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 the data from csv file to pandas DataFrame#

vedika = pd.read_csv(r'C:\Users\hp\Desktop\files.csv\train.csv')

vedika.head(10)
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3
5            6         0       3
6            7         0       1
7            8         0       3
8            9         1       3
9           10         1       2

                                                Name     Sex   Age
SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0
1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
1
2                             Heikkinen, Miss. Laina  female  26.0
0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
1
4                           Allen, Mr. William Henry    male  35.0
0
5                                   Moran, Mr. James    male   NaN
0
6                            McCarthy, Mr. Timothy J    male  54.0
0
```

```
7                       Palsson, Master. Gosta Leonard    male   2.0
3
8  Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.0
0
9                Nasser, Mrs. Nicholas (Adele Achem)  female  14.0
1

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
5      0            330877   8.4583   NaN        Q
6      0             17463  51.8625   E46        S
7      1            349909  21.0750   NaN        S
8      2            347742  11.1333   NaN        S
9      0            237736  30.0708   NaN        C
```

```
vedika.tail()
```

```
     PassengerId  Survived  Pclass
Name  \
886          887         0       2                     Montvila, Rev.
Juozas
887          888         1       1              Graham, Miss. Margaret
Edith
888          889         0       3  Johnston, Miss. Catherine Helen
"Carrie"
889          890         1       1                     Behr, Mr. Karl
Howell
890          891         0       3                       Dooley, Mr.
Patrick

        Sex   Age  SibSp  Parch      Ticket   Fare Cabin Embarked
886    male  27.0      0      0      211536  13.00   NaN        S
887  female  19.0      0      0      112053  30.00   B42        S
888  female   NaN      1      2  W./C. 6607  23.45   NaN        S
889    male  26.0      0      0      111369  30.00  C148        C
890    male  32.0      0      0      370376   7.75   NaN        Q
```

```
# getting some information about the data

vedika.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 in each column

vedika.isnull()
```

```
     PassengerId  Survived  Pclass   Name    Sex    Age  SibSp  Parch
Ticket  \
0          False     False   False  False  False  False  False  False
False
1          False     False   False  False  False  False  False  False
False
2          False     False   False  False  False  False  False  False
False
3          False     False   False  False  False  False  False  False
False
4          False     False   False  False  False  False  False  False
False
..           ...       ...     ...    ...    ...    ...    ...    ...
...
886        False     False   False  False  False  False  False  False
False
887        False     False   False  False  False  False  False  False
False
888        False     False   False  False  False   True  False  False
False
889        False     False   False  False  False  False  False  False
False
890        False     False   False  False  False  False  False  False
False

      Fare  Cabin  Embarked
0    False   True     False
1    False  False     False
2    False   True     False
3    False  False     False
4    False   True     False
..     ...    ...       ...
```

```
886  False   True     False
887  False  False     False
888  False   True     False
889  False  False     False
890  False   True     False

[891 rows x 12 columns]
```

```
vedika.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
```

```
vedika.describe()
```

```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

Handling the Missing values

```
# drop the'Cabin' column from the dataframe
```

```
vedika = vedika.drop('Cabin',axis=1)

vedika.isnull().sum()

PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Embarked         2
dtype: int64

# number of rows and colums

vedika.shape

(891, 11)

# Replacing the missing values in 'Age' column with mean value

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

# Finding the mode value of 'Embarked' column

print(vedika['Embarked'].mode())

0    S
Name: Embarked, dtype: object

print(vedika['Embarked'].mode()[0])

S

# Replacing the missing value in the 'Embarked' column with mode value


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

#check the number of missing values in each column
vedika.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
```

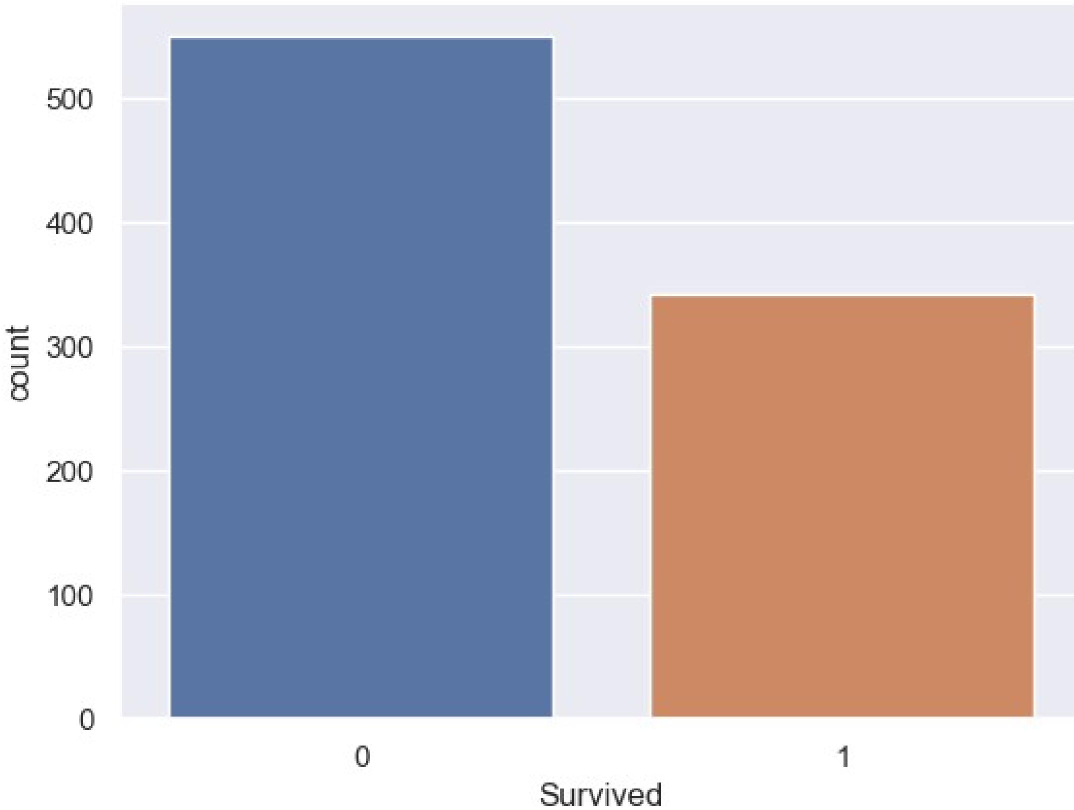
Data Analysis

```
# Getting some statistical measures about the data

vedika.describe()

       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  891.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   13.002015    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   22.000000    0.000000
50%     446.000000    0.000000    3.000000   29.699118    0.000000
75%     668.500000    1.000000    3.000000   35.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200

# Finding the number of people survived and not survived

vedika['Survived'].value_counts()

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

Data Visualization

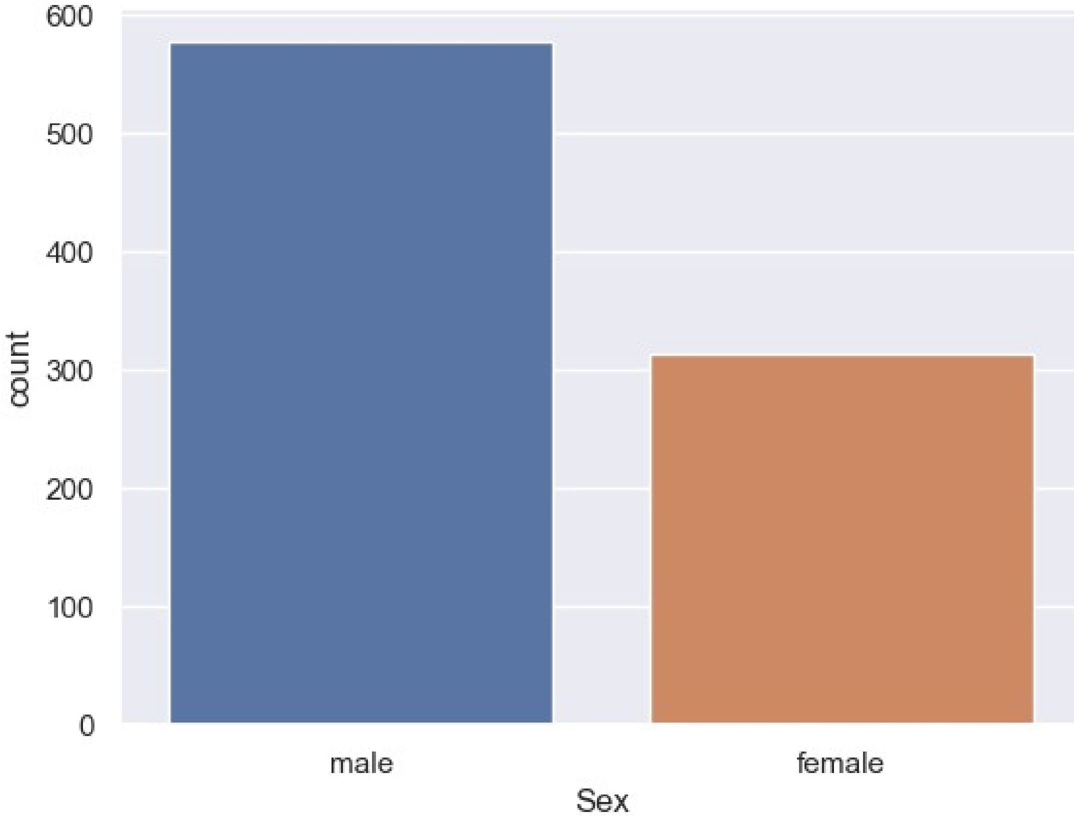
```
sns.set()

# Making a count plot for 'Survived' column
sns.countplot(x='Survived', data=vedika)
```

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

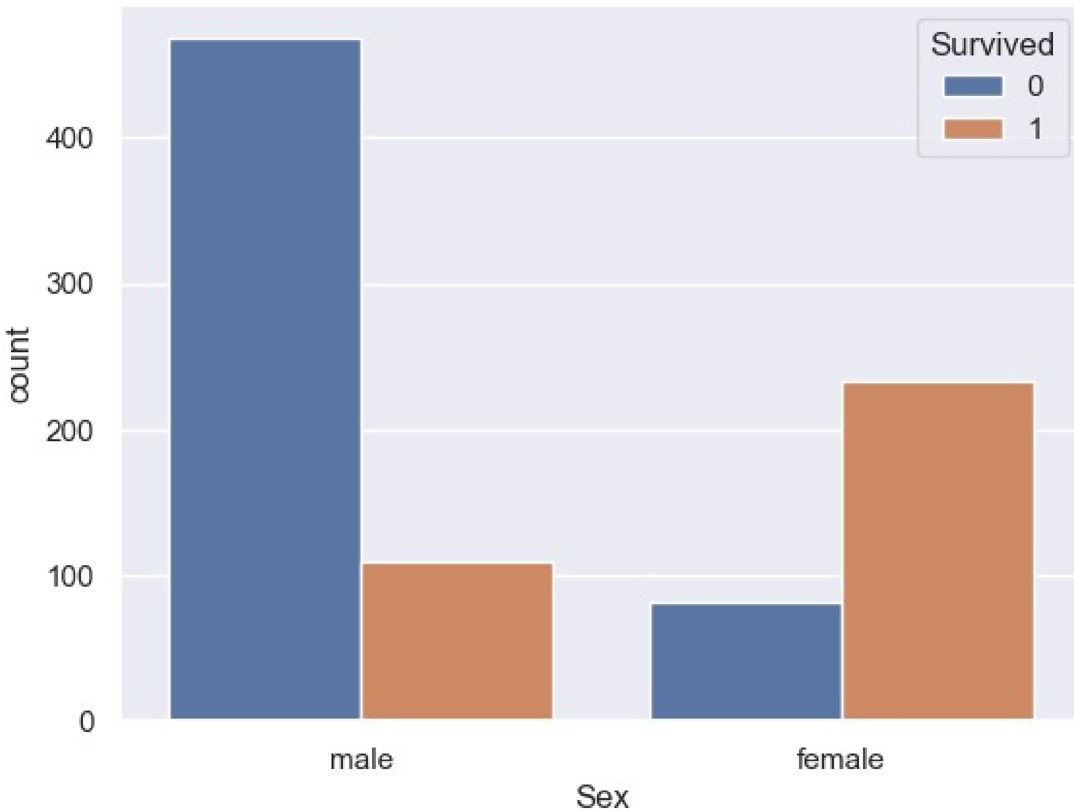



```
# Making a count plot for 'Sex' column
sns.countplot(x='Sex', data=vedika)
```

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

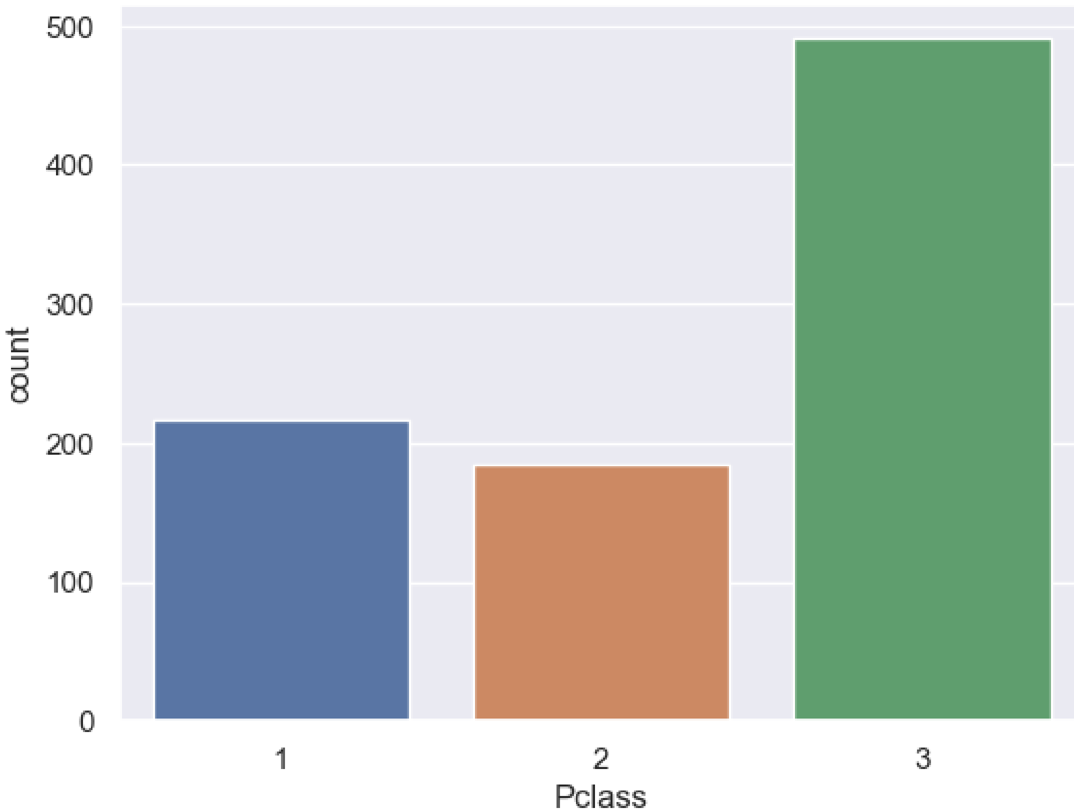



```
# Making a count plot for 'Sex' column

vedika['Sex'].value_counts()

male      577
female    314
Name: Sex, dtype: int64

# Number of survivors Gender wise
sns.countplot(x='Sex', hue='Survived', data=vedika)

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



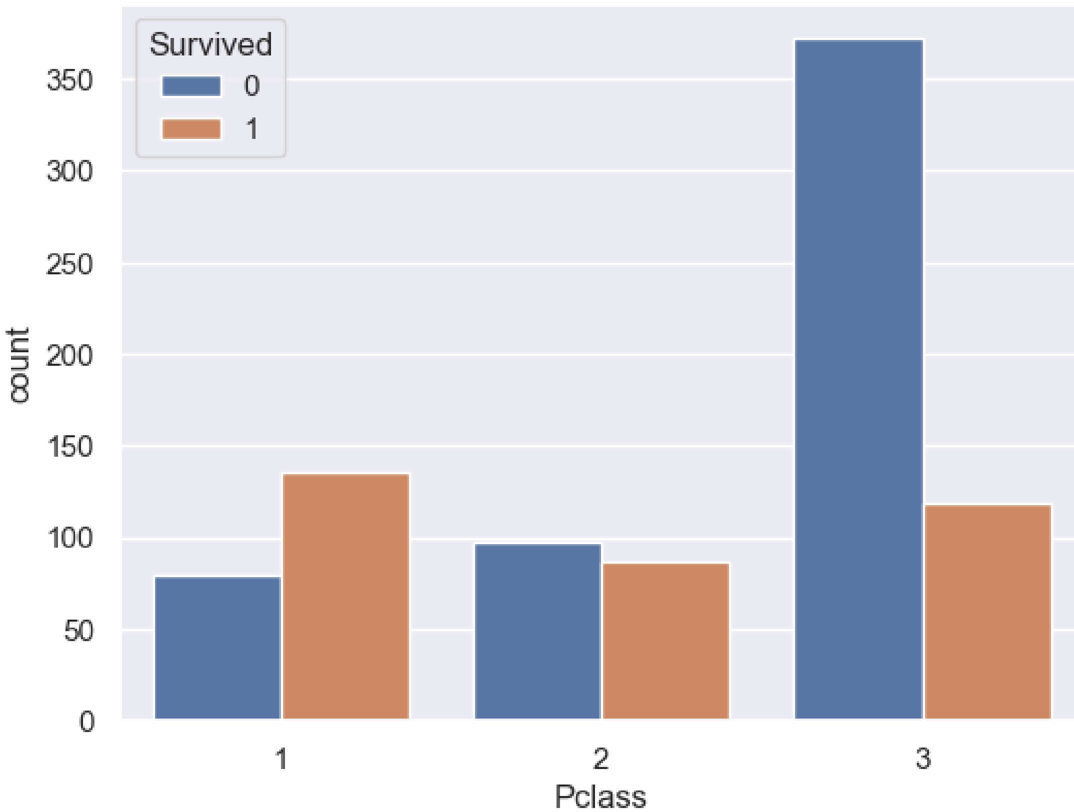

```
# Making a count plot for 'Pclass' column
sns.countplot(x='Pclass', data=vedika)
```

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




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

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



Encoding the Categorical Columns

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

male      577
female    314
Name: Sex, dtype: int64

vedika['Embarked'].value_counts()

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

# Converting categorical columns

vedika.replace({'Sex':{'male':0,'female':1}, 'Embarked':
{'S':0,'C':1,'Q':2}}, inplace=True)

vedika.head()

   PassengerId  Survived  Pclass  \
0            1         0       3
```

```
1            2          1        1
2            3          1        3
3            4          1        1
4            5          0        3

                                                Name  Sex   Age  SibSp
Parch  \
0                            Braund, Mr. Owen Harris    0  22.0      1
0
1  Cumings, Mrs. John Bradley (Florence Briggs Th...    1  38.0      1
0
2                             Heikkinen, Miss. Laina    1  26.0      0
0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)    1  35.0      1
0
4                           Allen, Mr. William Henry    0  35.0      0
0

             Ticket     Fare  Embarked
0         A/5 21171   7.2500         0
1          PC 17599  71.2833         1
2  STON/O2. 3101282   7.9250         0
3            113803  53.1000         0
4            373450   8.0500         0
```

Separating Features & Target

```
X = vedika.drop(columns=['PassengerId', 'Name', 'Ticket', 'Survived'],
axis=1)
y = vedika['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 & 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)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\
_logistic.py:460: 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 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()
```

Model Evaluation

Accuracy Score

```python
# Accuracy on training 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]
```

```python
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
```

```python
# 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]
```

```python
test_data_accuracy = accuracy_score(y_test, X_test_prediction)
print('Accuracy score of test data :', test_data_accuracy)
```

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