TASK 1: TITANIC SURVIVAL PREDICTION

Name : Prathamesh Santosh Tondilkar

Batch : December

Domain : Data Science

The dataset typically used for this project contains information about and individual passengers, such as their age, gender, ticket class, fare, cabin, and whether or not they survived.

IMPORTING IMPORTANT LIBRARIES

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sbn
```

IMPORTING DATASET

In [2]:
```
ds=pd.read_csv("tested.csv")
```

In [3]:
```
ds.head()
```

Out[3]:

	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 [4]:
```
ds.tail()
```

Out[4]:

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

In [5]:
```
ds.shape
```

Out[5]:
```
(418, 12)
```

In [6]:
```
ds.dtypes
```

Out[6]:
```
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 [7]:
```
ds.describe()
```

Out[7]:

	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

In [8]:
```
ds.isna().mean()
```

Out[8]:
```
PassengerId    0.000000
Survived       0.000000
Pclass         0.000000
Name           0.000000
Sex            0.000000
Age            0.205742
SibSp          0.000000
Parch          0.000000
Ticket         0.000000
Fare           0.002392
Cabin          0.782297
Embarked       0.000000
dtype: float64
```

In [9]:
```
ds['Survived'].value_counts()
```

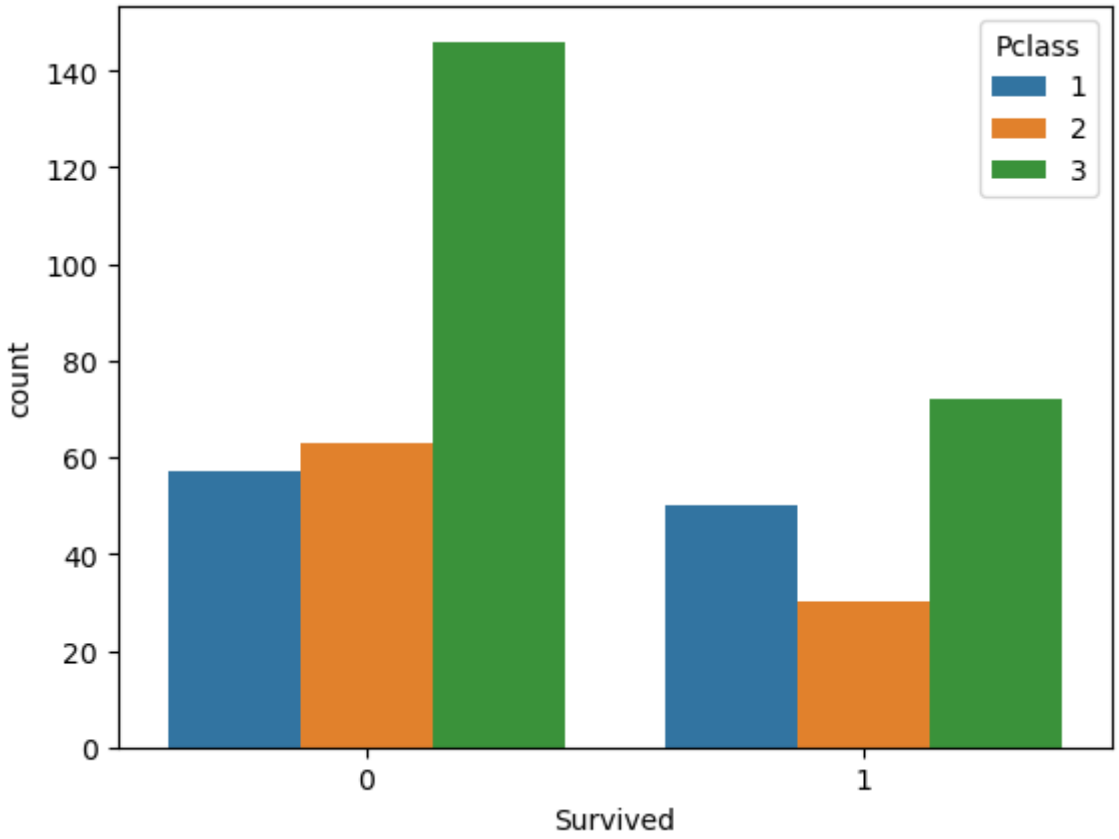
Out[9]:
```
0    266
1    152
Name: Survived, dtype: int64
```

DATA VISUALIZATION

Visualization of survival count wrt Pclass

In [10]:
```
sbn.countplot(x=ds['Survived'], hue=ds['Pclass'])
```

Out[10]:
```
<Axes: xlabel='Survived', ylabel='count'>
```



In [11]:
```
ds["Sex"]
```

Out[11]:
```
0        male
1      female
2        male
3        male
4      female
        ...
413      male
414    female
415      male
416      male
417      male
Name: Sex, Length: 418, dtype: object
```

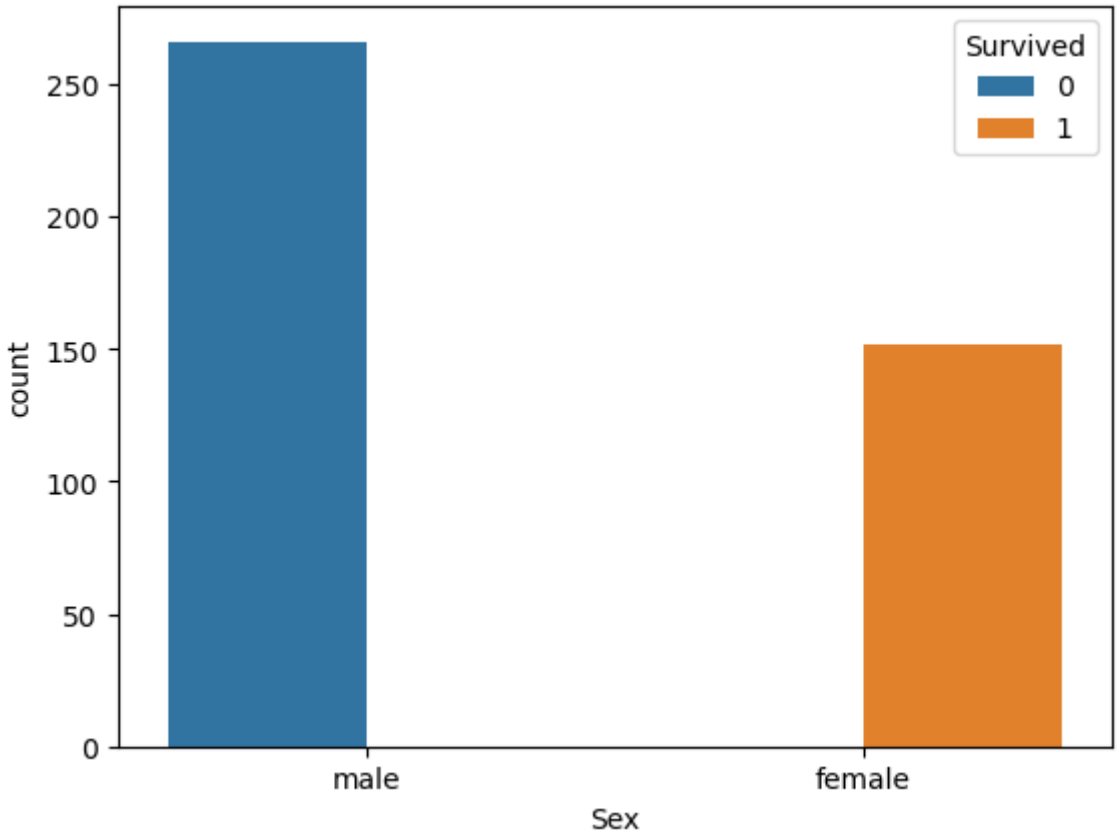
In [12]:
```
ds['Sex'].value_counts()
```

Out[12]:
```
male      266
female    152
Name: Sex, dtype: int64
```

Visualization of survival count wrt Gender

In [13]:
```
sbn.countplot(x=ds['Sex'], hue=ds['Survived'])
```

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



Survival Rate by Sex

In [17]:
```
ds.groupby(('Sex'))[['Survived']].mean()
```

Out[17]:

Sex	Survived
female	1.0
male	0.0

In [18]:
```
ds["Sex"].unique()
```

Out[18]:
```
array(['male', 'female'], dtype=object)
```

In [21]:
```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
```

In [22]:
```
ds['Sex']= labelencoder.fit_transform(ds['Sex'])
```

In [23]:
```
ds.head()
```

Out[23]:

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

In [25]:
```
ds['Sex']
```

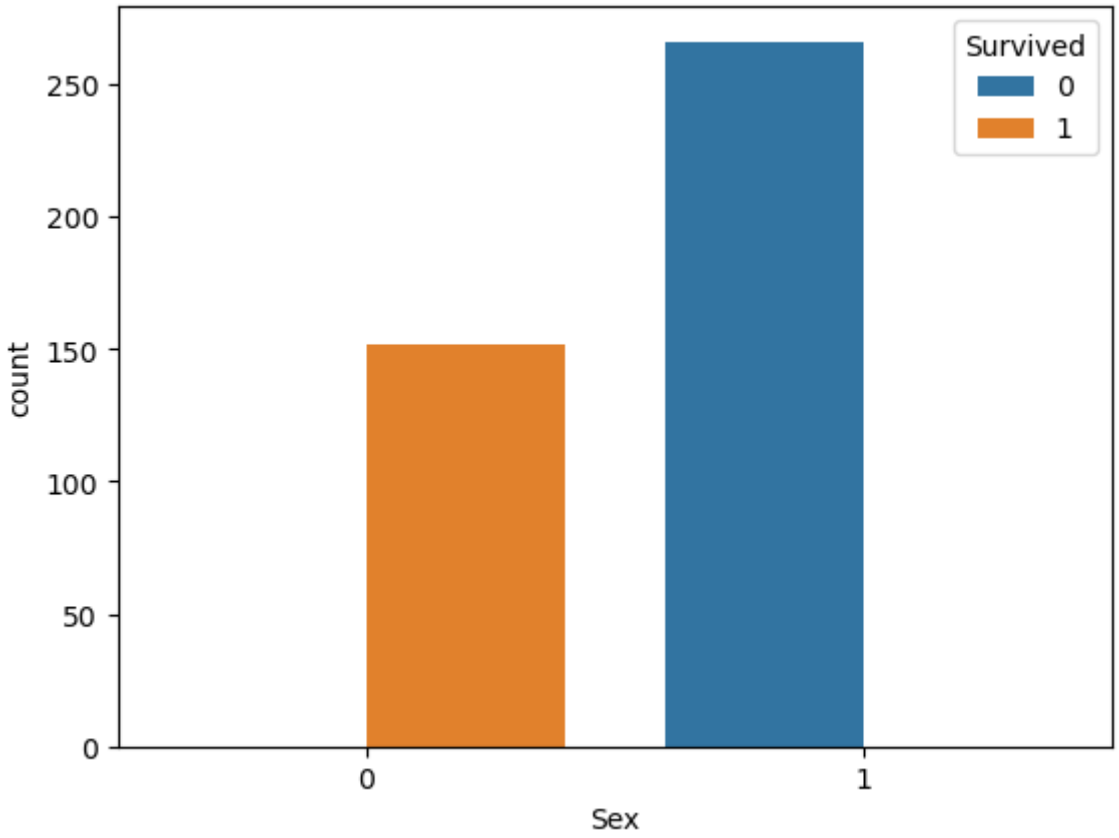
Out[25]:
```
0      1
1      0
2      1
3      1
4      0
      ..
413    1
414    0
415    1
416    1
417    1
Name: Sex, Length: 418, dtype: int32
```

In [26]:
```
ds['Survived']
```

Out[26]:
```
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
```

In [28]:
```
sbn.countplot(x=ds['Sex'], hue=ds['Survived'])
```

Out[28]:
```
<Axes: xlabel='Sex', ylabel='count'>
```



In [29]:
```
ds.isnull().sum()
```

Out[29]:
```
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
```

Dropping non requried Column

In [30]:
```
ds=ds.drop(['Age'], axis=1)
```

In [32]:
```
ds.tail()
```

Out[32]:

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

In [33]:
```
ds=ds.drop(['Cabin'], axis=1)
```

In [36]:
```
ds_final =ds
ds_final.head(15)
```

Out[36]:

	PassengerId	Survived	Pclass	Name	Sex	SibSp	Parch	Ticket	Fare	Embarked
0	892	0	3	Kelly, Mr. James	1	0	0	330911	7.8292	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	0	1	0	363272	7.0000	S
2	894	0	2	Myles, Mr. Thomas Francis	1	0	0	240276	9.6875	Q
3	895	0	3	Wirz, Mr. Albert	1	0	0	315154	8.6625	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	0	1	1	3101298	12.2875	S
5	897	0	3	Svensson, Mr. Johan Cervin	1	0	0	7538	9.2250	S
6	898	1	3	Connolly, Miss. Kate	0	0	0	330972	7.6292	Q
7	899	0	2	Caldwell, Mr. Albert Francis	1	1	1	248738	29.0000	S
8	900	1	3	Abrahim, Mrs. Joseph (Sophie Halaut Easu)	0	0	0	2657	7.2292	C
9	901	0	3	Davies, Mr. John Samuel	1	2	0	A/4 48871	24.1500	S
10	902	0	3	Ilieff, Mr. Ylio	1	0	0	349220	7.8958	S
11	903	0	1	Jones, Mr. Charles Cresson	1	0	0	694	26.0000	S
12	904	1	1	Snyder, Mrs. John Pillsbury (Nelle Stevenson)	0	1	0	21228	82.2667	S
13	905	0	2	Howard, Mr. Benjamin	1	1	0	24065	26.0000	S
14	906	1	1	Chaffee, Mrs. Herbert Fuller (Carrie Constance...	0	1	0	W.E.P. 5734	61.1750	S

MODEL TRAINING

In [37]:
```
X= ds[['Pclass','Sex']]
Y= ds['Survived']
```

In [38]:
```
from sklearn.model_selection import train_test_split
X_train ,X_test, Y_train, Y_test = train_test_split(X, Y , test_size=0.2 , random_state=0)
```

In [39]:
```
from sklearn.linear_model import LogisticRegression
```

In [41]:
```
logistic = LogisticRegression(random_state=0)
logistic.fit(X_train, Y_train)
```

Out[41]:
```
LogisticRegression(random_state=0)
```

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 PREDICTION

In [43]:
```
pred = print(logistic.predict(X_test))
```

```
[0 0 1 0 1 0 1 0 0 0 1 1 0 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 1 1 1 0 0
 1 1 1 1 0 1 1 0 1 0 0 0 0 0 1 1 0 0 1 0 1 0 0 0 1 1 0 0 1 1 1 1 0 0 1 1 1
 1 0 0 1 0 1 0 1 0 0]
```

In [44]:
```
print(Y_test)
```

```
360    0
170    0
224    1
358    0
309    1
      ..
100    1
7      0
22     1
68     0
328    0
Name: Survived, Length: 84, dtype: int64
```

In [59]:
```
import warnings
warnings.filterwarnings("ignore")

res= logistic.predict([[2,0]])
if(res==0):
    print(" So Sorry! Not Survived")
else:
    print("Survived")
```

```
Survived
```

In [ ]: