Assignment -3-Titanic Survival Prediction

January 31, 2024

Titanic Survival Predictions

Importing the Dependencies

[1]:
```
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
import warnings
warnings.filterwarnings('ignore')
```

Data Collection & preprocessing

[2]:
```
#Loading Data
train_data=pd.read_csv("C:\\Users\\prith\\Downloads\\train.csv")
```

[3]:
```
train_data.head()
```

[3]:
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
```

```
3      0             113803  53.1000  C123         S
4      0             373450   8.0500   NaN         S
```

[4]:
```python
# Number of Rows and Columns
train_data.shape
```

[4]:
```
(891, 12)
```

[5]:
```python
#Information abut the data
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

[6]:
```python
train_data.isnull().sum()
```

[6]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

Handling Missing values

[7]:
```
#Drop the "Cabin" Column from the dataframe
train_data=train_data.drop(columns='Cabin', axis=1)
```

[8]:
```
#Replacing the missing value in "Age" column with mean value
train_data['Age'].fillna(train_data['Age'].mean(),inplace=True)
```

[9]:
```
#Finding the Mode value of "Embraked" Column
print(train_data['Embarked'].mode())
```

```
0    S
Name: Embarked, dtype: object
```

[10]:
```
print(train_data['Embarked'].mode()[0])
```

```
S
```

[11]:
```
#Replacing the missing values in "Embracked Column with mode value"
train_data['Embarked'].fillna(train_data['Embarked'].mode()[0],inplace=True)
```

[12]:
```
#Check the number of missing values in each column
train_data.isnull().sum()
```

[12]:
```
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 Analytics

[13]:
```
#Getting some statistical measures about the data
train_data.describe()
```

[13]:
```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  891.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   13.002015    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   22.000000    0.000000
50%     446.000000    0.000000    3.000000   29.699118    0.000000
75%     668.500000    1.000000    3.000000   35.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000
```

```
            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

[14]:
```
#Finding the number of people survived and not survived
train_data['Survived'].value_counts()
```

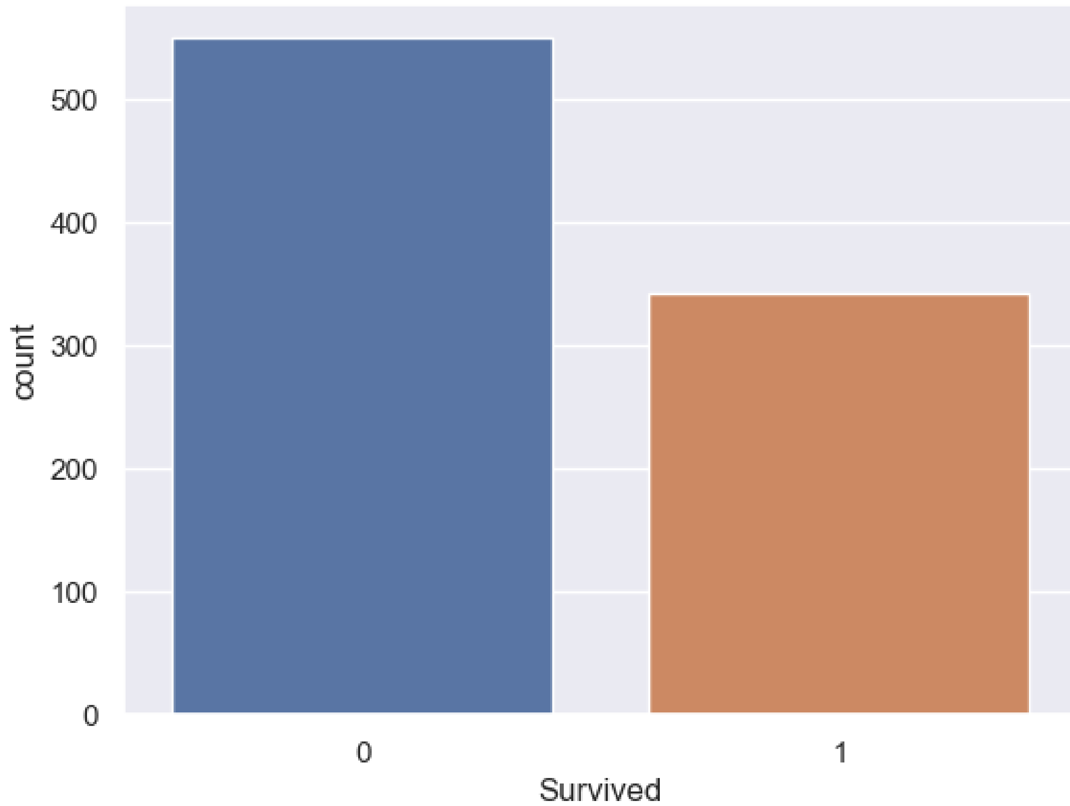
[14]:
```
0    549
1    342
Name: Survived, dtype: int64
```

Data Visualization

[15]:
```
sns.set()
```

[16]:
```
#Making a count plot for "Survived" column
sns.countplot("Survived", data=train_data)
```

[16]:
```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```

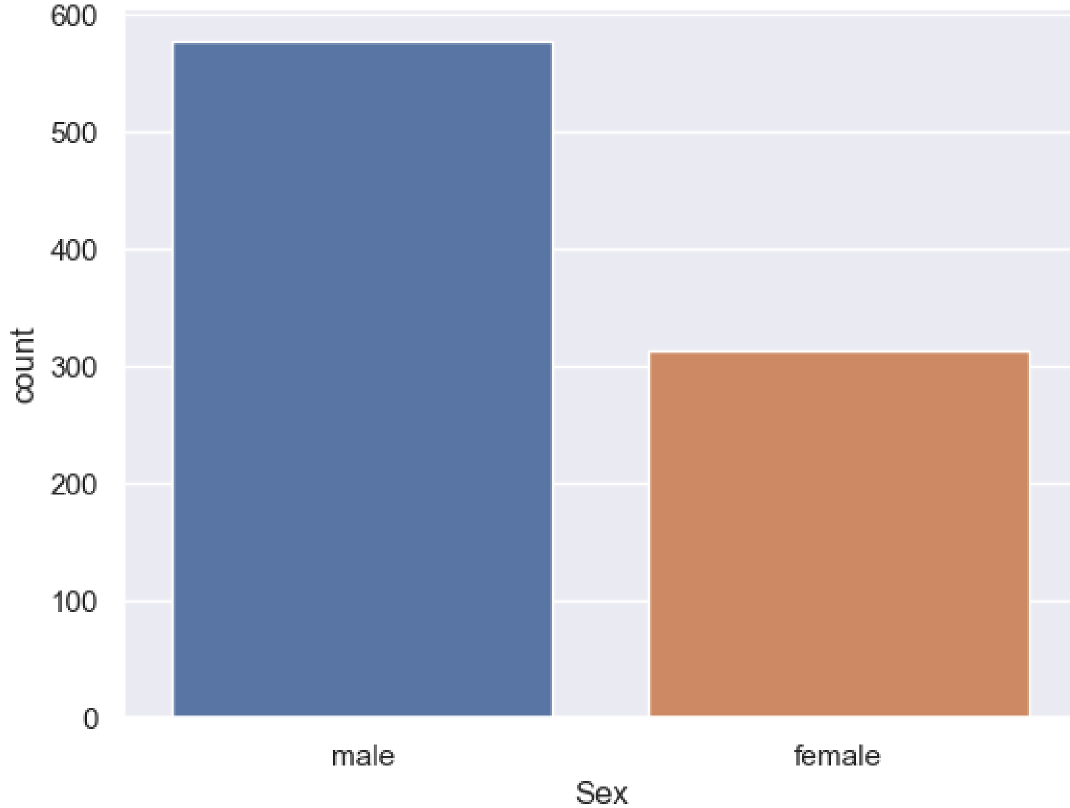


[17]:
```
#Finding the number of Male and Female
train_data['Sex'].value_counts()
```

[17]:
```
male      577
female    314
Name: Sex, dtype: int64
```

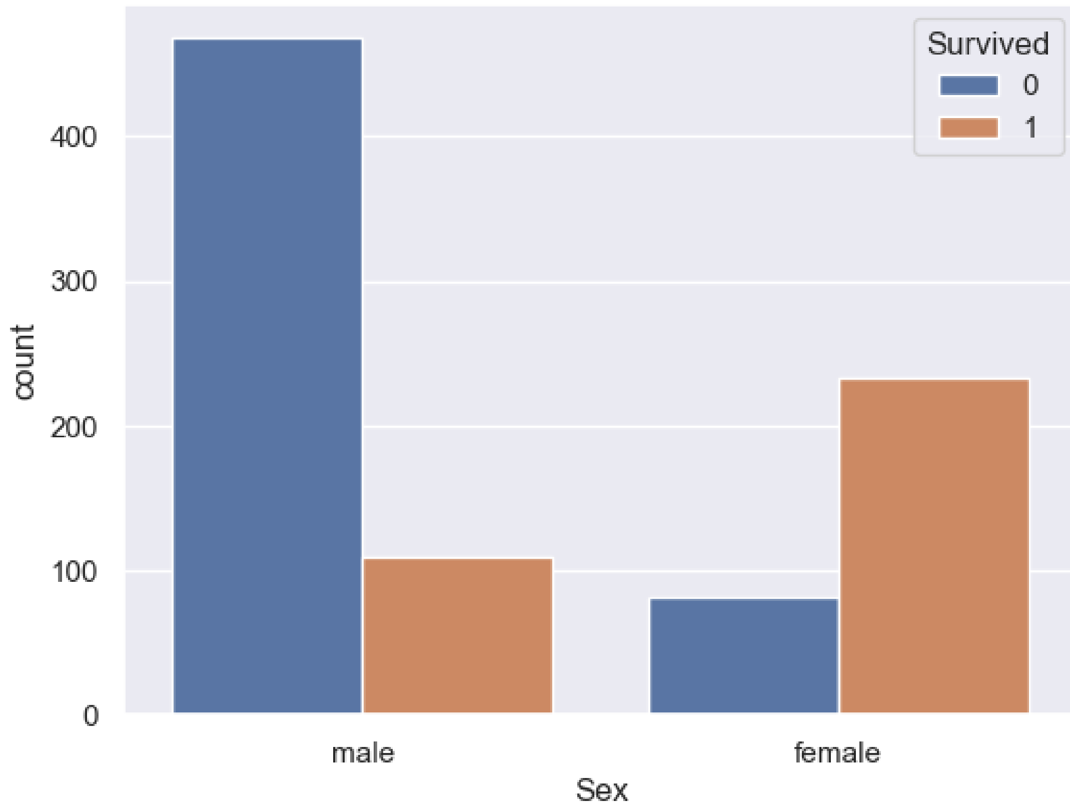
[18]:
```
#Making a count plot for "Survived" column
sns.countplot("Sex", data=train_data)
```

[18]: <AxesSubplot:xlabel='Sex', ylabel='count'>



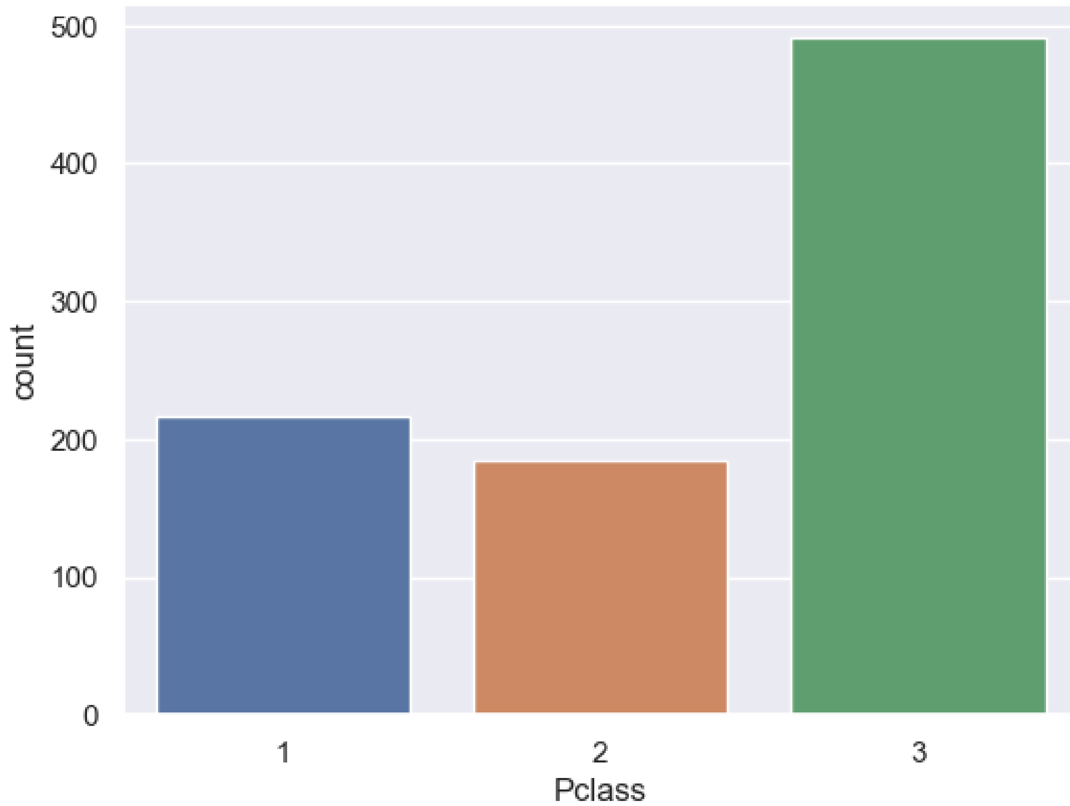
[19]:
```
#Number of Survivers Gender wise
sns.countplot('Sex',hue='Survived',data=train_data)
```

[19]: <AxesSubplot:xlabel='Sex', ylabel='count'>



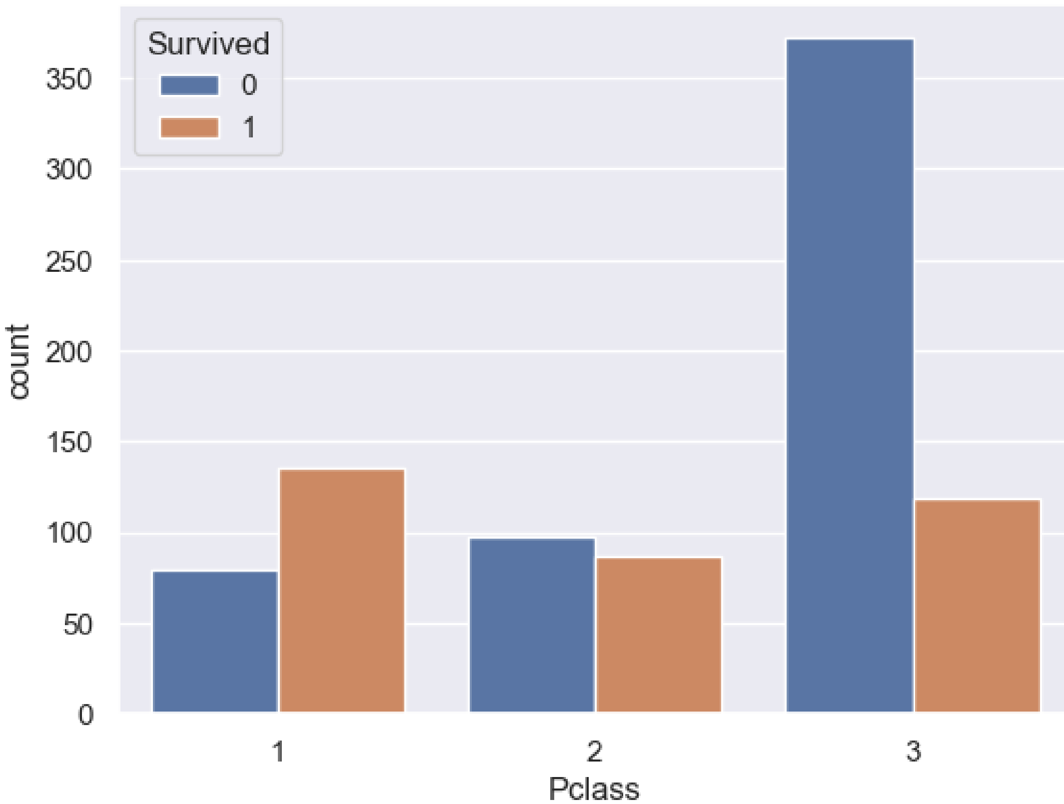
[20]:
```
#Making a count plot for "Pclass" column
sns.countplot("Pclass", data=train_data)
```

[20]: <AxesSubplot:xlabel='Pclass', ylabel='count'>



[21]:
```
sns.countplot('Pclass',hue='Survived',data=train_data)
```

[21]: <AxesSubplot:xlabel='Pclass', ylabel='count'>



Encolding the Categorical Columns

[22]:
```
train_data['Sex'].value_counts()
```

[22]:
```
male      577
female    314
Name: Sex, dtype: int64
```

[23]:
```
train_data['Embarked'].value_counts()
```

[23]:
```
S    646
C    168
Q     77
Name: Embarked, dtype: int64
```

[24]:
```
#Converting categorical columns
train_data.replace({'Sex':{'male':0,'female':1},'Embarked':{'S':0,'C':1,'Q':
 ↪2}},inplace=True)
```

[25]:
```
train_data.head()
```

[25]:
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name  Sex   Age  SibSp  Parch  \
0                            Braund, Mr. Owen Harris    0  22.0      1      0
1  Cumings, Mrs. John Bradley (Florence Briggs Th…    1  38.0      1      0
2                             Heikkinen, Miss. Laina    1  26.0      0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)    1  35.0      1      0
4                           Allen, Mr. William Henry    0  35.0      0      0

             Ticket     Fare  Embarked
0         A/5 21171   7.2500         0
1          PC 17599  71.2833         1
2  STON/O2. 3101282   7.9250         0
3            113803  53.1000         0
4            373450   8.0500         0
```

Separating Features & Target

[26]:
```
X=train_data.drop(columns=['PassengerId','Name','Ticket','Survived'],axis=1)
Y=train_data['Survived']
```

[27]:
```
print(X)
```

```
     Pclass  Sex        Age  SibSp  Parch     Fare  Embarked
0         3    0  22.000000      1      0   7.2500         0
1         1    1  38.000000      1      0  71.2833         1
2         3    1  26.000000      0      0   7.9250         0
3         1    1  35.000000      1      0  53.1000         0
4         3    0  35.000000      0      0   8.0500         0
..      …   …        …    …    …      …       …
886       2    0  27.000000      0      0  13.0000         0
887       1    1  19.000000      0      0  30.0000         0
888       3    1  29.699118      1      2  23.4500         0
889       1    0  26.000000      0      0  30.0000         1
890       3    0  32.000000      0      0   7.7500         2

[891 rows x 7 columns]
```

[28]:
```
print(Y)
```

```
0      0
1      1
2      1
3      1
```

```
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

Splitting the data into training & Test Data

[29]:
```python
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=2)
```

[30]:
```python
print(X.shape,X_train.shape,X_test.shape)
```

```
(891, 7) (712, 7) (179, 7)
```

Model Training

Logestic Regression

[31]:
```python
model=LogisticRegression()
```

[32]:
```python
#Training the logistic Regression Model with training data
model.fit(X_train,Y_train)
```

[32]:
```
LogisticRegression()
```

Model Evaluation

Accuracy Score

[33]:
```python
#Accuracy on training data
X_train_prediction=model.predict(X_train)
```

[34]:
```python
print(X_train_prediction)
```

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

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

[35]:
```python
training_data_accuracy=accuracy_score(Y_train,X_train_prediction)
print("Accuracy score of Training Data : ",training_data_accuracy)
```

```
Accuracy score of Training Data :  0.8075842696629213
```

[36]:
```python
#Accuracy on test data
X_test_prediction=model.predict(X_test)
```

[37]:
```python
print(X_test_prediction)
```

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

[38]:
```python
test_data_accuracy=accuracy_score(Y_test,X_test_prediction)
print("Accuracy score of test data : ", test_data_accuracy)
```

```
Accuracy score of test data :  0.7821229050279329
```