Titanic Survival Prediction Using Logistic Regression

In [1]:
```
#import Libraries
```

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Load the Data

In [3]:
```
titanic_data = pd.read_csv("Titanic Dataset.csv")
```

In [4]:
```
titanic_data
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embar
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	
...	...	...	...	...	...	...	...	...	...	...	...	
413	1305	0	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	
414	1306	1	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	
415	1307	0	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	
416	1308	0	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	
417	1309	0	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	

418 rows × 12 columns

In [5]:
```
# Length of Data
len(titanic_data)
```

Out[5]: 418

View the data using head function which return top 5 rows

In [6]:
```
titanic_data.head()
```

Out[6]:

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

In [7]:
```
titanic_data.index
```

Out[7]:
```
RangeIndex(start=0, stop=418, step=1)
```

In [8]:
```
titanic_data.columns
```

Out[8]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

summary of the DataFrame

In [9]:
```
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 [10]:
```
titanic_data.dtypes
```

Out[10]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

In [11]:
```
titanic_data.describe()
```

Out[11]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	418.000000	332.000000	418.000000	418.000000	417.000000
mean	1100.500000	0.363636	2.265550	30.272590	0.447368	0.392344	35.627188
std	120.810458	0.481622	0.841838	14.181209	0.896760	0.981429	55.907576
min	892.000000	0.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	996.250000	0.000000	1.000000	21.000000	0.000000	0.000000	7.895800
50%	1100.500000	0.000000	3.000000	27.000000	0.000000	0.000000	14.454200
75%	1204.750000	1.000000	3.000000	39.000000	1.000000	0.000000	31.500000
max	1309.000000	1.000000	3.000000	76.000000	8.000000	9.000000	512.329200

Explaining Datasets

survival : Survival 0 = No, 1 = Yes
pclass : Ticket class 1 = 1st, 2 = 2nd, 3 = 3rd
sex : Sex
Age : Age in years
sibsp : Number of siblings / spouses aboard the Titanic
parch # of parents / children aboard the Titanic
ticket : Ticket number fare Passenger fare cabin Cabin number
embarked : Port of Embarkation C = Cherbourg, Q = Queenstown, S = Southampton

Data Analysis

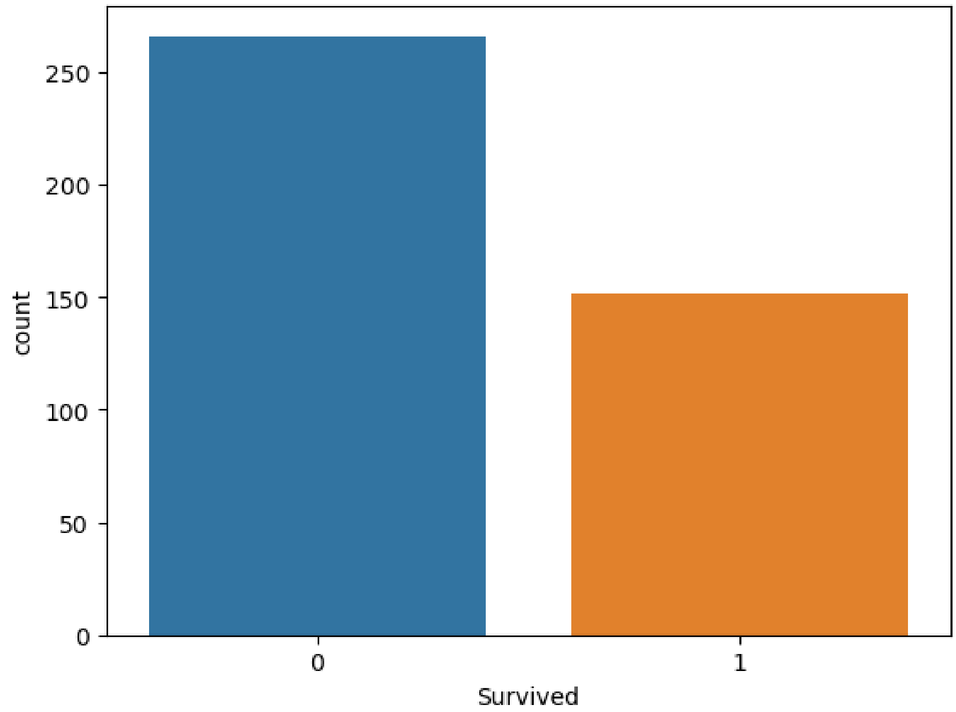
import Seaborn for visually analysing the data

Find out how many survived vs Died using countplot method of seaborn

In [12]:
```
#countplot of survived vs not survived
```

In [13]:
```
sns.countplot(x='Survived', data=titanic_data)
```

Out[13]: <Axes: xlabel='Survived', ylabel='count'>

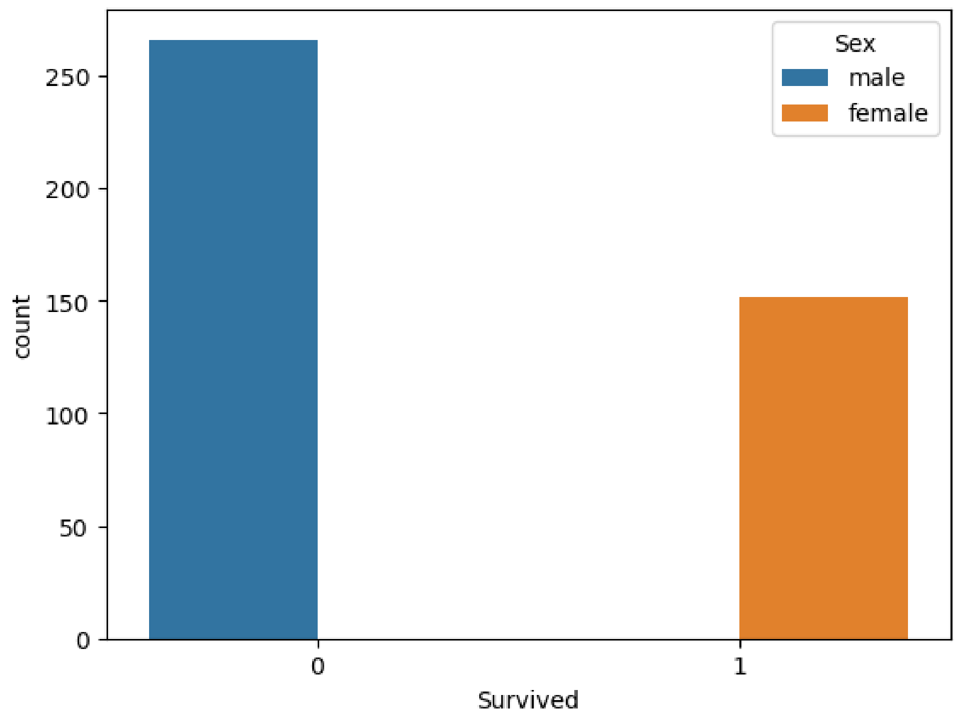


Male vs Female Survival

In [14]:
```
#Male vs Female Survival
```

In [15]:
```
sns.countplot(x='Survived', data=titanic_data, hue='Sex')
```

Out[15]: <Axes: xlabel='Survived', ylabel='count'>



Only Females are Survived and Male are not Survived as per the countplot

**See age group of passengeres travelled **
Note: We will use displot method to see the histogram. However some records does not have age hence the method will throw an error. In order to avoid that we will use dropna method to eliminate null values from graph

Check for null

In [16]:
```
titanic_data.isna()
```

Out[16]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	True	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	True	False
4	False	False	False	False	False	False	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...
413	False	False	False	False	False	True	False	False	False	False	True	False
414	False	False	False	False	False	False	False	False	False	False	False	False
415	False	False	False	False	False	False	False	False	False	False	True	False
416	False	False	False	False	False	True	False	False	False	False	True	False
417	False	False	False	False	False	True	False	False	False	False	True	False

418 rows × 12 columns

Check how many values are null

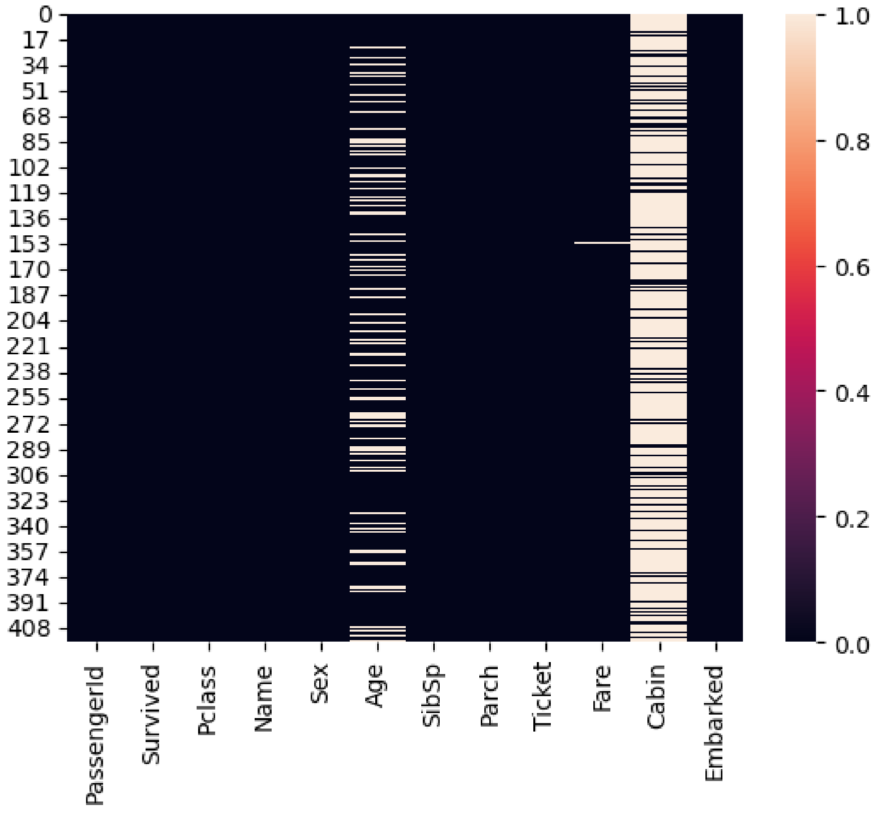
In [17]:
```
titanic_data.isna().sum()
```

Out[17]:
```
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
```

Visualize null values help of Heatmap

In [18]:
```
sns.heatmap(titanic_data.isna())
```

Out[18]: <Axes: >



find the % of null values in age column

In [19]:
```
(titanic_data['Age'].isna().sum()/len(titanic_data['Age']))*100
```

Out[19]: 20.574162679425836

find the % of null values in cabin column

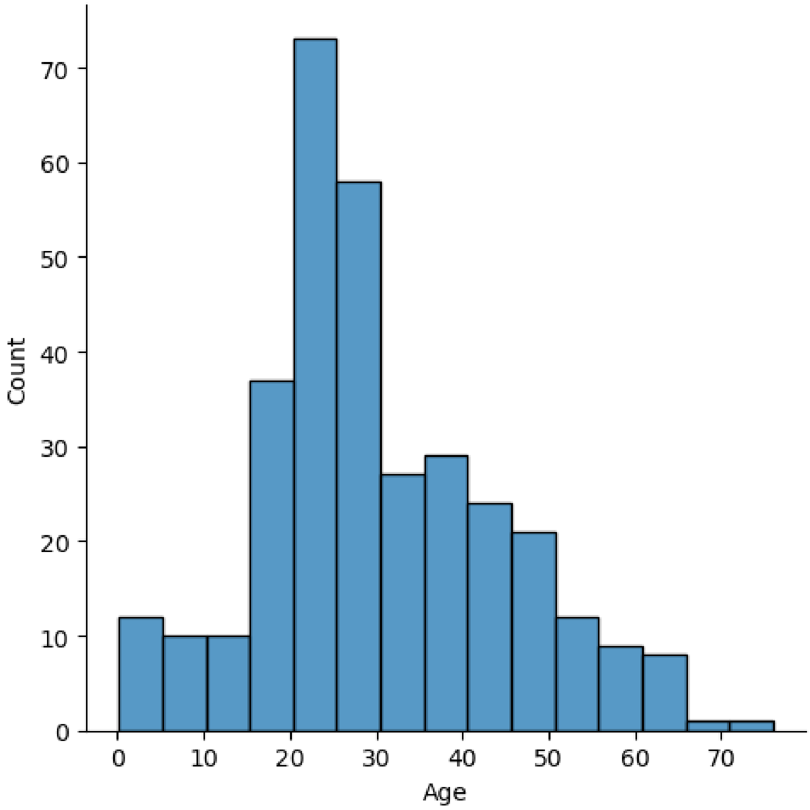
In [20]:
```
(titanic_data['Cabin'].isna().sum()/len(titanic_data['Cabin']))*100
```

Out[20]: 78.22966507177034

find the distribution for the age column

In [21]:
```
sns.displot(x='Age', data=titanic_data)
```

Out[21]:
```
<seaborn.axisgrid.FacetGrid at 0x2303ac93910>
```



Data Cleaning

Fill the missing values
we will find the missing values for age. In order to fill missing values we use fillna method.
For now we will fill the missing age by taking average of all age.

fill age column

In [22]:
```
titanic_data['Age'].fillna(titanic_data['Age'].mean(), inplace=True)
```

verify null values

In [23]:
```
titanic_data['Age'].isna().sum()
```

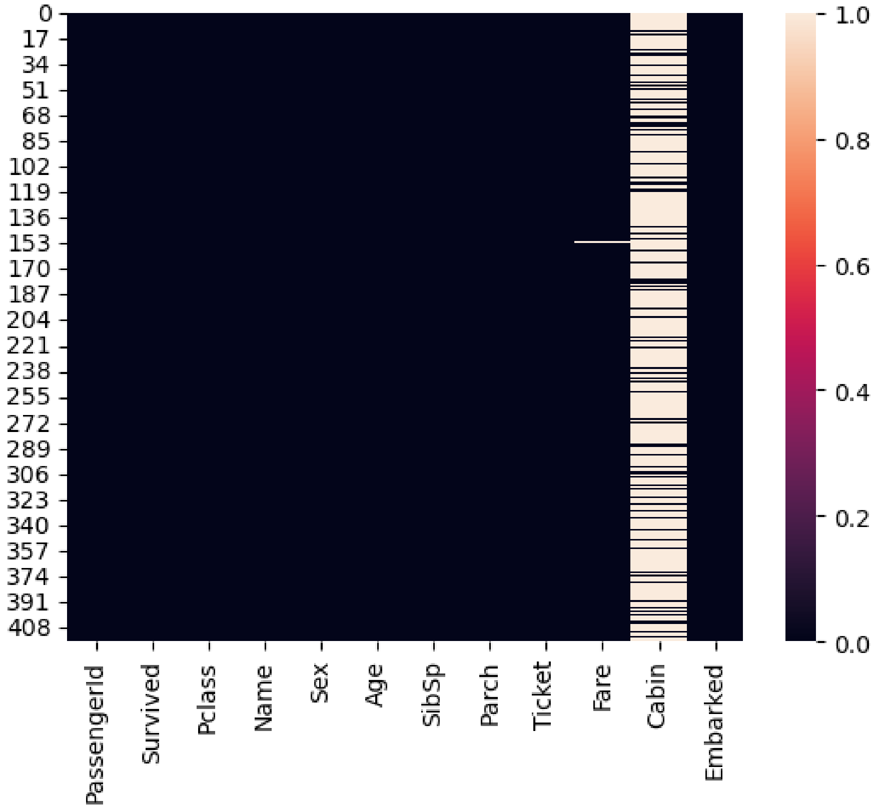
Out[23]:
```
0
```

Alternatively we will visualise the null value using heatmap.
we will use heatmap method by passing only records which are null.

In [24]:
```
#visualize null values again
```

In [25]:
```
sns.heatmap(titanic_data.isna())
```

Out[25]:
```
<Axes: >
```



we can see the cabin column has a number of null values, as such we can not use it for prediction. Hence we will drop it.

In [26]:
```
#drop cabin column
```

In [27]:
```
titanic_data.drop('Cabin', axis=1, inplace=True)
```

In [28]:
```
#see the content of data
```

In [29]:
```
titanic_data.head()
```

Out[29]:

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

Preaparing Data for Model

No we will require to convert all non-numerical columns to numeric. Please note this is required for feeding data into model. Lets see which columns are non numeric info describe method

In [30]:
```
#Check for the non-numeric column
```

In [31]:
```
titanic_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 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          418 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  Embarked     418 non-null    object 
dtypes: float64(2), int64(5), object(4)
memory usage: 36.0+ KB
```

In [32]:
```
titanic_data.dtypes
```

Out[32]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Embarked        object
dtype: object
```

We can see, Name, Sex, Ticket and Embarked are non-numerical.It seems Name,Embarked and Ticket number are not useful for Machine Learning Prediction hence we will eventually drop it. For Now we would convert Sex Column to dummies numerical values**

In [33]:
```
#convert sex column to numerical values
```

In [34]:
```
gender=pd.get_dummies(titanic_data['Sex'],drop_first=True)
```

In [35]:
```
titanic_data['Gender']=gender
```

In [36]:
```
titanic_data.head()
#Here, in Gender male=1 & female=0
```

Out[36]:

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

In [37]:
```
#drop the column which are not require
```

In [38]:
```
titanic_data.drop(['Name','Sex','Ticket','Embarked'], axis=1, inplace=True)
```

In [39]:
```
titanic_data.head()
```

Out[39]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	Gender
0	892	0	3	34.5	0	0	7.8292	1
1	893	1	3	47.0	1	0	7.0000	0
2	894	0	2	62.0	0	0	9.6875	1
3	895	0	3	27.0	0	0	8.6625	1
4	896	1	3	22.0	1	1	12.2875	0

In [40]:
```
titanic_data.isna().sum()
```

Out[40]:
```
PassengerId    0
Survived       0
Pclass         0
Age            0
SibSp          0
Parch          0
Fare           1
Gender         0
dtype: int64
```

In [43]:
```
# Fill null value with average in Fare Column
titanic_data['Fare'].fillna(titanic_data['Fare'].mean(), inplace=True)
```

In [44]:
```
#Seperate Dependent and Independent variables
```

In [45]:
```
x=titanic_data[['PassengerId','Pclass','Age','SibSp','Parch','Fare','Gender']]
y=titanic_data['Survived']
```

In [46]:
```
x
```

Out[46]:

	PassengerId	Pclass	Age	SibSp	Parch	Fare	Gender
0	892	3	34.50000	0	0	7.8292	1
1	893	3	47.00000	1	0	7.0000	0
2	894	2	62.00000	0	0	9.6875	1
3	895	3	27.00000	0	0	8.6625	1
4	896	3	22.00000	1	1	12.2875	0
...	...	...	...	...	...	...	...
413	1305	3	30.27259	0	0	8.0500	1
414	1306	1	39.00000	0	0	108.9000	0
415	1307	3	38.50000	0	0	7.2500	1
416	1308	3	30.27259	0	0	8.0500	1
417	1309	3	30.27259	1	1	22.3583	1

418 rows × 7 columns

In [47]:
```
y
```

Out[47]:
```
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
```

Data Modeling

Building Model using Logistic Regression

Build the model

In [48]:
```
#import train test split method
```

In [49]:
```
from sklearn.model_selection import train_test_split
```

In [50]:
```
#train test split
```

In [51]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)
```

In [52]:
```
#import Logistic Regression
```

In [53]:
```
from sklearn.linear_model import LogisticRegression
```

In [54]:
```
#fit Logistic Regression
```

In [55]:
```
lr = LogisticRegression()
```

In [57]:
```
lr.fit(X_train,y_train)

# Ignoring errors
import warnings
warnings.filterwarnings('ignore')
```

In [58]:
```
#predict
```

In [59]:
```
predict=lr.predict(X_test)
```

Testing

See how our model is performing

In [60]:
```
#print confusion matrix
```

In [61]:
```
from sklearn.metrics import confusion_matrix
```

In [62]:
```
pd.DataFrame(confusion_matrix(y_test,predict),columns=['Predicted No','Predicted Yes'],
             index=['Actual No','Actual Yes'])
```

Out[62]:

	Predicted No	Predicted Yes
Actual No	92	0
Actual Yes	0	46

In [63]:
```
#import classification report
```

In [64]:
```
from sklearn.metrics import classification_report
```

In [65]:
```
print(classification_report(y_test, predict))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        92
           1       1.00      1.00      1.00        46

    accuracy                           1.00       138
   macro avg       1.00      1.00      1.00       138
weighted avg       1.00      1.00      1.00       138
```

Precision is fine considering Model Selected and Available Data. Accuracy can be increased by further using more features (which we dropped earlier) and/or by using other model

Note:
Precision : Precision is the ratio of correctly predicted positive observations to the total predicted positive observations Recall : Recall is the ratio of correctly predicted positive observations to the all observations in actual class F1 score - F1 Score is the weighted average of Precision and Recall.