15_Predicting_Titanic_Survival_with_Logistic_Regression_and_Machine_

July 10, 2024

0.1 Analyzing Titanic Survival Using Logistic Regression: A Data-Driven Approach -Vignesh Prabhu

Explore how logistic regression models helped predict survival on the Titanic. By analyzing passenger data like age, gender, and ticket class, we aimed to understand factors influencing survival rates aboard the historic voyage. Dive into our findings and see how machine learning techniques shed light on this poignant chapter in maritime history.

Import Dependencies

[1]:
```
import pandas as pd
import numpy as np
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 Preprocessing

[2]:
```
#Loading the Data to DataFrame
titanic_data = pd.read_csv('/content/Titanic.csv')
```

[3]:
```
#To Display First 5 Data's in dataset
titanic_data.head()
```

[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th…	female	38.0	1
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1
4	5	0	3				

```
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]:
```
#To show Last 5 data's in dataset
titanic_data.tail()
```

[4]:
```
     PassengerId  Survived  Pclass                                          Name  \
886          887         0       2                         Montvila, Rev. Juozas
887          888         1       1                  Graham, Miss. Margaret Edith
888          889         0       3      Johnston, Miss. Catherine Helen "Carrie"
889          890         1       1                          Behr, Mr. Karl Howell
890          891         0       3                            Dooley, Mr. Patrick

        Sex   Age  SibSp  Parch      Ticket   Fare Cabin Embarked
886    male  27.0      0      0      211536  13.00   NaN        S
887  female  19.0      0      0      112053  30.00   B42        S
888  female   NaN      1      2  W./C. 6607  23.45   NaN        S
889    male  26.0      0      0      111369  30.00  C148        C
890    male  32.0      0      0      370376   7.75   NaN        Q
```

[5]:
```
#To Check Number of rows and columns
titanic_data.shape
```

[5]:
```
(891, 12)
```

[6]:
```
#To Check Null values In dataset
titanic_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]:
```
#Remove Cabin Column in dataFrame
titanic_data = titanic_data.drop(columns='Cabin', axis=1)
```

[8]:
```
#Replacing Age Column with Mean Values
titanic_data['Age'].fillna(titanic_data['Age'].mean(), inplace=True)
```

[9]:
```
#Find The Mode Value
print(titanic_data['Embarked'].mode())
```

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

[10]:
```
#Replacing Embarked column with Mode Values -Repeated values
titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0],␣
 ↪inplace=True)   #[0]- Index Value
```

[11]:
```
#To Check Null values In dataset
titanic_data.isnull().sum()
```

[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
```

[12]:
```
#Information
titanic_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 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          891 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  Embarked     891 non-null    object
dtypes: float64(2), int64(5), object(4)
memory usage: 76.7+ KB
```

Data Analysis

[13]:
```
#statistical Values
titanic_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

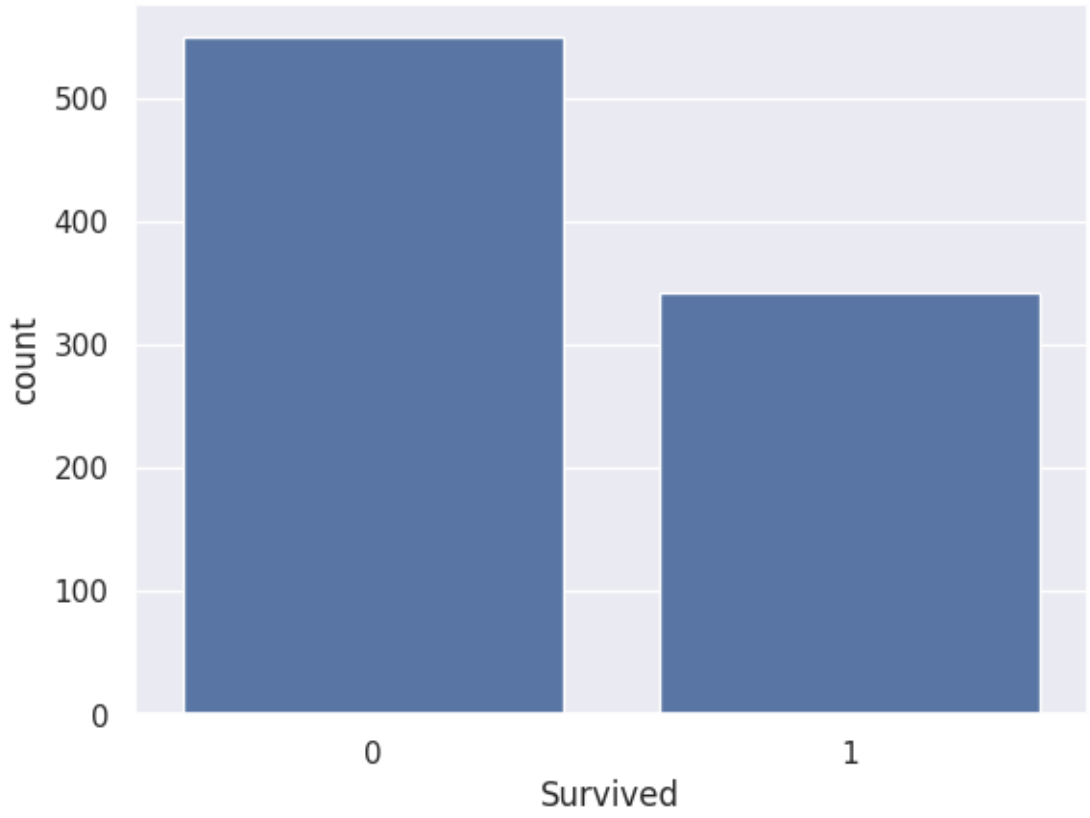
[14]:
```
#Finding The number Of people survived and Not Survived
titanic_data['Survived'].value_counts()
```

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

Data Visualization

[15]:
```
#Ploting
sns.set()          #Gives the some theme
sns.countplot(x='Survived', data=titanic_data) # Specify 'x' to avoid confusion
```

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

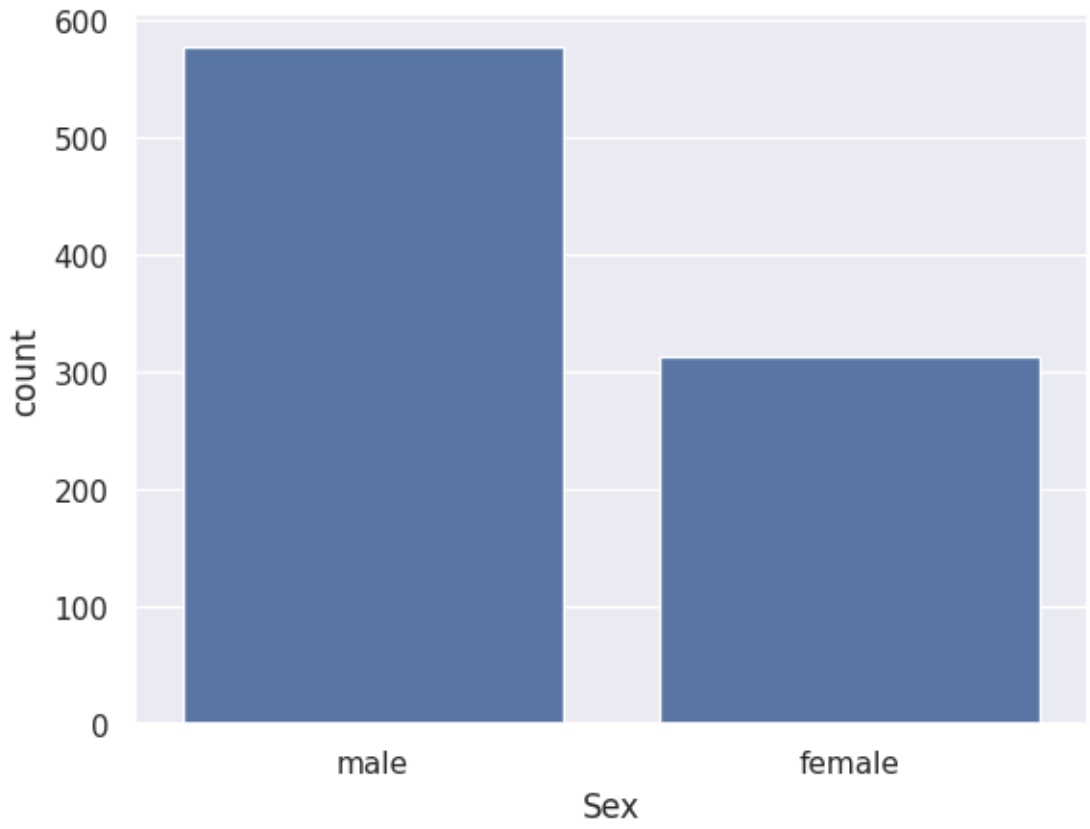


[16]:
```
#Finding The number Genders
titanic_data['Sex'].value_counts()
```

[16]:
```
Sex
male      577
female    314
Name: count, dtype: int64
```

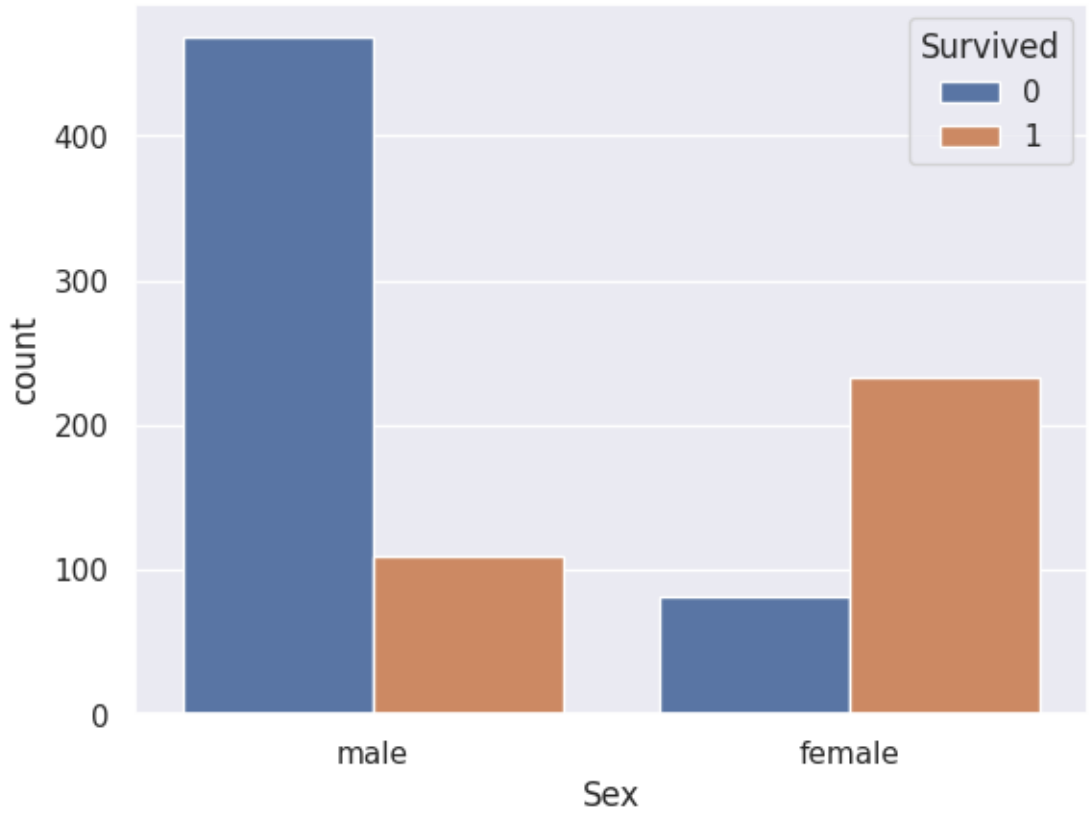
[17]:
```
#Countplot For gender
sns.countplot(x='Sex', data=titanic_data)
```

[17]:
```
<Axes: xlabel='Sex', ylabel='count'>
```



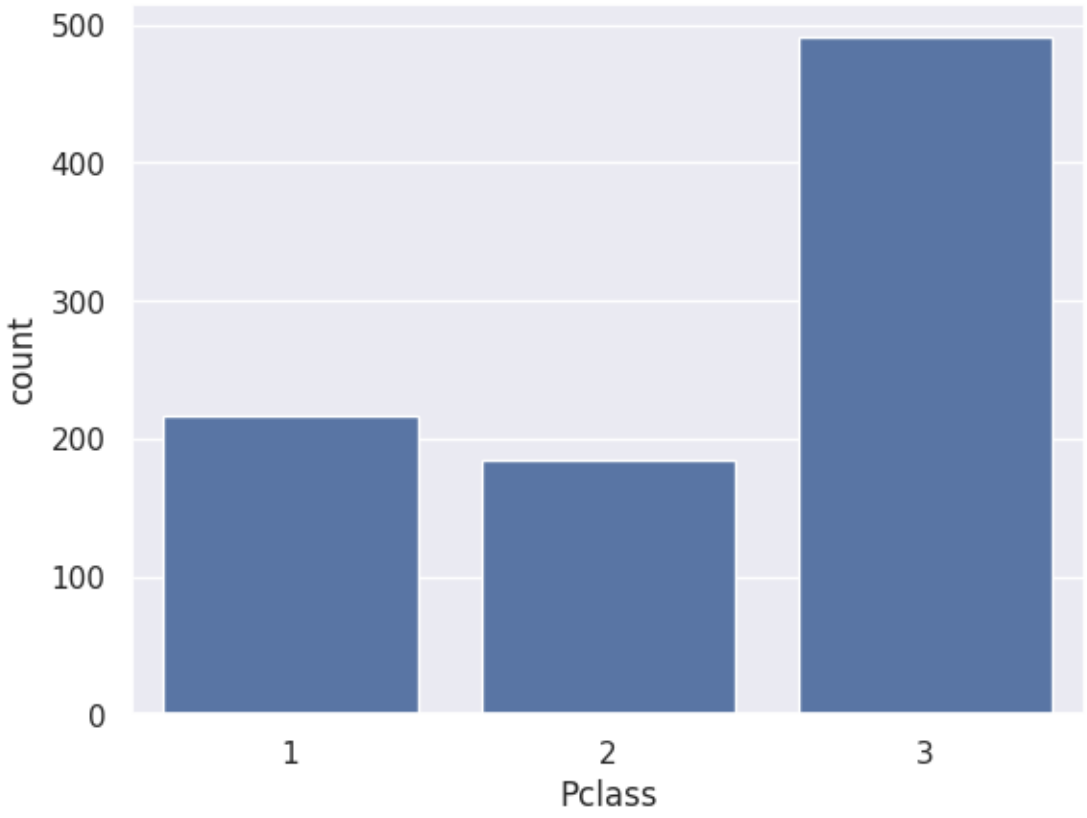
[18]:
```
#Number Survivors gender wise
sns.countplot(x='Sex', hue='Survived', data=titanic_data)
```

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



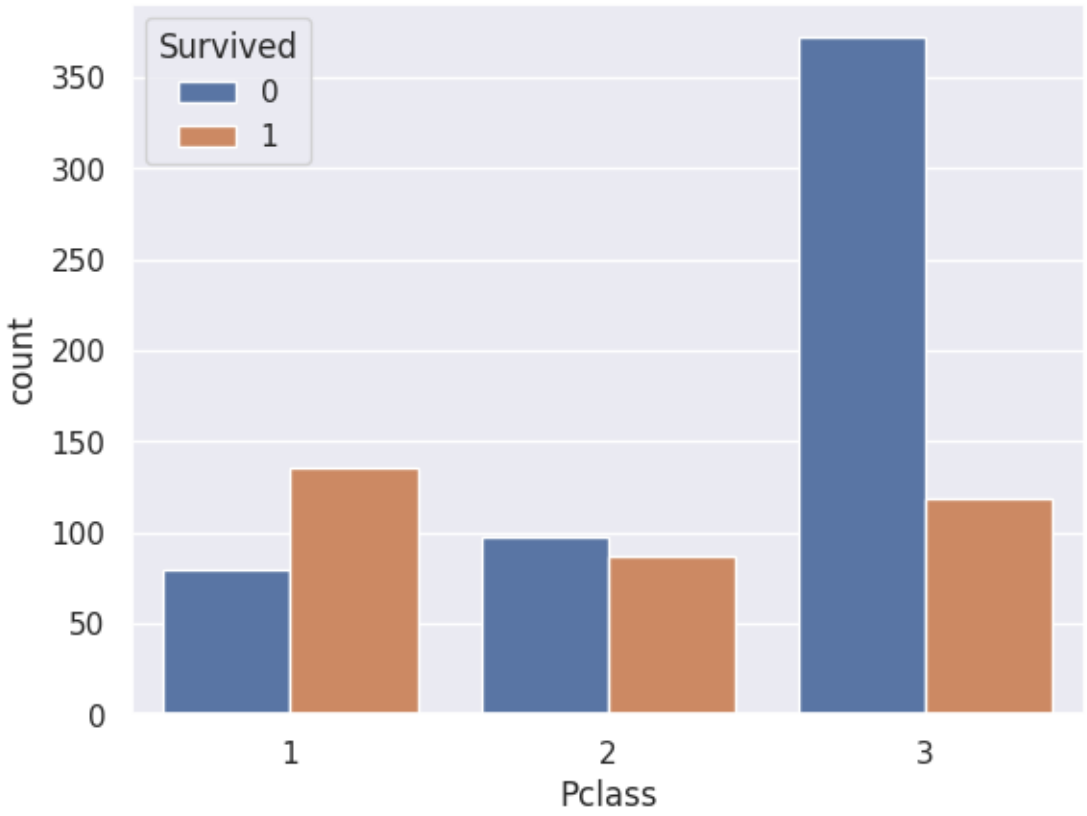
[19]:
```
#plotting  'pclass'
sns.countplot(x='Pclass', data=titanic_data)
```

[19]: <Axes: xlabel='Pclass', ylabel='count'>



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

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



Encoding Categorical Columns

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

[21]:
```
Sex
male      577
female    314
Name: count, dtype: int64
```

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

[22]:
```
Embarked
S    646
C    168
Q     77
Name: count, dtype: int64
```

[23]:
```
#Converting categorical columns
titanic_data.replace({'Sex':{'male':0,'female':1}, 'Embarked':{'S':0,'C':1,'Q':
 ↪2}}, inplace=True)
```

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

[24]:
```
   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
```

Seperating Features and Targets

[25]:
```
X= titanic_data.drop(columns =␣
 ↪['PassengerId','Name','Ticket','Survived'],axis=1)
Y= titanic_data['Survived']
```

[26]:
```
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]
```

[27]:
```
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
```

Spliting Data into Training and Testing data

[28]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2,␣
↪random_state=2)
```

[29]:
```
print(X.shape, X_train.shape, X_test.shape)
```

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

Model Training

[30]:
```
Model= LogisticRegression(max_iter=1000, solver='lbfgs')
```

[31]:
```
Model.fit(X_train, Y_train)
```

[31]:
```
LogisticRegression(max_iter=1000)
```

Model Evaluation

[32]:
```
#Prediction on Training Data
X_train_prediction = Model.predict(X_train)
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 0 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 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 0 1 0 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
 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 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
 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 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 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
 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 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 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
```

```
 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]
```

[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 :  0.8089887640449438
```

[34]:
```
#Prediction On Test data
X_test_prediction = Model.predict(X_test)
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 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]
```

[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 :  0.7821229050279329
```

Model Testing

[36]:
```
#input_data = (1,1,38.000000,1,0,71.2833,1)

input_data =(1,0,35.000000,1,0,88.2123,0)


#changing  input_data to a numpy array
input_data_as_numpy_array=np.asarray(input_data)

#reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = Model.predict(input_data_reshaped)
print(prediction)

if (prediction[0]==0):
  print('The Person does not Survived')
else:
  print('The Person Survived')
```

```
[0]
```

```
The Person does not Survived
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but LogisticRegression was fitted with feature
names
  warnings.warn(
```

In conclusion, this project successfully applied logistic regression and machine learning techniques to analyze Titanic passenger data. By examining factors such as age, gender, and class, we predicted survival probabilities with significant accuracy. This not only enhances our understanding of historical events but also demonstrates the effectiveness of data-driven approaches in predicting real-world outcomes.

0.2 Thank You!