```
# importing the dependencies
```

```
import numpy as np
import pandas as pd
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
```

```
# data collection & processing
```

```
#load th data from csv file to pandas dataframe
titanic_data = pd.read_csv('/Titanic-Dataset (3).csv')
```

```
# print the first 5 rows
titanic_data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	5	0	3	Allen, Mr. William	male	35.0	0	0	373450	8.0

```
#rows and columns
titanic_data.shape
```

```
(891, 12)
```

```
# get some info about the data
titanic_data.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
```

```
# check the number of missing values from each column
titanic_data.isnull().sum()
```

```
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
```

handalling the missing values in a data set

```
#drop the "cabin" column
titanic_data = titanic_data.drop(columns='Cabin', axis=1)
```

```
#replacing the age column with mean value
titanic_data['Age'].fillna(titanic_data['Age'].mean() , inplace=True)
```

```
# finding the mode value of "Embarked" column
print(titanic_data['Embarked'].mode())
```

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

```
# replacing the missing value of Embarked column with mode value
titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0] , inplace=True)
```

```
# check the number of missing values from each column
titanic_data.isnull().sum()
```

```
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

```
#getting some stasistis about the data
titanic_data.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	F
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204
std	257.353842	0.486592	0.836071	13.002015	1.102743	0.806057	49.693
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000
25%	223.500000	0.000000	2.000000	22.000000	0.000000	0.000000	7.910
50%	446.000000	0.000000	3.000000	29.699118	0.000000	0.000000	14.454
75%	668.500000	1.000000	3.000000	35.000000	1.000000	0.000000	31.000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329

```
# finding the number of people survived and not survived
titanic_data['Survived'].value_counts()
```

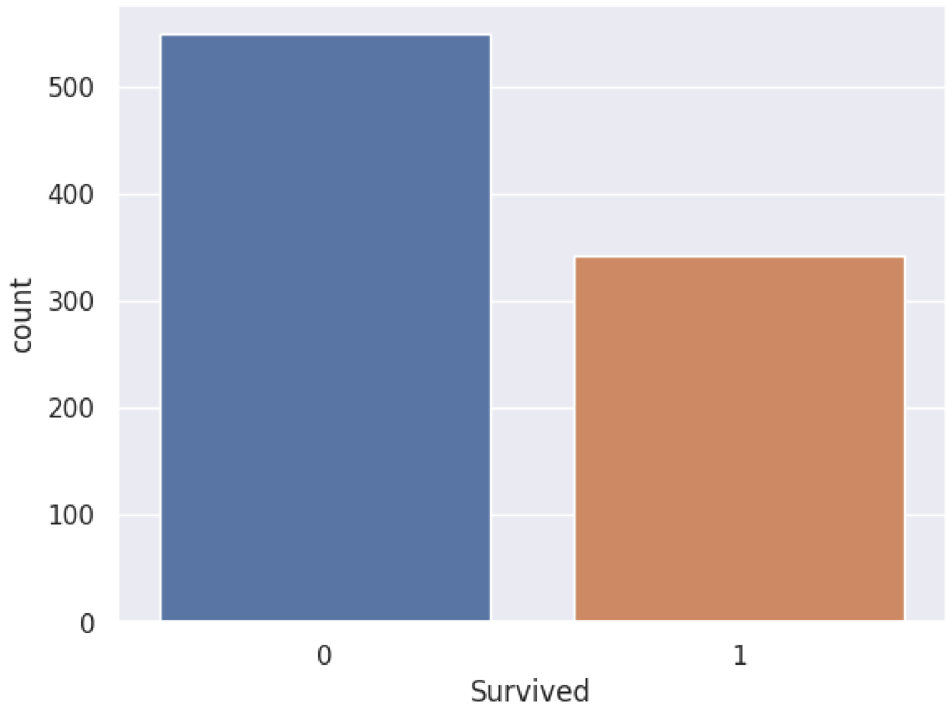
```
0    549
1    342
Name: Survived, dtype: int64
```

Data Visualization

```
sns.set()
```

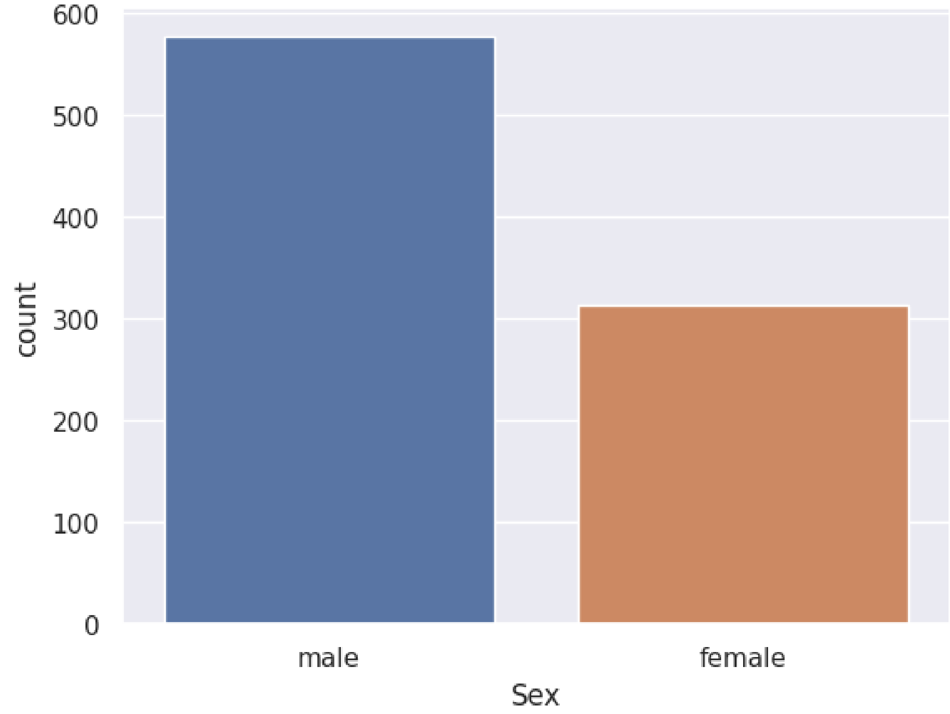
```
# making a count plot for "survived" column
sns.countplot(x='Survived',data=titanic_data)
```

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




```
# making a count plot for "Sex" column
sns.countplot(x='Sex',data=titanic_data)
```

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

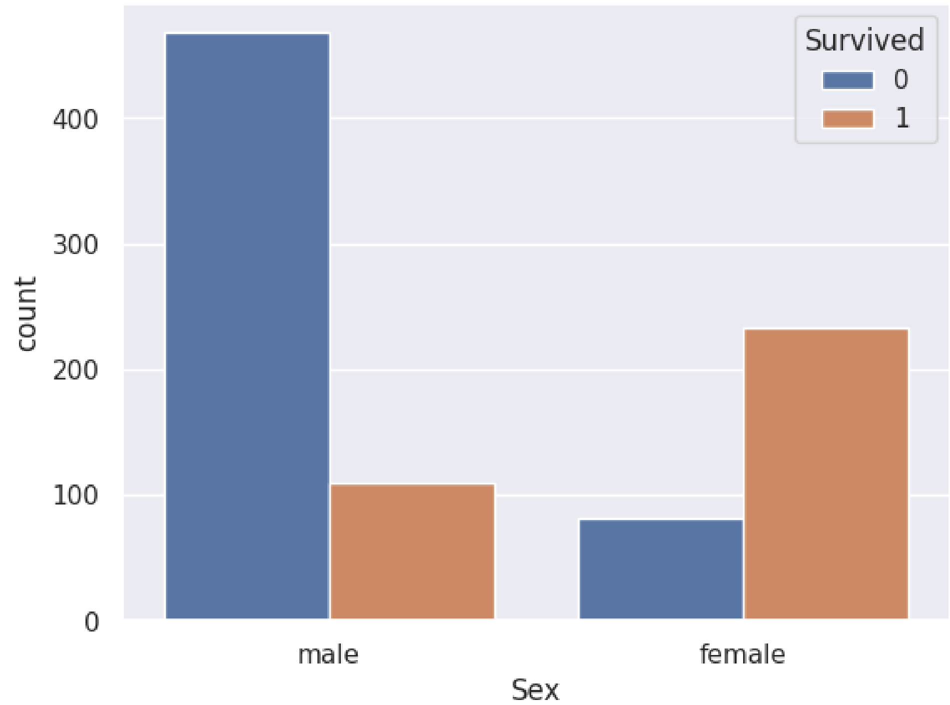



```
titanic_data['Sex'].value_counts()
```

```
male      577
female    314
Name: Sex, dtype: int64
```

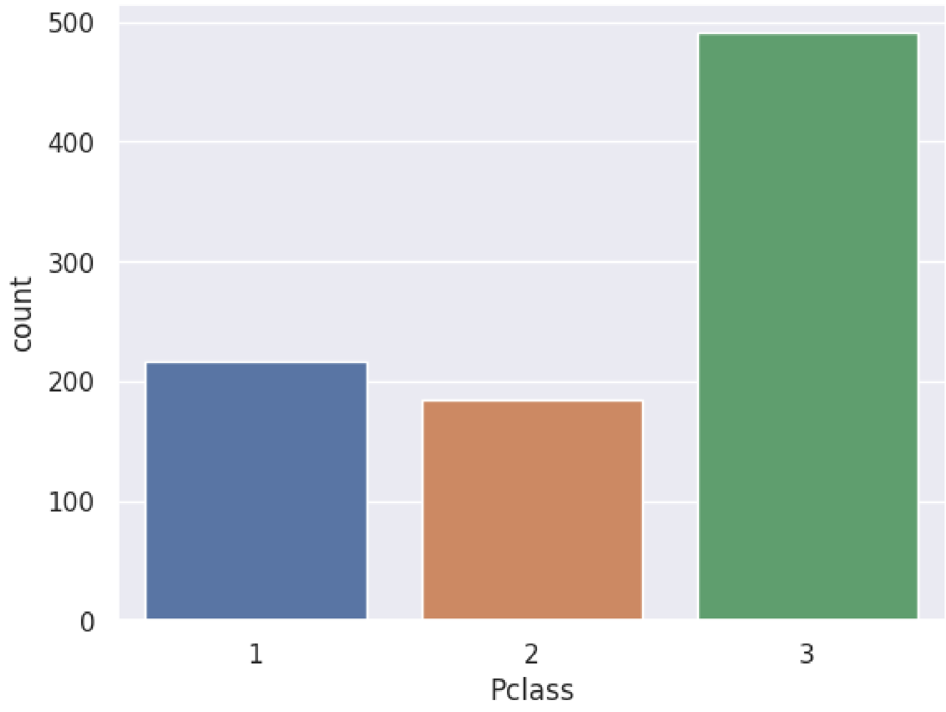
```
# number of survivors gender wise
sns.countplot(x='Sex', hue='Survived', data=titanic_data)
```

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



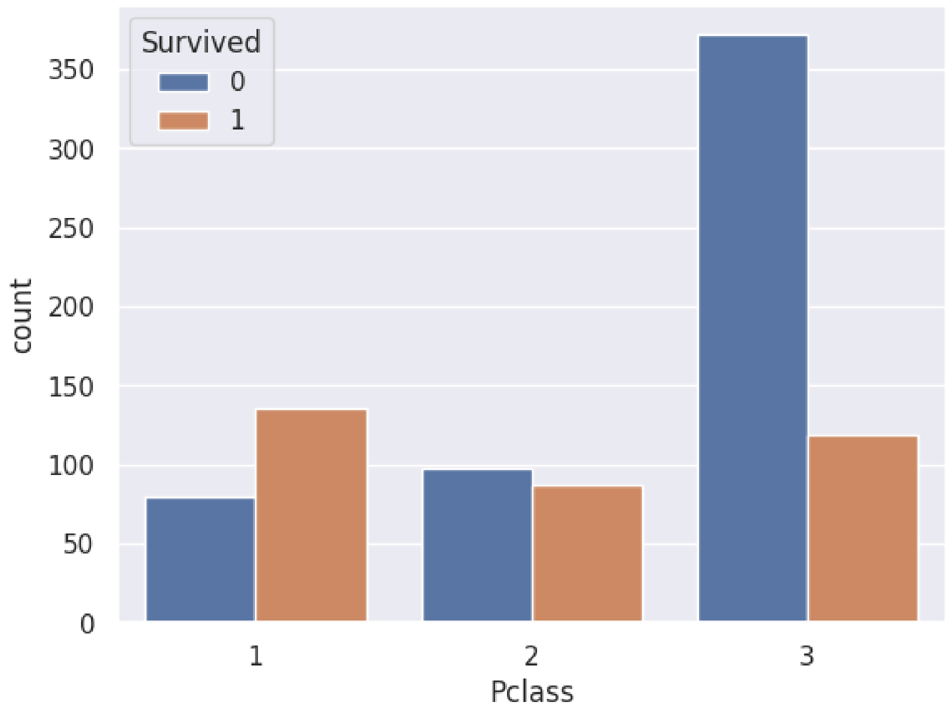

```
# making a count plot for "Pclass" column
sns.countplot(x='Pclass',data=titanic_data)
```

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




```
sns.countplot(x='Pclass', hue='Survived', data=titanic_data)
```

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



encoding the categorical column

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

```
male      577
female    314
Name: Sex, dtype: int64
```

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

```
S    646
C    168
Q     77
Name: Embarked, dtype: int64
```

```
#converting the categorical column
titanic_data.replace({'Sex':{'male':0,'female':1},'Embarked':{'S':0,'C':1,'Q':2}},inplace= True)
```

```
titanic_data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0	3	Braund, Mr. Owen Harris	0	22.0	1	0	A/5 21171	7.250
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.0	1	0	PC 17599	71.283
2	3	1	3	Heikkinen, Miss. Laina	1	26.0	0	0	STON/O2. 3101282	7.925

Separating features & Target

```
X = titanic_data.drop(columns= ['PassengerId','Name','Ticket','Survived'],axis=1)
Y = titanic_data['Survived']
```

```
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]
```

```
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 data and test data

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2, random_state=2)
```

```
print(X.shape, X_train.shape, X_test.shape)
```

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

- Model Training
- Logistic Regression

```
model = LogisticRegression()
```

```
# training the logistic regression model with training data
model.fit(X_train, Y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Convei
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-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-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()
```

- Model Evaluation
- Accuracy Score

```
#accuracy on traning data
X_train_prediction = model.predict(X_train)
```

```
print(X_train_prediction)
```

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

```
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
print("Accuracy score of training data : ", training_data_accuracy)
```

```
Accuracy score of training data :  0.8075842696629213
```

```
# accuracy on test data
X_test_prediction = model.predict(X_test)
```

```
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]
```

```
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
print("Accuracy score of training data : ", test_data_accuracy)
```

```
Accuracy score of training data :  0.7821229050279329
```