In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries inst
# It is defined by the kaggle/python Docker image: https://github.com/kaggle
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will l

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that
# You can also write temporary files to /kaggle/temp/, but they won't be sav
```

```
/kaggle/input/titanic-dataset/Titanic-Dataset.csv
```

In [2]:
```
df = pd.read_csv("/kaggle/input/titanic-dataset/Titanic-Dataset.csv")
```

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

Out[3]:

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

In [4]:
```
df.shape
```

Out[4]:
```
(891, 12)
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 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          714 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  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [6]:
```
df.isnull().sum() # Cheking of any NULl values
```

Out[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
```

In [7]:
```
df.duplicated().sum() # Cheking of any duplicated value
```

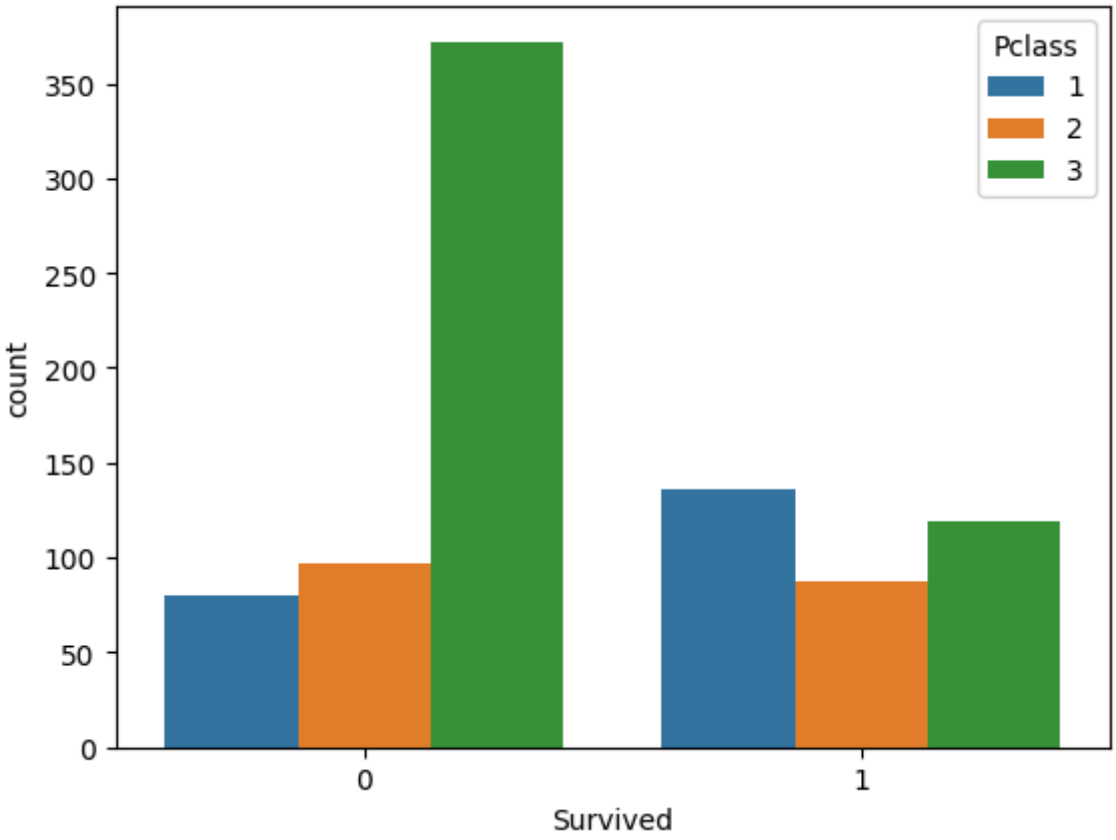
Out[7]:
```
0
```

In [8]:
```
Survived = df[df["Survived"]==1]
Non_Survived = df[df["Survived"]==0]
outlier = len(Survived)/float(len(Non_Survived))
print(outlier)
print("Survived : {} " .format(len(Survived)))
print("Non_Survived : {} " .format(len(Non_Survived)))
```

```
0.6229508196721312
Survived : 342 
Non_Survived : 549 
```

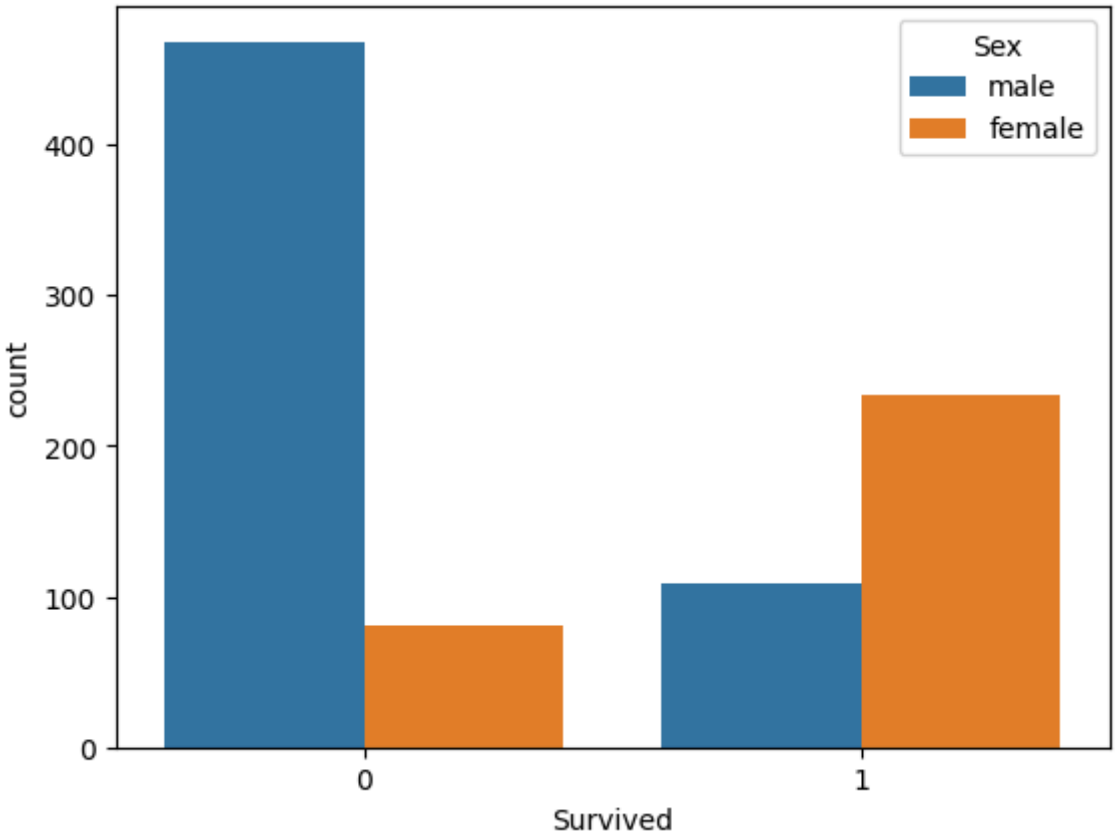
In [9]:
```
import seaborn as sns
sns.countplot(x= df["Survived"] , hue = df["Pclass"])
```

Out[9]: <Axes: xlabel='Survived', ylabel='count'>



In [10]:
```
sns.countplot(x= df["Survived"] , hue = df["Sex"])
```

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



In [11]:
```
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
```

In [12]:
```
labelencoder = LabelEncoder() # Conversion of Categorical values into Numer
df['Sex'] = labelencoder.fit_transform(df['Sex'])

df.head()
```

Out[12]:

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

In [13]:
```
features = df[["Pclass", "Sex", "Age", "SibSp", "Parch", "Fare"]]
target = df["Survived"]
```

In [14]:
```
df['Age'].fillna(df['Age'].median(), inplace=True)
```

```
/tmp/ipykernel_18/1933487976.py:1: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment using an
inplace method.
The behavior will change in pandas 3.0. This inplace method will never wor
k because the intermediate object on which we are setting values always be
haves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Age'].fillna(df['Age'].median(), inplace=True)
```

In [15]:
```
df.isnull().sum()
```

Out[15]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [16]:
```
x=df[['Pclass', 'Sex']]
y=target
```

In [17]:
```
x_train , x_test, y_train, y_test = train_test_split(features,y, test_size=

from sklearn.impute import SimpleImputer # It is used to fill the missing v
imputer = SimpleImputer(strategy='mean')
x_train_imputed = imputer.fit_transform(x_train)
x_test_imputed = imputer.transform(x_test)
```

In [18]:
```
model = RandomForestClassifier()
model.fit(x_train_imputed, y_train)
```

Out[18]:
```
RandomForestClassifier()
```
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.

In [19]:
```
predictions = model.predict(x_test_imputed)
```

In [20]:
```
from sklearn.metrics import accuracy_score, precision_score , recall_score
acc = accuracy_score(y_test , predictions)
print("The accuracy is {}".format(acc))

prec = precision_score(y_test , predictions)
print("The precision is {}".format(prec))

rec = recall_score(y_test , predictions)
print("The recall is {}".format(rec))

f1 = f1_score(y_test , predictions)
print("The F1-Score is {}".format(f1))
```

```
The accuracy is 0.8268156424581006
The precision is 0.8275862068965517
The recall is 0.6956521739130435
The F1-Score is 0.7559055118110236
```

In [21]:
```
import joblib
joblib.dump(model,"Titanic_Survival")
```

Out[21]:
```
['Titanic_Survival']
```

In [22]:
```
m = joblib.load("Titanic_Survival")
```

In [23]:
```
prediction  = m.predict([[1,1,0,1,1,1]])
prediction
```

Out[23]:
```
array([1])
```

In [24]:
```
if prediction==0:
  print("Non Survived")
else:
  print("Survived")
```

```
Survived
```

In [ ]:
```

```