Titanic survived Project

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import pearsonr

import warnings
warnings.filterwarnings("ignore")
```

In [5]:
```
df=pd.read_csv("https://raw.githubusercontent.com/dsrscientist/dataset1/master/ti
```

In [6]:
```
df
```

Out[6]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabi
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Na
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C8
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	Na
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C12
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	Na
...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	Na
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B4
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	Na
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C14
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	Na

891 rows × 12 columns

```
Data Dictionary

Passenger id- Unique Id of the passenger
Pclass- Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
Survived- Survived (0 = No; 1 = Yes)
Name- Name of the passenger
Sex- Sex of the passenger (Male, Female)
Age- Age of the passenger
Sibsp- Number of Siblings/Spouses Aboard
Parch- Number of Parents/Children Aboard
Ticket- Ticket Number
Fare- Passenger Fare (British pound)
Cabin- Cabin
```

```
Embarked- Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)
```

Data Preprocessing

In [7]:
```
df.shape
```

Out[7]:
```
(891, 12)
```

In [8]:
```
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 [9]:
```
df.isnull().sum()
```

Out[9]:
```
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 the missing values If the null value is more than 30% then drop the column. Else, just impute the column with mean.

In [10]:
```
df=df.drop(columns='Cabin',axis=1)
df.head()
```

Out[10]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embar
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 [11]:
```
#Replacing the missing values in "Age" column with mean

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

In [13]:
```
#Finding the mode value of "Embarked" column

print(df['Embarked'].mode())
```

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

In [14]:
```
print(df['Embarked'].mode()[0]) # 0 in the index
```

```
S
```

In [15]:
```
# Replacing the missing values in "Embarked" column with the mode value

df['Embarked'].fillna(df['Embarked'].mode()[0],inplace=True)
```

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

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

Data Analysis

In [17]:
```
df.describe()
```

Out[17]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	13.002015	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	22.000000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	29.699118	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	35.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

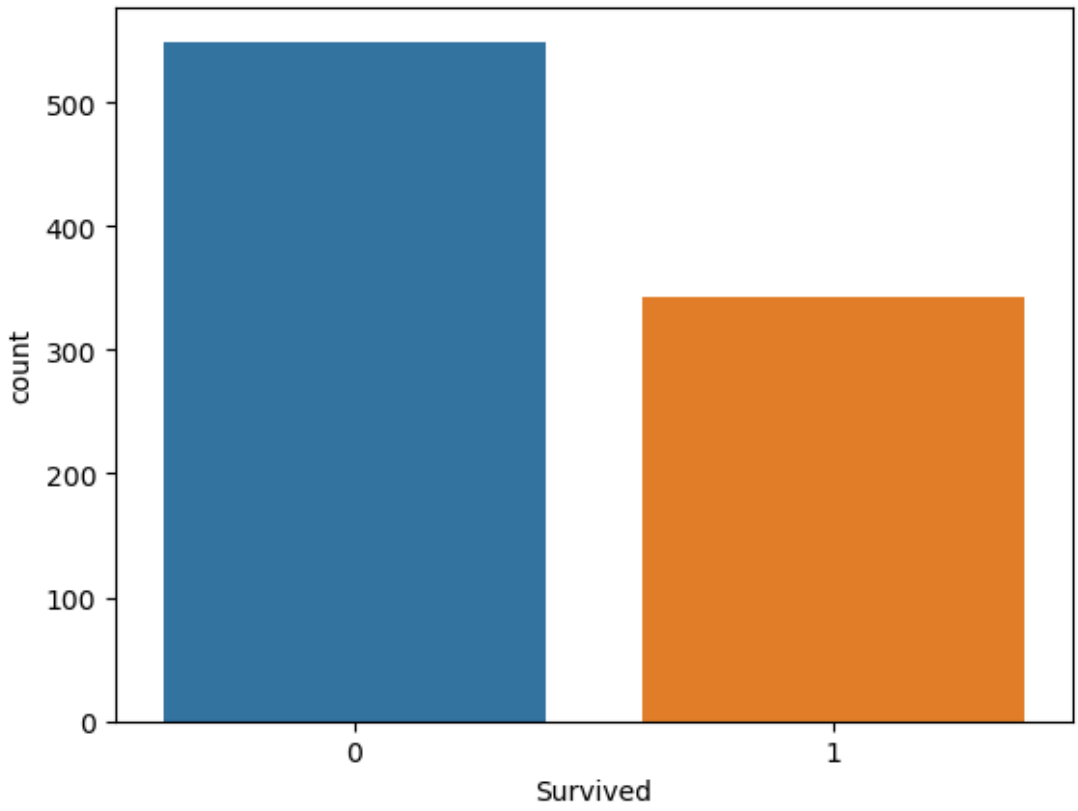
In [18]:
```
df['Survived'].value_counts()
```

Out[18]:
```
Survived
0    549
1    342
Name: count, dtype: int64
```

Data Visualization

In [19]:
```
sns.countplot(x='Survived',data=df)
```

Out[19]:
```
<Axes: xlabel='Survived', ylabel='count'>
```

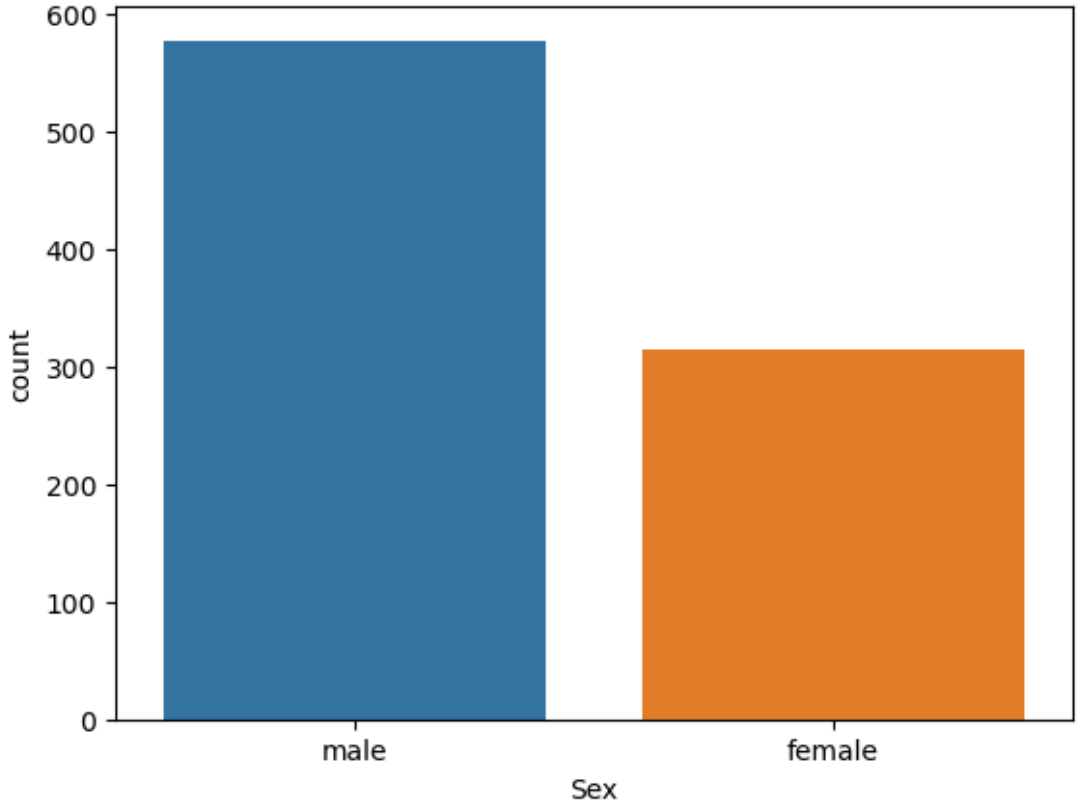


In [20]:
```
df['Sex'].value_counts()
```

Out[20]:
```
Sex
male      577
female    314
Name: count, dtype: int64
```

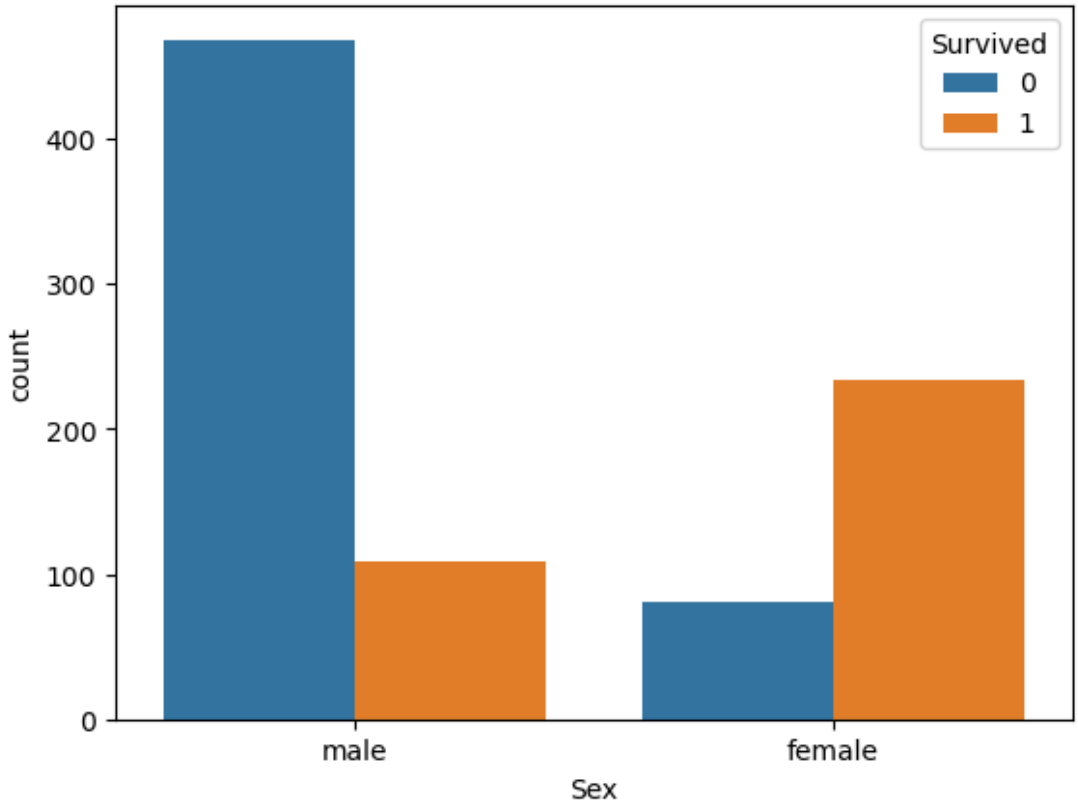
In [21]:
```
sns.countplot(x='Sex',data=df)
```

Out[21]: <Axes: xlabel='Sex', ylabel='count'>



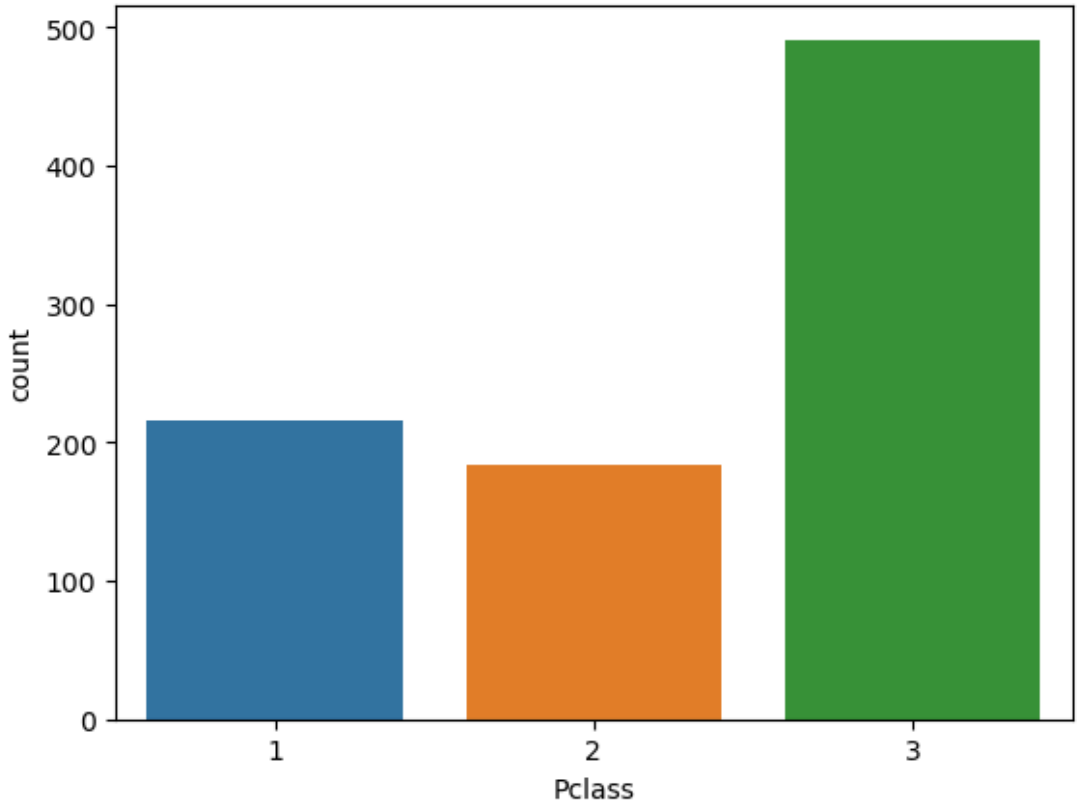
In [22]:
```
sns.countplot(x="Sex", hue="Survived", data= df)
```

Out[22]: <Axes: xlabel='Sex', ylabel='count'>



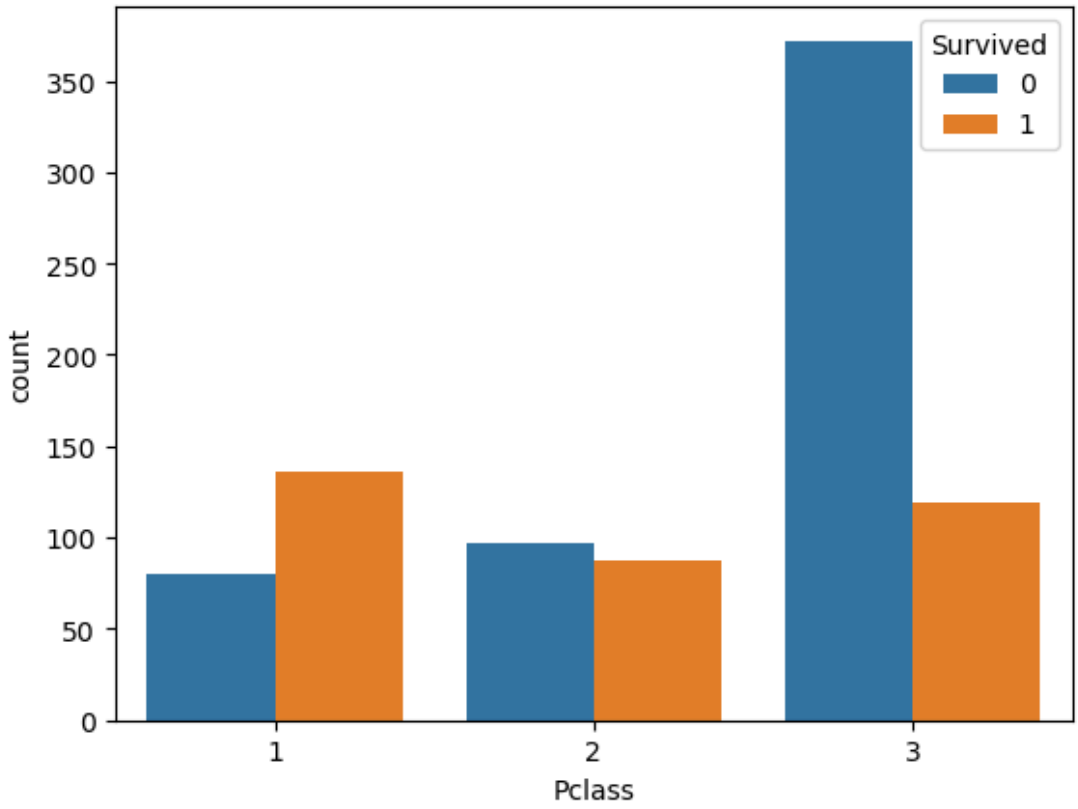
In [23]:
```
sns.countplot(x="Pclass", data=df)
```

Out[23]: <Axes: xlabel='Pclass', ylabel='count'>



In [24]:
```
sns.countplot(x="Pclass", hue="Survived", data=df)
```

Out[24]: <Axes: xlabel='Pclass', ylabel='count'>



Encoding the Categorical Columns

In [25]:
```
df["Sex"].value_counts()
```

Out[25]:
```
Sex
male      577
female    314
Name: count, dtype: int64
```

In [27]:
```
df["Embarked"].value_counts()
```

Out[27]:
```
Embarked
S    646
C    168
Q     77
Name: count, dtype: int64
```

In [39]:
```
df.replace({'Sex':{'male':0,'female':1},'Embarked':{'S':0,'C':1,'Q':2}},inplace=T
```

In [40]:
```
df.head()
```

Out[40]:

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

In [41]:
```
X=df.drop(columns=["PassengerId","Name","Ticket","Survived"],axis=1)
Y=df["Survived"]
```

In [42]:
```
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 [43]:
```
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
```

Training and Testing Split

In [44]:
```
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=2)
```

In [45]:
```
print(X.shape,X_train.shape,X_test.shape)
```

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

Model Training

In [46]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,classification_repor
```

Logistic Regression

In [47]:
```
model=LogisticRegression()
model.fit(X_train, Y_train)
```

Out[47]:
```
LogisticRegression
LogisticRegression()
```

In [48]:
```
model_prediction=model.predict(X_test)
```

In [49]:
```
accuracy_score(model_prediction,Y_test)
```

Out[49]:
```
0.7821229050279329
```

Model Evaluation

In [50]:
```
confusion_matrix(Y_test, model_prediction)
```

Out[50]:
```
array([[91,  9],
       [30, 49]], dtype=int64)
```

Combining all the model score

In [51]:
```
results=pd.DataFrame({'Model':['Logistic Regression'], 'Score':[0.78]})
```

In [52]:
```
results
```

Out[52]:

	Model	Score
0	Logistic Regression	0.78

Model Prediction

In [53]:
```
model_prediction
```

Out[53]:
```
array([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], dtype=int64)
```

Model Building and Deployment

In [54]:
```
# To save the model in a pkl file

import pickle as pkl

pkl.dump(model,open('model.pkl','wb'))
```

In [55]:
```
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 [56]:
```
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
```

In [57]:
```
X_train.iloc[0,:]
```

Out[57]:
```
Pclass       1.0000
Sex          0.0000
Age         40.0000
SibSp        0.0000
Parch        0.0000
Fare        27.7208
Embarked     1.0000
Name: 30, dtype: float64
```

In [59]:
```
a=list(X_train.iloc[0,:])
a=np.array(a)
```

In [60]:
```
ypred=model.predict(a.reshape(-1,7))
ypred
```

Out[60]:
```
array([0], dtype=int64)
```

In [61]:
```
Y_train[0]
```

Out[61]:
```
0
```

Infrence

In [62]:
```
loaded_model=pkl.load(open('model.pkl','rb'))
```

In [63]:
```
type(loaded_model)
```

Out[63]:
```
sklearn.linear_model._logistic.LogisticRegression
```

In [64]:
```
ypred=loaded_model.predict(a.reshape(-1,7))
```

In [65]:
```
ypred
```

Out[65]:
```
array([0], dtype=int64)
```

In [ ]: