```
<AxesSubplot:xlabel='Sex', ylabel='count'>
```



In [79]:
```
sns.countplot('Sex', hue = 'Survived', data = titanic_data)
```

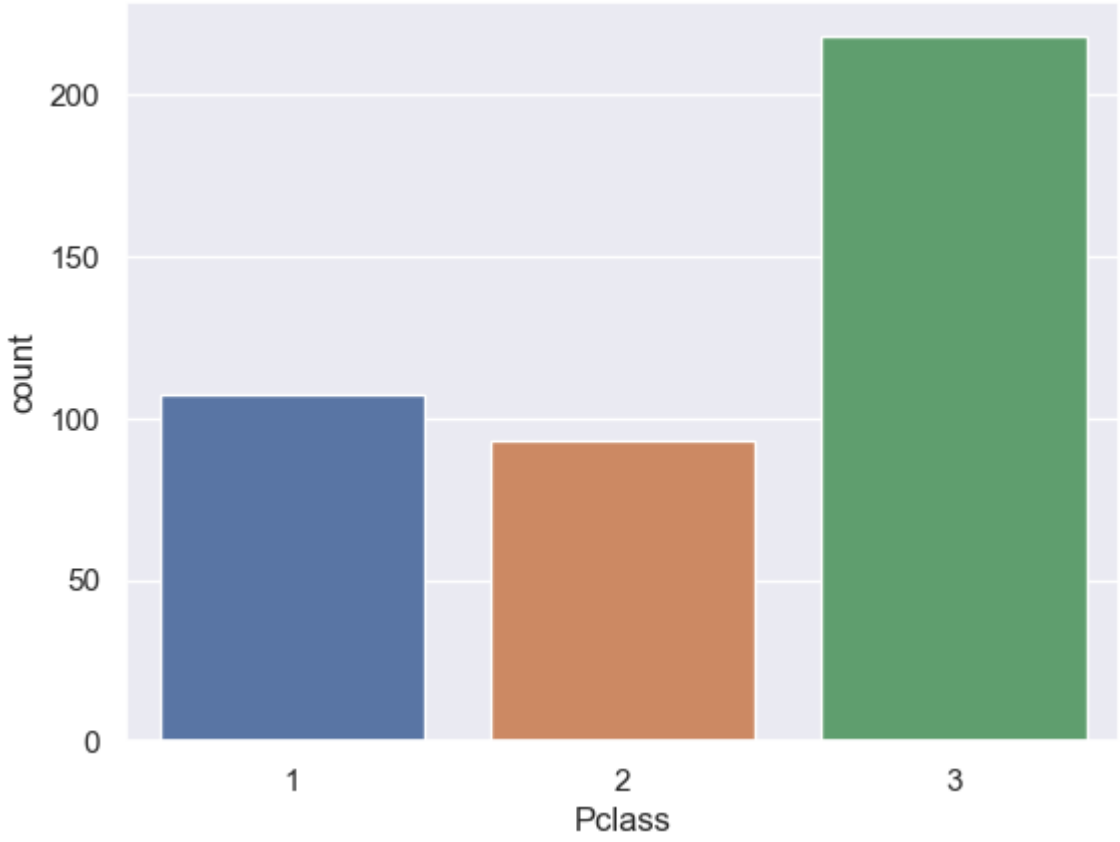
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.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

Out[79]:
```
<AxesSubplot:xlabel='Sex', ylabel='count'>
```



In [81]:
```
sns.countplot('Pclass', data = titanic_data)
```

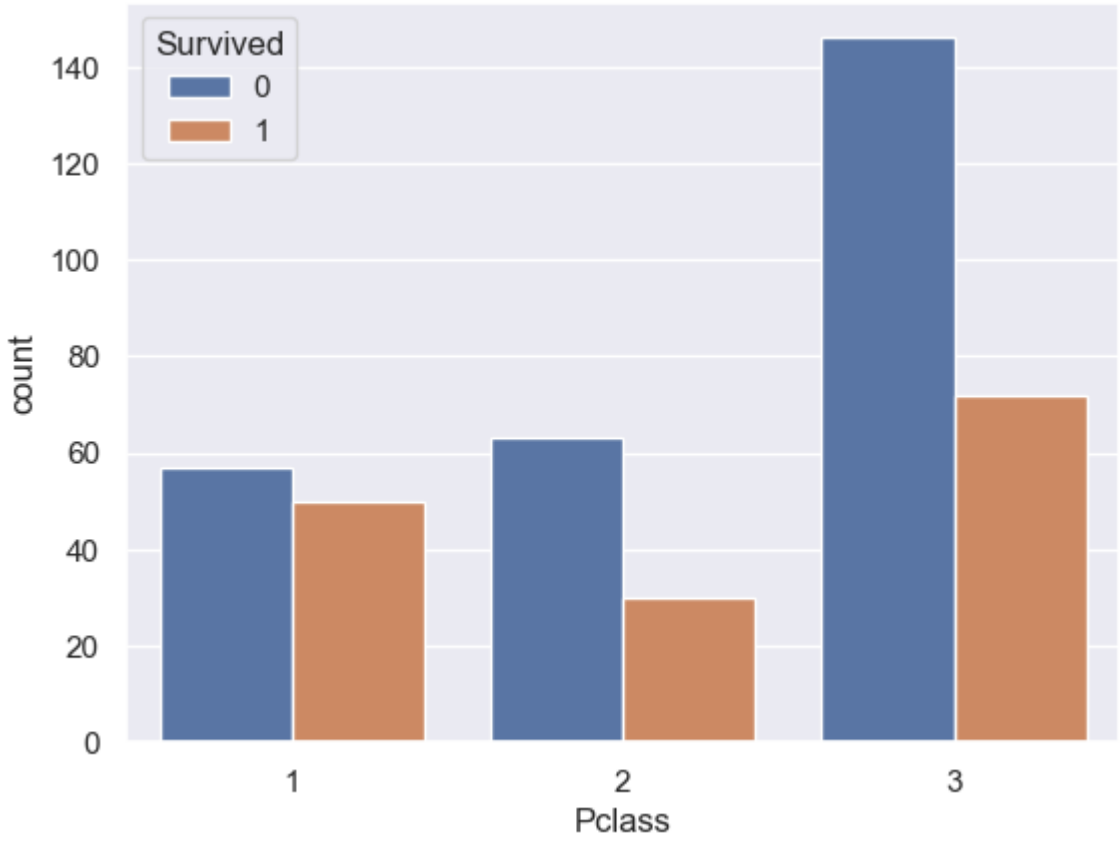
Out[81]:
```
<AxesSubplot:xlabel='Pclass', ylabel='count'>
```



In [83]:
```
sns.countplot('Pclass', hue = 'Survived', data = titanic_data)
```

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

Out[83]:
```
<AxesSubplot:xlabel='Pclass', ylabel='count'>
```



encoding the categorical columns

In [86]:
```
titanic_data['Sex'].value_counts()
```

Out[86]:
```
male      266
female    152
Name: Sex, dtype: int64
```

In [93]:
```
titanic_data['Embarked']
```

Out[93]:
```
0      2
1      0
2      2
3      0
4      0
      ..
413    0
414    1
415    0
416    0
417    1
Name: Embarked, Length: 418, dtype: int64
```

#converting categorical columns

In [91]:
```
titanic_data.replace({'Sex':{'male':0 ,'female':1},'Embarked':{'S':0,'C':1,'Q':2}},
```

#separating future and target

In [94]:
```
X = titanic_data.drop(columns = ['PassengerId','Name','Ticket','Survived'],axis=1)
Y = titanic_data['Survived']
```

In [98]:
```
print(X)
```

```
     Pclass  Sex       Age  SibSp  Parch  Embarked
0         3    0  34.50000      0      0         2
1         3    1  47.00000      1      0         0
2         2    0  62.00000      0      0         2
3         3    0  27.00000      0      0         0
4         3    1  22.00000      1      1         0
..      ...  ...       ...    ...    ...       ...
413       3    0  30.27259      0      0         0
414       1    1  39.00000      0      0         1
415       3    0  38.50000      0      0         0
416       3    0  30.27259      0      0         0
417       3    0  30.27259      1      1         1

[418 rows x 6 columns]
```

In [99]:
```
print(Y)
```

```
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
```

splitting Data into training and testing Data

In [101…
```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size= 0.2,random_state =
print(X.shape,X_train.shape,X_test.shape)
```

```
(418, 6) (334, 6) (84, 6)
```

MODEL TRAINING

In [105…
```
model = LogisticRegression()
```

In [106…
```
model.fit(X_train,Y_train)
```

Out[106]:
```
LogisticRegression()
```

model Evaluation on Accuracy Score

In [110…
```
X_train_prediction = model.predict(X_train)
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 0 1 0 1 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 [111…
```
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 [112…
```
X_test_prediction = model.predict(X_test)
```

In [113…
```
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 [114…

```
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
```