In [1]:
```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib
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 & Processing

In [1]:
```
#load the data from csv file to pandas dataframe
titanic_data = pd.read_csv('tested.csv')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call las
t)
Cell In[1], line 2
      1 #load the data from csv file to pandas dataframe
----> 2 titanic_data = pd.read_csv('tested.csv')

NameError: name 'pd' is not defined
```

In [4]:
```
# printing the first five rows of dataframe
titanic_data.head()
```

Out[4]:

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

In [5]:
```
# number of rows and columns
titanic_data.shape
```

Out[5]:
```
(418, 12)
```

In [6]:
```
#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 [7]:
```
#check the number of missing values in each column
titanic_data.isnull().sum()
```

Out[7]:
```
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 [8]:
```
# drop the "Cabin" column from the dataframe
titanic_data = titanic_data.drop(columns = 'Cabin' , axis=1)
```

In [9]:
```
#replacing the missing values in "Age" column with mean value
titanic_data['Age'].fillna(titanic_data['Age'].mean() , inplace = True)
```

In [10]:
```
#replacing the missing values in "Fare" column with mean value

titanic_data['Fare'].fillna(titanic_data['Fare'].mean() , inplace = True)
```

In [11]:
```
#check the number of missing values in each column
titanic_data.isnull().sum()
```

Out[11]:
```
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 [12]:
```
#getting some statistical measures about the data
titanic_data.describe()
```

Out[12]:

	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	35.627188
std	120.810458	0.481622	0.841838	12.634534	0.896760	0.981429	55.840500
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	31.500000
max	1309.000000	1.000000	3.000000	76.000000	8.000000	9.000000	512.329200

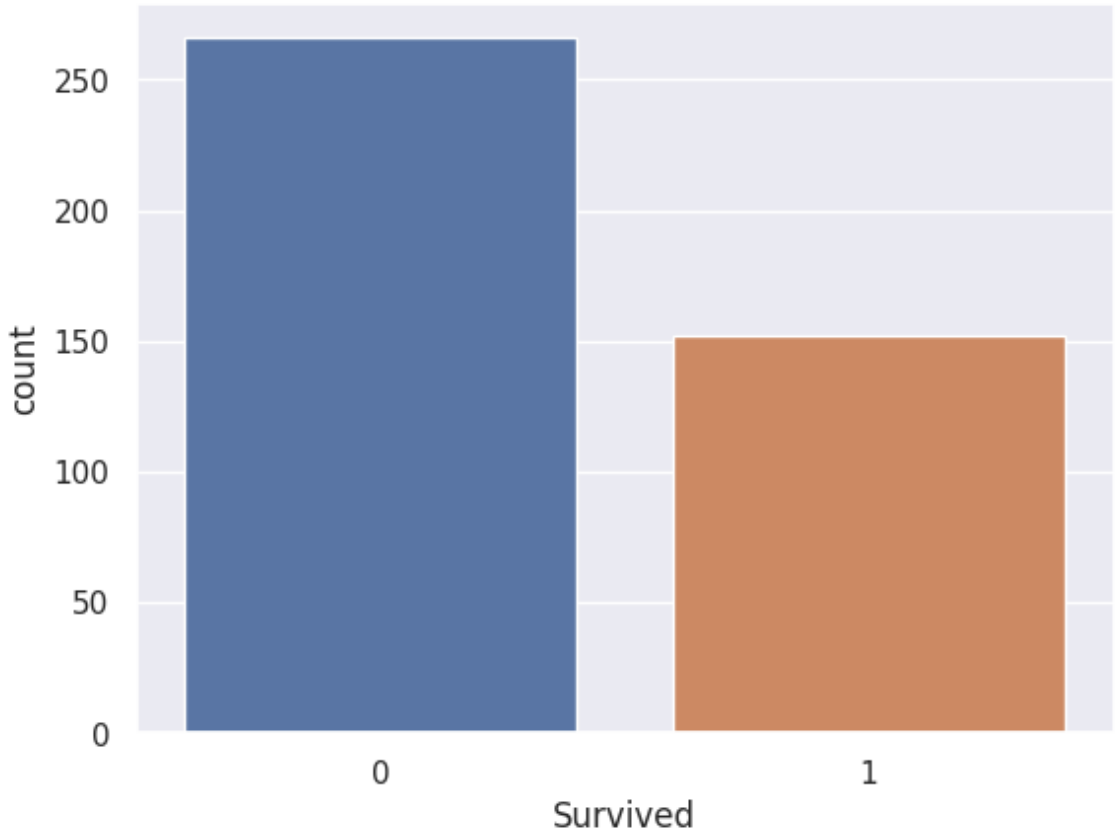
In [13]:
```
# finding the number of people survived and not survived
titanic_data['Survived'].value_counts()
```

Out[13]:
```
0    266
1    152
Name: Survived, dtype: int64
```

Data Visualization

In [14]:
```
sns.set()
```

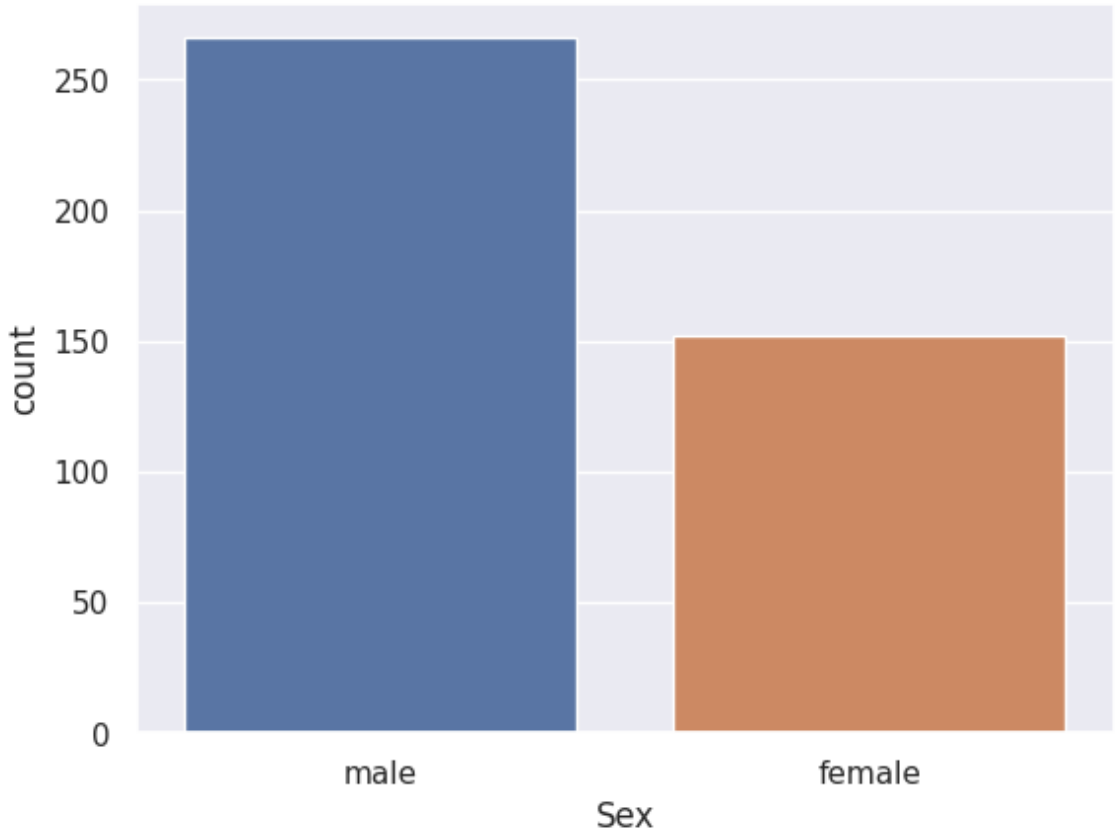
In [15]:
```
# making a count for "Survived" coloumn
sns.countplot(x='Survived', data=titanic_data)
plt.show()
```



In [16]:
```
#finding number of males and females
titanic_data['Sex'].value_counts()
```

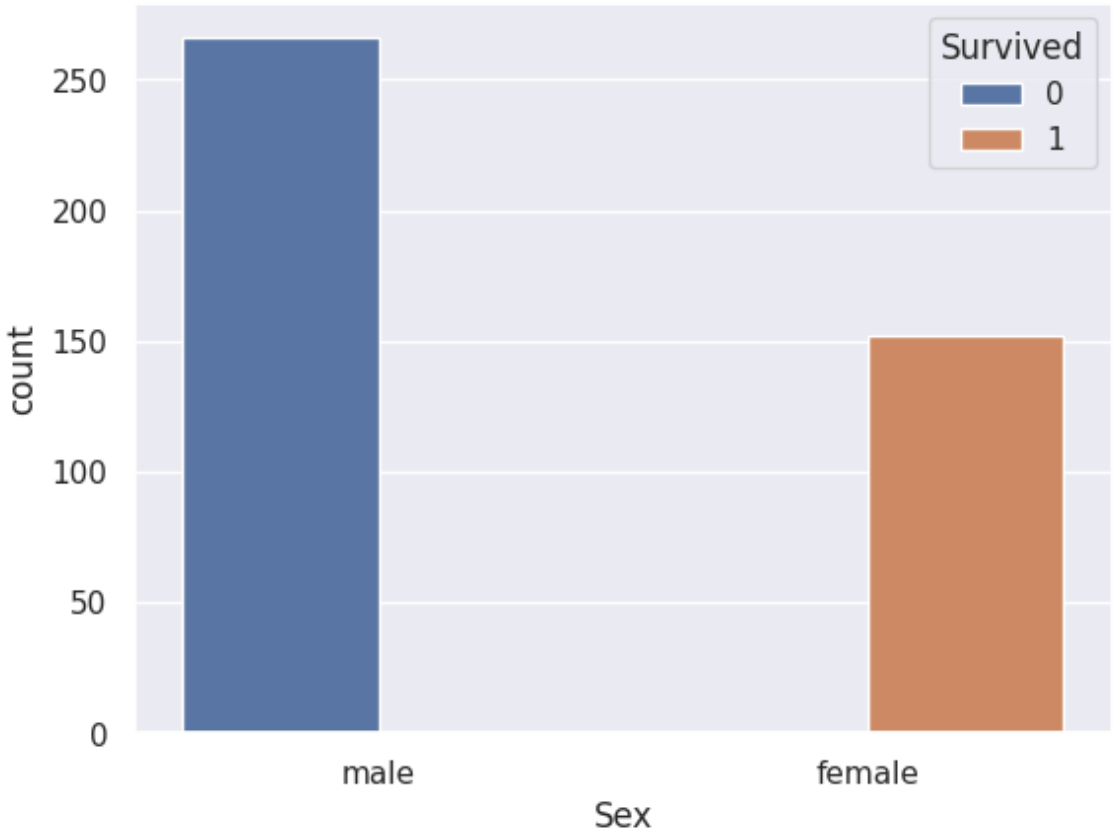
Out[16]:
```
male      266
female    152
Name: Sex, dtype: int64
```

In [17]:
```
# making a count for "Sex" coloumn
sns.countplot(x='Sex', data=titanic_data)
plt.show()
```

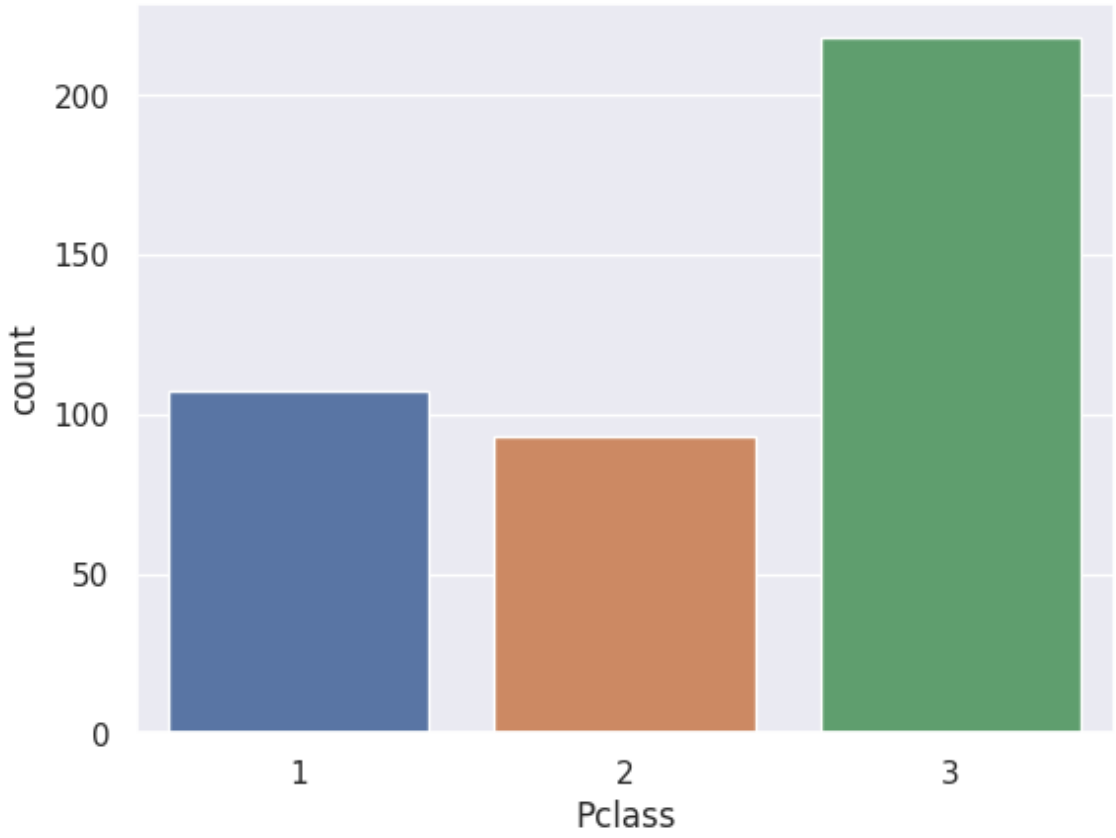


Type *Markdown* and LaTeX: α^2

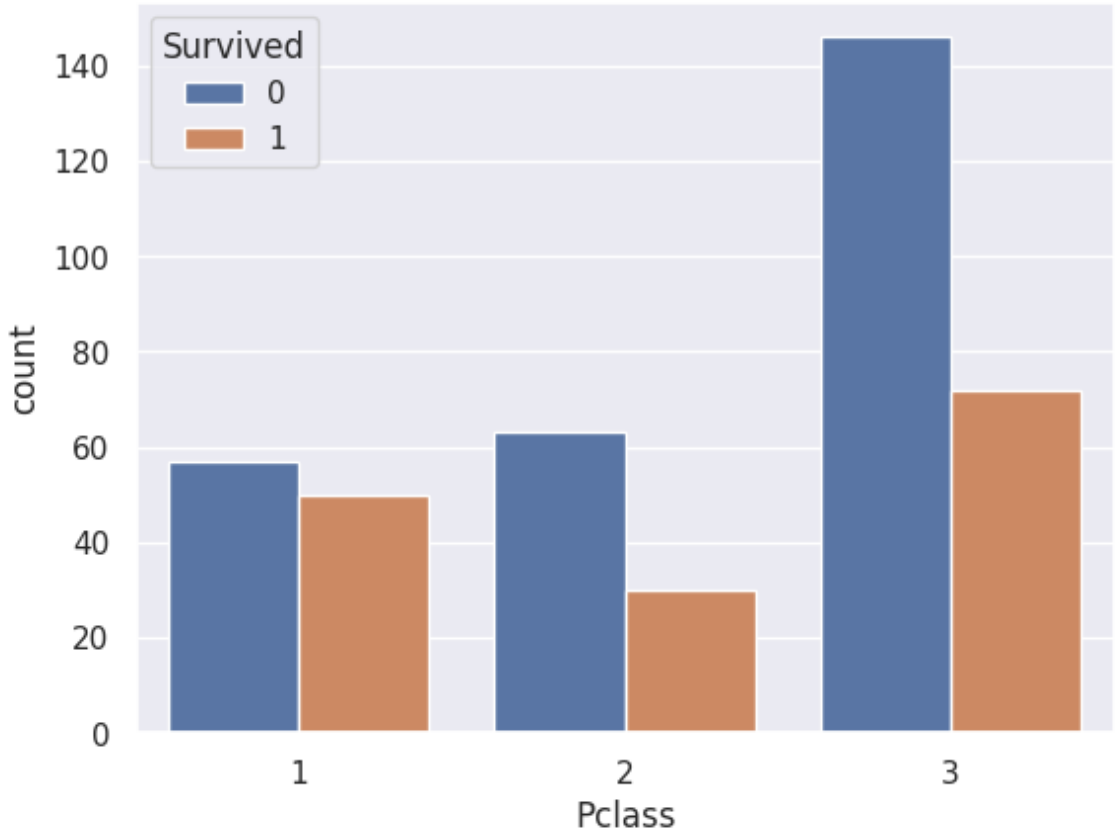
In [18]:
```
# Number of survivors based on genders
sns.countplot(x='Sex', hue='Survived', data=titanic_data)
plt.show()
```



In [19]:
```
# making a count for "PClass" coloumn
sns.countplot(x='Pclass', data=titanic_data)
plt.show()
```



In [20]:
```
sns.countplot(x='Pclass', hue='Survived', data=titanic_data)
plt.show()
```



Encoding categorical columns

In [21]:
```
titanic_data['Sex'].value_counts()
```

Out[21]:
```
male      266
female    152
Name: Sex, dtype: int64
```

In [22]:
```
titanic_data['Embarked'].value_counts()
```

Out[22]:
```
S    270
C    102
Q     46
Name: Embarked, dtype: int64
```

In [23]:
```
# converting categorical columns
titanic_data.replace({'Sex' : {'male':0 , 'female':1} , 'Embarked': {'S':0
```

In [24]:
```
titanic_data.head()
```

Out[24]:

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

Separating features and targets

In [25]:
```
x= titanic_data.drop(columns = ['PassengerId' , 'Name' , 'Ticket' , 'Surviv
y=titanic_data['Survived']
```

In [26]:
```
print(x)
```

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

[418 rows x 7 columns]
```

In [27]:
```
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 the data into training data and test data

In [28]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, ran
```

In [29]:
```
print(x.shape, x_train.shape, x_test.shape)
```

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

MODEL TRAINING

Logistic Regression

In [30]:
```
model = LogisticRegression()
```

In [31]:
```
#training the logistic regression model with training data
model.fit(x_train, y_train)
```

Out[31]:
```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Evaluating Model

Accuracy Score

In [32]:
```
#accuracy on the training data
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 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 1 0 0 0 1 1 0 1 1 0 0 0 1 1 1
 1]
```

In [33]:
```
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 [34]:
```
#accuracy on test data
x_test_prediction = model.predict(x_test)
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 [35]:
```
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
```