Importing important libraries

In [2]:

```
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
```

Importing data

In [3]:

```
data = pd.read_csv("Titanic-Dataset.csv")
```

In [4]:
```
data
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fa
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.28
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.10
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75

891 rows × 12 columns

In [10]:
```
new_data = data.drop(['PassengerId'], axis=1)
```

In [11]:
```
new_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  891 non-null    int64  
 1   Pclass    891 non-null    int64  
 2   Name      891 non-null    object 
 3   Sex       891 non-null    object 
 4   Age       714 non-null    float64
 5   SibSp     891 non-null    int64  
 6   Parch     891 non-null    int64  
 7   Ticket    891 non-null    object 
 8   Fare      891 non-null    float64
 9   Cabin     204 non-null    object 
 10  Embarked  889 non-null    object 
dtypes: float64(2), int64(4), object(5)
memory usage: 76.7+ KB
```

In [12]:
```
new_data.isnull().sum()
```

Out[12]:
```
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

In [13]:
```
titanic_data = new_data.drop(['Cabin'], axis =1)
```

In [14]:
```
titanic_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  891 non-null    int64  
 1   Pclass    891 non-null    int64  
 2   Name      891 non-null    object 
 3   Sex       891 non-null    object 
 4   Age       714 non-null    float64
 5   SibSp     891 non-null    int64  
 6   Parch     891 non-null    int64  
 7   Ticket    891 non-null    object 
 8   Fare      891 non-null    float64
 9   Embarked  889 non-null    object 
dtypes: float64(2), int64(4), object(4)
memory usage: 69.7+ KB
```

In [15]:
```
titanic_data['Age'].fillna(titanic_data['Age'].mean(),inplace=True)
```

In [16]:
```
titanic_data.isnull().sum()
```

Out[16]:
```
Survived    0
Pclass      0
Name        0
Sex         0
Age         0
SibSp       0
Parch       0
Ticket      0
Fare        0
Embarked    2
dtype: int64
```

In [17]:
```
# Finding the mode value of Embarked column.
print(titanic_data['Embarked'].mode())
```

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

In [21]:
```
# Replacing the missing values in embarked column with mode value.
titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0], inplace
```

In [2]:
```
titanic_data.isnull().sum()
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
Cell In[2], line 1
----> 1 titanic_data.isnull().sum()

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

Data Analysis

In [23]:

```
titanic_data.describe()
```

Out[23]:

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

In [24]:

```
titanic_data['Survived'].value_counts()
```

Out[24]:

```
0    549
1    342
Name: Survived, dtype: int64
```

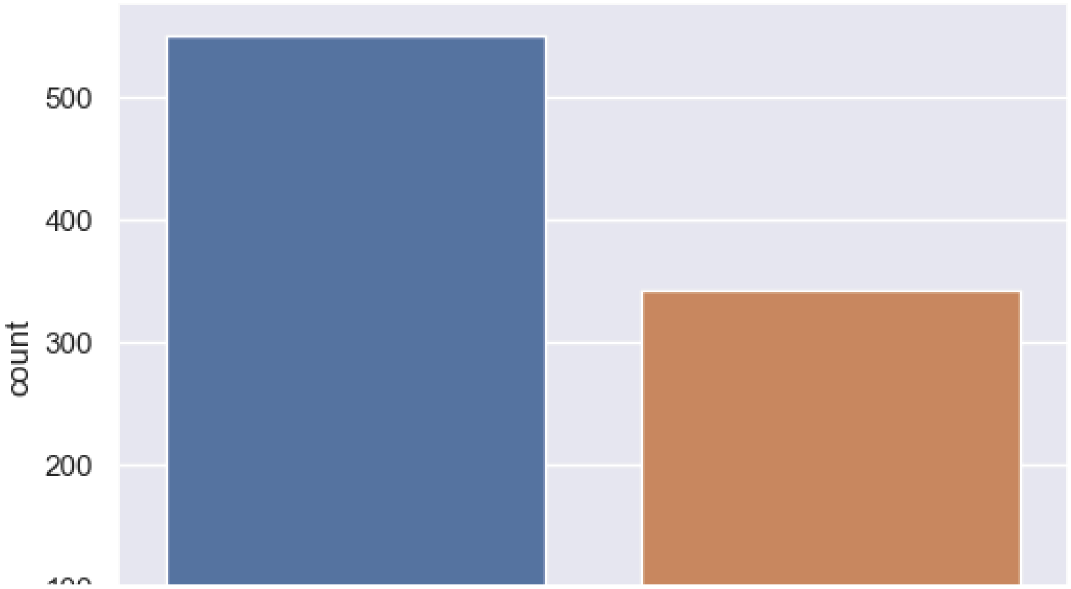
Data Visualization

In [29]:

```
# Count plot - survived column
sns.countplot(x='Survived', data= titanic_data)
```

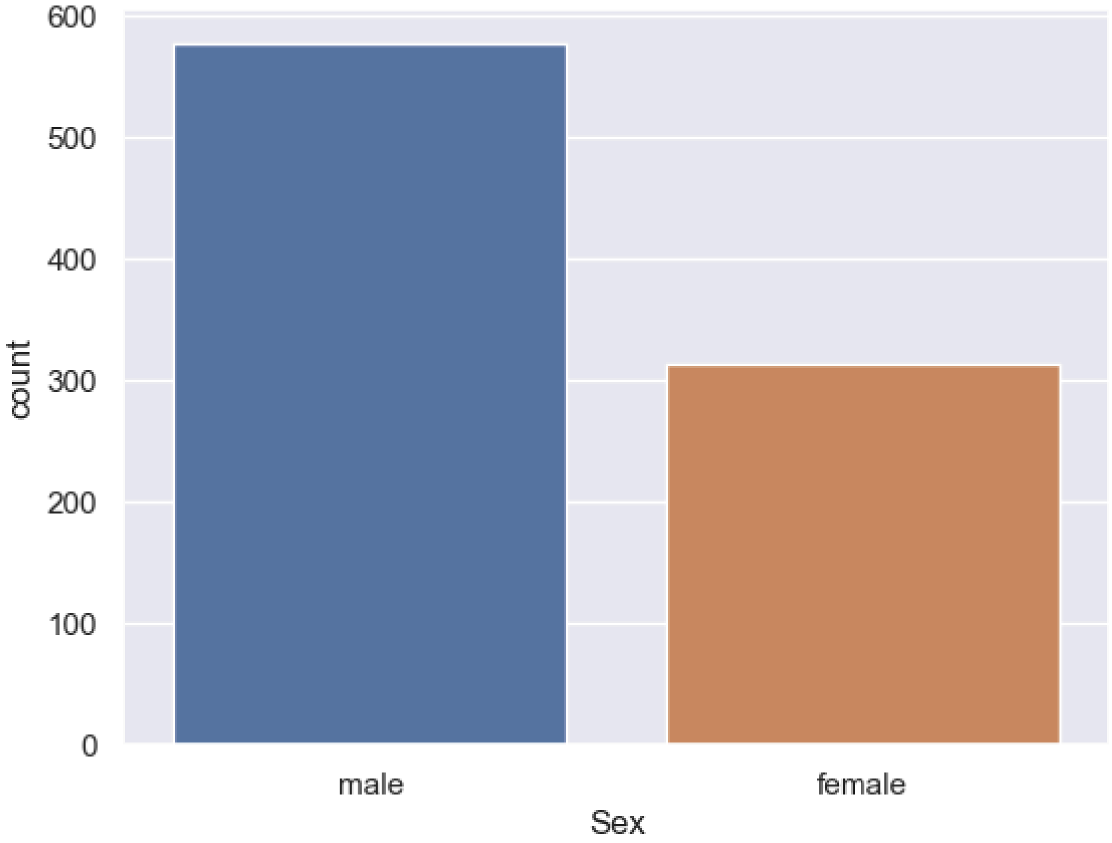
Out[29]:

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



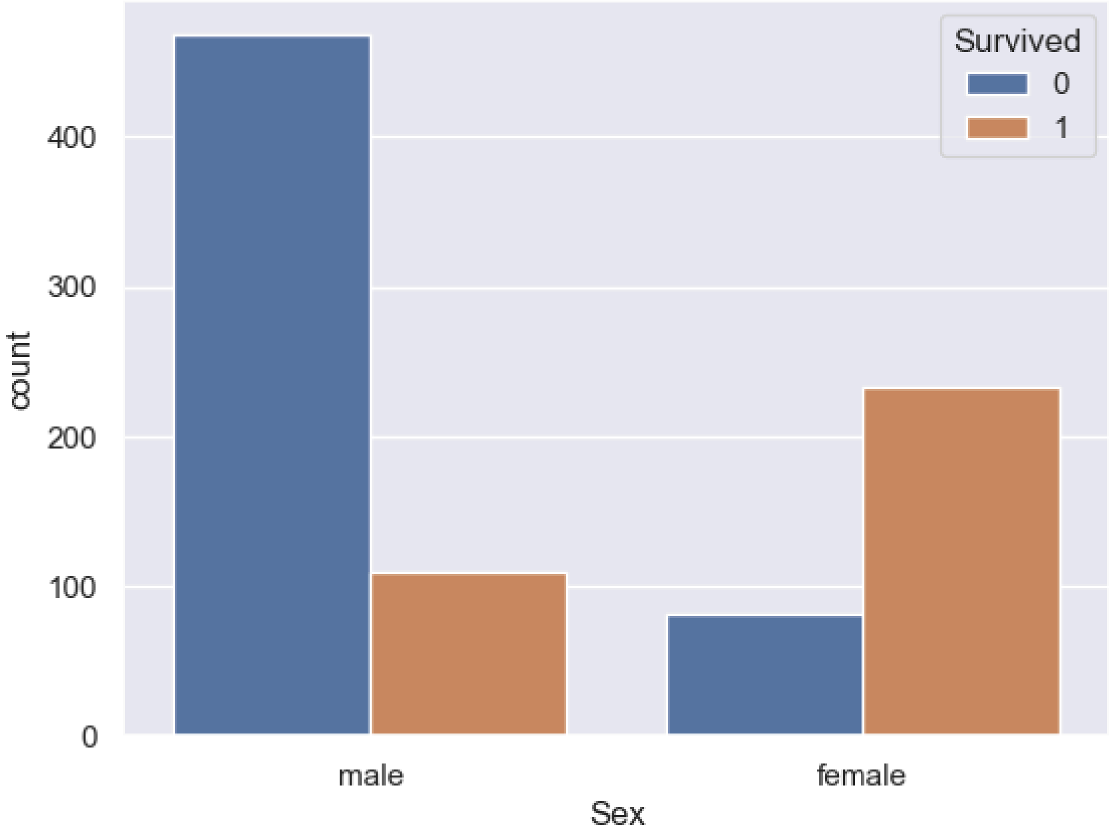
```
In [30]: sns.countplot(x='Sex', data= titanic_data)
```

```
Out[30]: <Axes: xlabel='Sex', ylabel='count'>
```



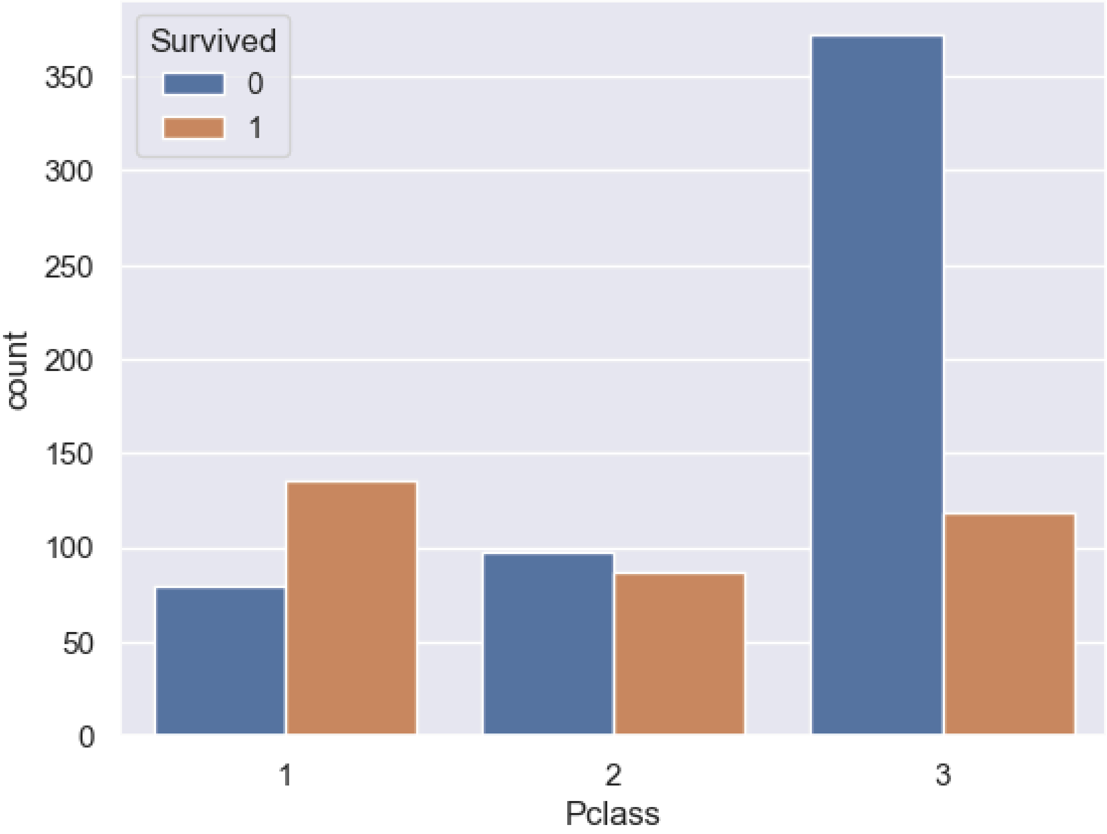
In [32]:
```
sns.countplot(x="Sex",hue="Survived",data=titanic_data)
```

Out[32]:
```
<Axes: xlabel='Sex', ylabel='count'>
```



```
In [33]: sns.countplot(x="Pclass",hue="Survived",data=titanic_data)
```

```
Out[33]: <Axes: xlabel='Pclass', ylabel='count'>
```



Encoding the categorical columns

```
In [36]: titanic_data.replace({'Sex': {'male': 0, 'female': 1}, 'Embarked': {'S': 0,
```

```
In [37]: titanic_data.head()
```

Out[37]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	0	3	Braund, Mr. Owen Harris	0	22.0	1	0	A/5 21171	7.2500	0
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.0	1	0	PC 17599	71.2833	1
2	1	3	Heikkinen, Miss. Laina	1	26.0	0	0	STON/O2. 3101282	7.9250	0
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.0	1	0	113803	53.1000	0
4	0	3	Allen, Mr. William Henry	0	35.0	0	0	373450	8.0500	0

Separating features and targets

In [42]:

```
x = titanic_data.drop(["Name","Ticket","Survived"],axis=1)
```

In [43]:

```
y = titanic_data['Survived']
```

In [44]:

```
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]
```

In [45]:

```
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
```

Splitting the data into training and testing data

In [46]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_s
```

In [48]:

```
print(x.shape,x_train.shape,x_test.shape)
```

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

Training model

In [49]:
```
model = LogisticRegression()
```

In [50]:
```
model.fit(x_train,y_train)
```

```
C:\Users\Apoorva\anaconda3\Lib\site-packages\sklearn\linear_model\_logisti
c.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

Out[50]:
```
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.

Model evaluation

In [51]:
```
x_train_prediction = model.predict(x_train)
```

```
x_train_prediction
```

In [53]:
```
training_data_accuracy = accuracy_score(y_train,x_train_prediction)
```

In [54]:
```
print("Accuracy score of trainig data:",training_data_accuracy )
```

```
Accuracy score of trainig data: 0.8075842696629213
```

In [55]:
```
x_test_prediction = model.predict(x_test)
```

In [56]:
```
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 [58]:

```
test_data_accuracy = accuracy_score(y_test,x_test_prediction)
```

In [60]:

```
print("Accuracy score of test data", test_data_accuracy)
```

```
Accuracy score of test data 0.7821229050279329
```

In [ ]: