## Importing the Dependencies

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

## Data Collection

```
# Load the data from .csv file to pandas dataframe
data = pd.read_csv('https://raw.githubusercontent.com/YBI-Foundation/Dataset/main/Titanic.
```

```
# Printing the first five rows of the dataframe
data.head()
```

| | pclass | survived | name | sex | age | sibsp | parch | ticket | fare | cabin | e |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | Allen, Miss. Elisabeth Walton | female | 29.00 | 0 | 0 | 24160 | 211.3375 | B5 | |
| **1** | 1 | 1 | Allison, Master. Hudson T... | male | 0.92 | 1 | 2 | 113781 | 151.5500 | C22 C26 | |

```
# Number of rows and columns in our dataset
data.shape
```

```
(1309, 14)
```

```
# Information about our data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   pclass     1309 non-null   int64
 1   survived   1309 non-null   int64
 2   name       1309 non-null   object
 3   sex        1309 non-null   object
 4   age        1046 non-null   float64
 5   sibsp      1309 non-null   int64
 6   parch      1309 non-null   int64
```

```
 7   ticket     1309 non-null   object
 8   fare       1308 non-null   float64
 9   cabin      295 non-null    object
 10  embarked   1307 non-null   object
 11  boat       486 non-null    object
 12  body       121 non-null    float64
 13  home.dest  745 non-null    object
dtypes: float64(3), int64(4), object(7)
memory usage: 143.3+ KB
```

```
# Number of missing values in each column
data.isnull().sum()
```

```
pclass          0
survived        0
name            0
sex             0
age           263
sibsp           0
parch           0
ticket          0
fare            1
cabin        1014
embarked        2
boat          823
body         1188
home.dest     564
dtype: int64
```

## Handling the Missing Values

```
# Dropping the 'cabin', 'boat', 'body' and 'home.dest' columns from the dataframe
data = data.drop(columns = ['cabin', 'boat', 'body', 'home.dest'], axis = 1)
```

```
# Replacing the missing values in 'age' column with the mean value
data['age'].fillna(data['age'].mean(), inplace = True)
```

```
# Replacing the missing values in 'embarked' column with the mode value
data['embarked'].fillna(data['embarked'].mode()[0], inplace = True)
```

```
# Replacing the missing values in 'fare' column with the mode value
data['fare'].fillna(data['fare'].mode()[0], inplace = True)
```

```
# Again checking the number of missing values in each column
data.isnull().sum()
```

```
pclass      0
survived    0
name        0
sex         0
age         0
sibsp       0
```

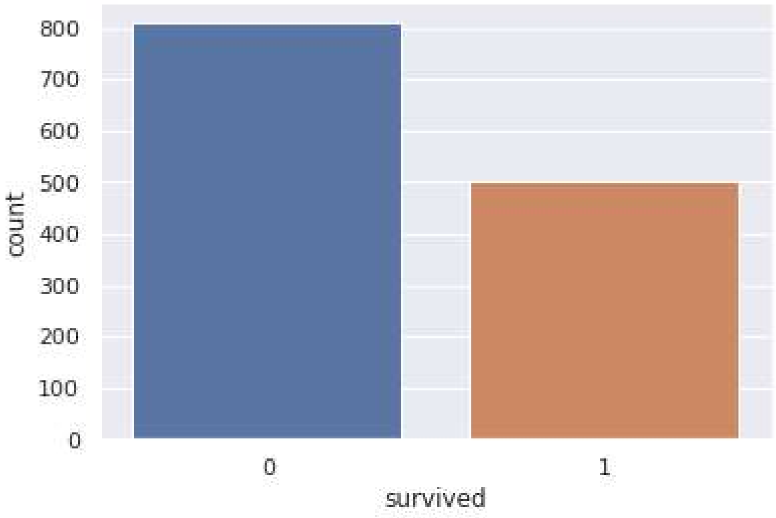
```
parch       0
ticket      0
fare        0
embarked    0
dtype: int64
```

## Data Visualization

```
sns.set()
```

```
# count plot for 'survived' column
sns.countplot('survived', data = data)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe639fee50>
```

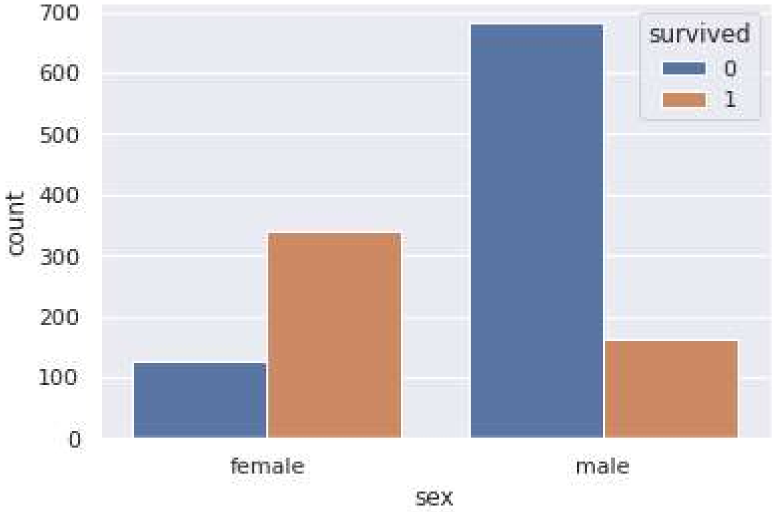



```
# count plot for 'sex' column
sns.countplot('sex', data = data)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
```

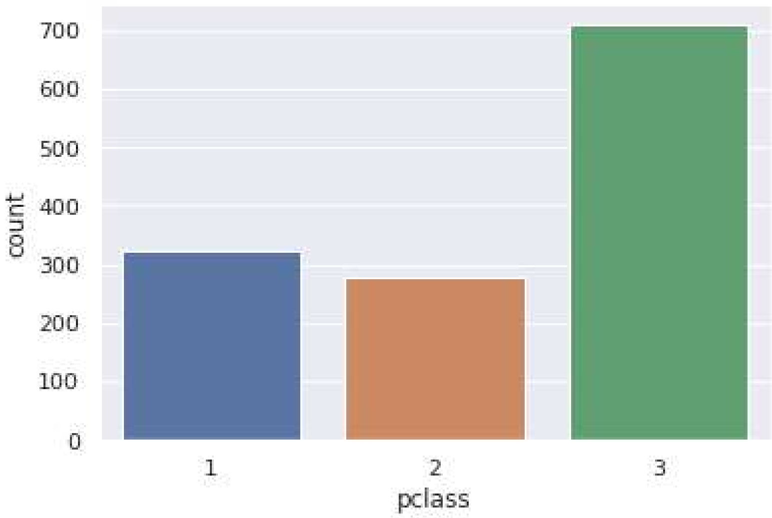
```
# count plot for 'survived' column gender-wise
sns.countplot('sex', hue = 'survived', data = data)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe52cac210>
```



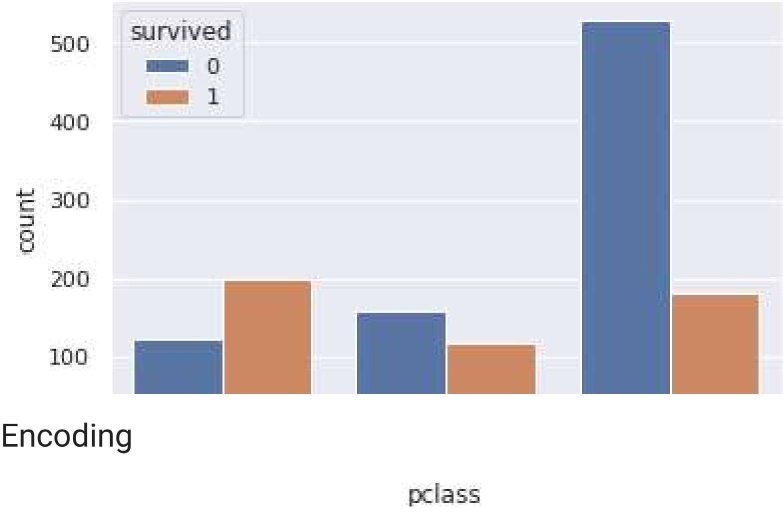

```
# count plot for 'pclass' column
sns.countplot('pclass', data = data)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe52c25b10>
```




```
# count plot for 'survived' column class-wise
sns.countplot('pclass', hue = 'survived', data = data)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas:
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe52c38c50>
```




## Data Encoding

```
data['sex'].value_counts()
```

```
male      843
female    466
Name: sex, dtype: int64
```

```
data['embarked'].value_counts()
```

```
S    916
C    270
Q    123
Name: embarked, dtype: int64
```

```
# Conversion to categorical columns
data.replace({'sex' : {'male' : 0, 'female' : 1}, 'embarked' : {'S' : 0, 'C' : 1, 'Q' : 2}
```

```
data.head()
```

| | pclass | survived | name | sex | age | sibsp | parch | ticket |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | Allen, Miss. Elisabeth Walton | 1 | 29.00 | 0 | 0 | 24160 |
| **1** | 1 | 1 | Allison, Master. Hudson Trevor | 0 | 0.92 | 1 | 2 | 113781 |
| **2** | 1 | 0 | Allison, Miss. Helen Loraine | 1 | 2.00 | 1 | 2 | 113781 |
| **3** | 1 | 0 | Allison, Mr. Hudson Joshua Creighton | 0 | 30.00 | 1 | 2 | 113781 |
| **4** | 1 | 0 | Allison, Mrs. Hudson J C (Bessie | 1 | 25.00 | 1 | 2 | 113781 |

## Data Analysis

```
# Getting statistical data about the dataset
data.describe()
```

| | pclass | survived | sex | age | sibsp | parch | |
|---|---|---|---|---|---|---|---|
| **count** | 1309.000000 | 1309.000000 | 1309.000000 | 1309.000000 | 1309.000000 | 1309.000000 | 13 |
| **mean** | 2.294882 | 0.381971 | 0.355997 | 29.881138 | 0.498854 | 0.385027 | |
| **std** | 0.837836 | 0.486055 | 0.478997 | 12.883193 | 1.041658 | 0.865560 | |
| **min** | 1.000000 | 0.000000 | 0.000000 | 0.170000 | 0.000000 | 0.000000 | |
| **25%** | 2.000000 | 0.000000 | 0.000000 | 22.000000 | 0.000000 | 0.000000 | |
| **50%** | 3.000000 | 0.000000 | 0.000000 | 29.881138 | 0.000000 | 0.000000 | |
| **75%** | 3.000000 | 1.000000 | 1.000000 | 35.000000 | 1.000000 | 0.000000 | |
| **max** | 3.000000 | 1.000000 | 1.000000 | 80.000000 | 8.000000 | 9.000000 | 5 |

```
# Getting co-relation data about the dataset
data.corr()
```

| | pclass | survived | sex | age | sibsp | parch | fare | em |
|---|---|---|---|---|---|---|---|---|
| **pclass** | 1.000000 | -0.312469 | -0.124617 | -0.366371 | 0.060832 | 0.018322 | -0.558740 | 0. |
| **survived** | -0.312469 | 1.000000 | 0.528693 | -0.050198 | -0.027825 | 0.082660 | 0.244479 | 0. |
| **sex** | -0.124617 | 0.528693 | 1.000000 | -0.057397 | 0.109609 | 0.213125 | 0.185744 | 0. |
| **age** | -0.366371 | -0.050198 | -0.057397 | 1.000000 | -0.190747 | -0.130872 | 0.170619 | 0. |
| **sibsp** | 0.060832 | -0.027825 | 0.109609 | -0.190747 | 1.000000 | 0.373587 | 0.160388 | -0. |
| **parch** | 0.018322 | 0.082660 | 0.213125 | -0.130872 | 0.373587 | 1.000000 | 0.221668 | -0. |
| **fare** | -0.558740 | 0.244479 | 0.185744 | 0.170619 | 0.160388 | 0.221668 | 1.000000 | 0. |
| **embarked** | 0.038875 | 0.098450 | 0.120423 | 0.035824 | -0.073461 | -0.095523 | 0.061337 | 1. |

## Separating Features and Target variable

```
X = data.drop(columns = ['name', 'ticket', 'survived'], axis = 1)
y = data['survived']
```

```
X
```

| | pclass | sex | age | sibsp | parch | fare | embarked |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 29.000000 | 0 | 0 | 211.3375 | 0 |
| **1** | 1 | 0 | 0.920000 | 1 | 2 | 151.5500 | 0 |
| **2** | 1 | 1 | 2.000000 | 1 | 2 | 151.5500 | 0 |
| **3** | 1 | 0 | 30.000000 | 1 | 2 | 151.5500 | 0 |
| **4** | 1 | 1 | 25.000000 | 1 | 2 | 151.5500 | 0 |

```
y
```

```
0       1
1       1
2       0
3       0
4       0
       ..
1304    0
1305    0
1306    0
1307    0
1308    0
Name: survived, Length: 1309, dtype: int64
```

```
X.shape, y.shape
```

```
((1309, 7), (1309,))
```

## Splitting the data into Training data and Testing data

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state =
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((916, 7), (393, 7), (916,), (393,))
```

## Training Model

```
model = LogisticRegression(max_iter = 500)
```

```
# Training our model
model.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=500)
```

## Model Evaluation

```
# Using our model to predict the values for X_test dataframe
y_predict = model.predict(X_test)
```

```
y_predict
```

```
array([1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,
       0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1])
```

```
confusion_matrix(y_test, y_predict)
```

```
array([[228,  26],
       [ 40,  99]])
```

```
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.85      0.90      0.87       254
           1       0.79      0.71      0.75       139

    accuracy                           0.83       393
   macro avg       0.82      0.80      0.81       393
weighted avg       0.83      0.83      0.83       393
```

```
accuracy_score(y_test, y_predict)
```

```
0.8320610687022901
```

✓ 0s completed at 23:19