In [37]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
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')
```

In [38]:
```python
df=pd.read_csv("titanic.csv")
df
```

Out[38]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [39]:
```python
df.shape
```

Out[39]:
```
(891, 12)
```

We can observe that the dataset contains 891 rows and 12 columns

In [15]:
```python
df.nunique().to_frame("No. of unique frame")
```

Out[15]:

	No. of unique frame
PassengerId	891
Survived	2
Pclass	3
Name	891
Sex	2
Age	88
SibSp	7
Parch	7
Ticket	681
Fare	248
Cabin	147
Embarked	3

We can observe that we have centrain unique values for each and every variable

In [13]:
```python
df.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
```

As we can observe there are certain missing values in age,cabin and embarked and the datatype are float, int and object

In [41]:
```python
df.isnull().sum()
```

Out[41]:
```
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
```

We have 177 missing values in age, 687 missing values in cabin and 2 missing values in embarked

In [42]:
```python
df1=df.drop(columns='Cabin',axis=1)        #handling the missing value
```

In [43]:
```python
df1
```

Out[43]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S
...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	Q

891 rows × 11 columns

We have dropped the Cabin column to handle the missing value as we have 687 missing value out of 891

In [44]:
```python
df1['Age'].fillna(df1['Age'].mean(), inplace=True)    #replacing the missing value in "Age" column with mean value
```

In [45]:
```python
df1
```

Out[45]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450	8.0500	S
...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	211536	13.0000	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	112053	30.0000	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W./C. 6607	23.4500	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	111369	30.0000	C
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	370376	7.7500	Q

891 rows × 11 columns

We have replaced the missing data of Age with mean value to fill the missing values

In [46]:
```python
print(df1['Embarked'].mode())  #finding the mode value of "Embarked" column
```

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

As we can observe that S is the most repititive embarked value so we will replace the 2 missing value with S as we can not find the mean value of categorical data, so we will be using the mode function to handle the missing value

In [47]:
```python
print(df1['Embarked'].mode()[0])
```

```
S
```

In [48]:
```python
df1['Embarked'].fillna(df1['Embarked'].mode()[0], inplace=True)  #replacing the missing value in embarked column with mode value
```

We have successfully fill the "Embarked" column with the mode value. Now we will check for other missing values

In [50]:
```python
df1.isnull().sum()
```

Out[50]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

As we can observe that there is no more missing value present in the dataset

In [51]:
```python
df1.describe()
```

Out[51]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	13.002015	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	22.000000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	29.699118	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	35.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

This gives the statistical information of the numerical columns. The summery of the dataset looks perfectly fine as there is no negative value present in the dataset.

From the above description we can observe the following-

1. It is showing the result for only numerical columns not for categorical columns

1. The count of all the columns are same which means there are no missing value present in the dataset

1. The mean value is greater than the median value in all the columns except pclass

1. By summarizing the data we can observe there is a huge difference 75% and maximum in fair, passengerid and age and little difference in parce amd sibsp hence outliers are present in the data

1. We also get the standard deviation, min, 25th quartile and 75th quartile values from the describe method.

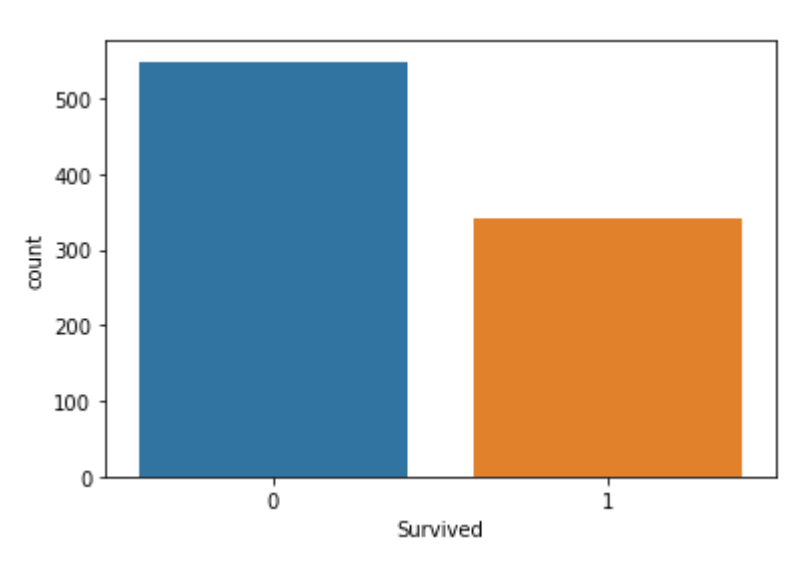
In [52]:
```python
df1['Survived'].value_counts()
```

Out[52]:
```
0    549
1    342
Name: Survived, dtype: int64
```

Ae we can observe out of 891 passeneger only 342 survived and 549 did not survive.

In [53]:
```python
sns.countplot('Survived',data=df1)
```

Out[53]: <AxesSubplot:xlabel='Survived', ylabel='count'>



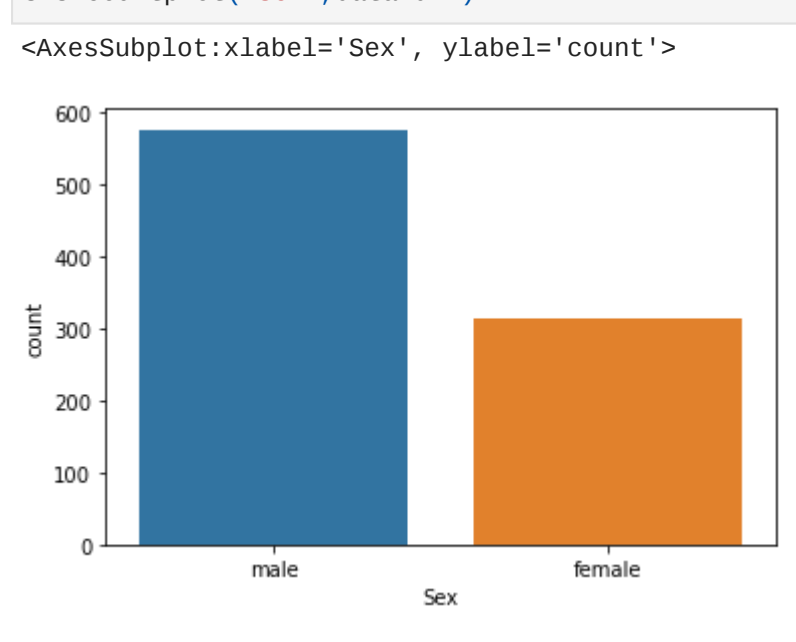
The visual representation of people who survived or not

In [55]:
```python
df1['Sex'].value_counts()
```

Out[55]:
```
male      577
female    314
Name: Sex, dtype: int64
```

In [54]:
```python
sns.countplot('Sex',data=df1)
```

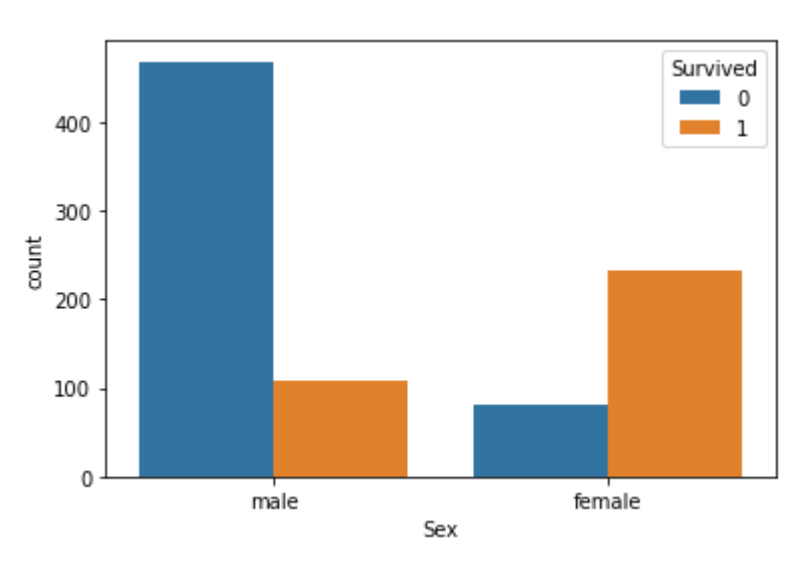
Out[54]: <AxesSubplot:xlabel='Sex', ylabel='count'>



As we can observe the number of male and number of female who were there

In [56]:
```python
sns.countplot(x='Sex', hue='Survived', data=df1)
```

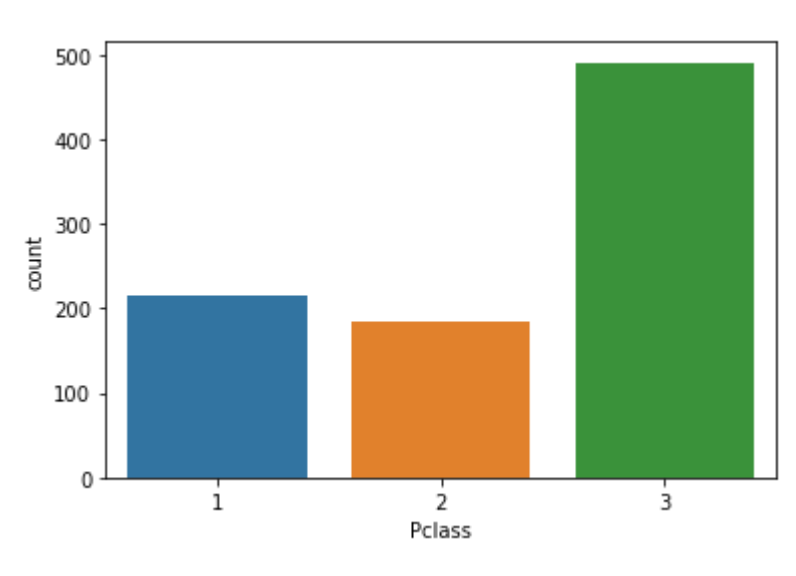
Out[56]: <AxesSubplot:xlabel='Sex', ylabel='count'>



As we can observe out of 577 males only around 100 have survived and out of 314 females only below 250 have survived

In [60]:
```python
sns.countplot('Pclass',data=df1)
```

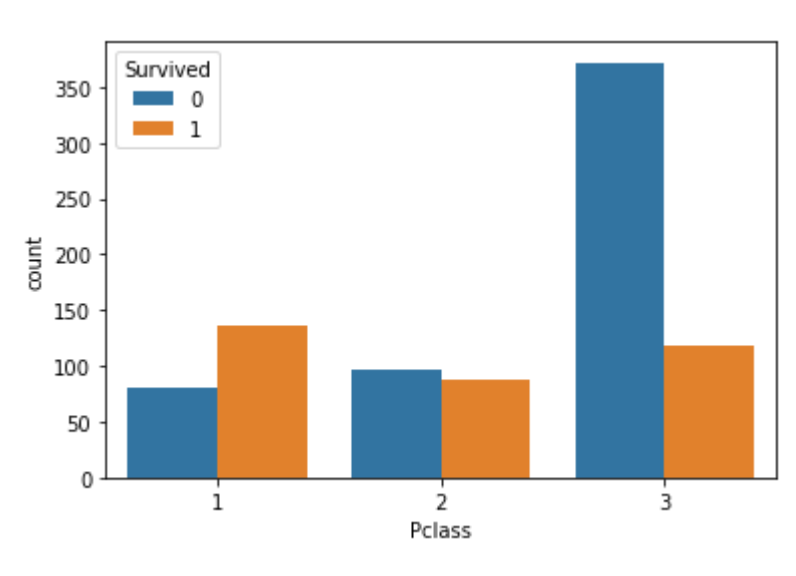
Out[60]: <AxesSubplot:xlabel='Pclass', ylabel='count'>



The three bars implies the ticket class of the Titanic

In [59]:
```python
sns.countplot('Pclass',hue='Survived',data=df1)
```

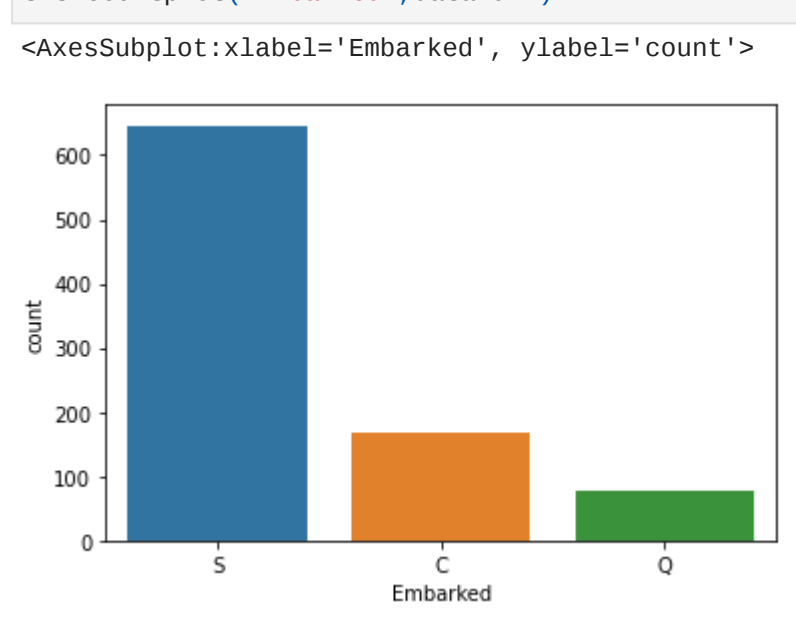
Out[59]: <AxesSubplot:xlabel='Pclass', ylabel='count'>



As we can observe the people who have survived based on the Pclass

In [61]:
```python
sns.countplot('Embarked',data=df1)
```

Out[61]: <AxesSubplot:xlabel='Embarked', ylabel='count'>



In [64]:
```python
df1.replace({'Sex':{'male':0,'female':1},'Embarked':{'S':0,'C':1,'Q':2}},inplace=True)
```

In [65]:
```python
df1
```

Out[65]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	0	22.000000	1	0	A/5 21171	7.2500	0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.000000	1	0	PC 17599	71.2833	1
2	3	1	3	Heikkinen, Miss. Laina	1	26.000000	0	0	STON/O2. 3101282	7.9250	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.000000	1	0	113803	53.1000	0
4	5	0	3	Allen, Mr. William Henry	0	35.000000	0	0	373450	8.0500	0
...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	0	27.000000	0	0	211536	13.0000	0
887	888	1	1	Graham, Miss. Margaret Edith	1	19.000000	0	0	112053	30.0000	0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	1	29.699118	1	2	W./C. 6607	23.4500	0
889	890	1	1	Behr, Mr. Karl Howell	0	26.000000	0	0	111369	30.0000	1
890	891	0	3	Dooley, Mr. Patrick	0	32.000000	0	0	370376	7.7500	2

891 rows × 11 columns

As we can observe that we have encoded the data andd replaced the categorical data into numerical data.

In [68]:
```python
X=df1.drop(columns=['Name','Ticket','PassengerId','Survived'],axis=1)
Y=df1['Survived']
```

In [69]:
```python
X
```

Out[69]:

	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 × 7 columns

In [70]:
```python
Y
```

Out[70]:
```
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
```

As we can observe that we have seperated features(input) and output with X and Y variable respectively

In [ ]:

SPLITTING THE DATA INTO TRAINING AND TESTING DATA

In [82]:
```python
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=2)
```

In [83]:
```python
print(X.shape,X_test.shape,X_train.shape)
```

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

MODEL TRAINING

In [84]:
```python
model=LogisticRegression()
```

In [85]:
```python
model.fit(X_train,Y_train)
```

Out[85]:
```
LogisticRegression()
```

MODEL EVALUATION

In [86]:
```python
X_train_prediction=model.predict(X_train)
```

In [87]:
```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 1 1 0 0 1 0 1 0 1 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 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 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]
```

As we can see the number of people who have survived(1) and who have not survived(0) by prediction through our training data model

In [88]:
```python
training_data_accuracy=accuracy_score(Y_train,X_train_prediction)
```

This is the comparing step of actual data adn predicted data of training

In [89]:
```python
print("ACCURACY SCORE OF TRAINING DATA:",training_data_accuracy)
```

```
ACCURACY SCORE OF TRAINING DATA: 0.8075842696629213
```

In [90]:
```python
X_test_prediction=model.predict(X_test)
```

In [91]:
```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]
```

In [92]:
```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
```

The accuracy score of training data and test data is very close which means the model is neither over fitted nor under fitted

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: