

# assignment-3

21BCE3132

September 21, 2023

**1 Aman kumar 21BCE3132**

## IMPORTING THE LIBRARIES

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

## IMPORTING THE DATASET

```
[2]: data=pd.read_csv("Titanic-Dataset.csv")
```

```
[3]: data.head()
```

```
[3]: 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 \
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
```

```
    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
```

```
[4]: 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

```

```
[5]: data.shape
```

```
[5]: (891, 12)
```

```
[6]: data.describe()
```

```

[6]:      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

```

### CHECKING FOR NULL VALUES

```
[7]: data.isnull().any()
```

```
[7]: PassengerId False
      Survived      False
      Pclass        False
      Name          False
      Sex           False
      Age           True
      SibSp         False
      Parch         False
      Ticket        False
      Fare          False
      Cabin         True
      Embarked      True
      dtype: bool
```

```
[8]: data.isnull().sum()
```

```
[8]: 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
```

```
[9]: data.corr()
```

<ipython-input-9-c44ded798807>:1: FutureWarning: The default value of numeric\_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric\_only to silence this warning.

```
data.corr()
```

```
[9]:
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	\
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-	
							0.001652
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322		
							0.081629
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081		
							0.018443
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-	
							0.189119

```

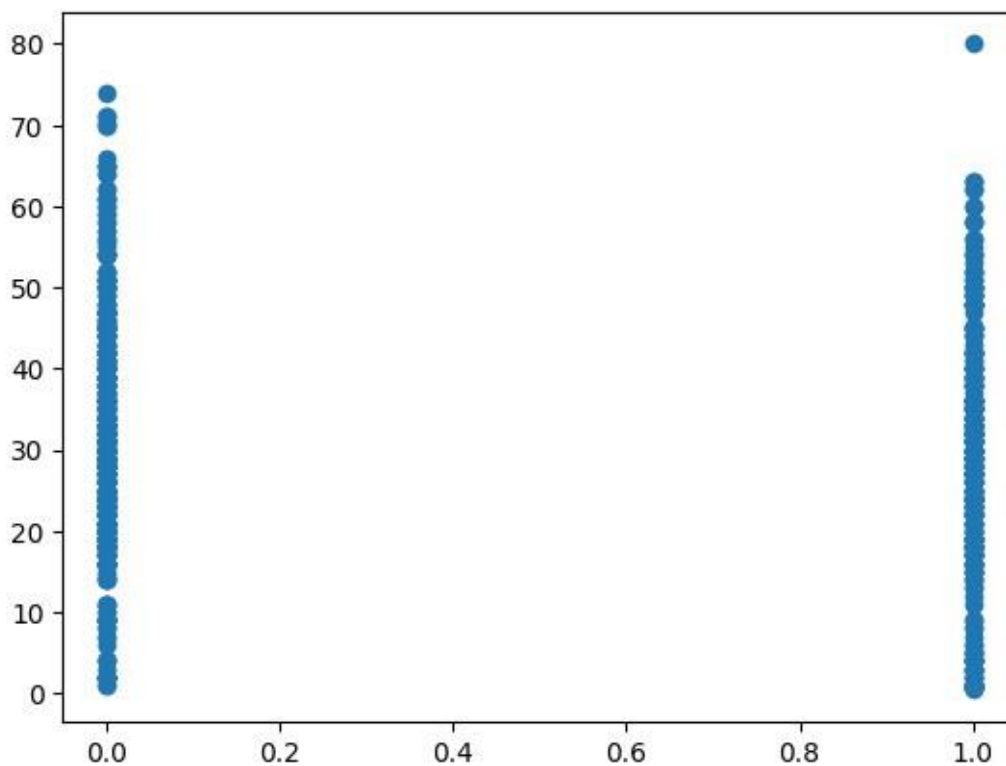
SibSp      -0.057527 -0.035322 0.083081 -0.308247 1.000000
           0.414838
Parch      -0.001652 0.081629 0.018443 -0.189119 0.414838
           1.000000
Fare        0.012658 0.257307 -0.549500 0.096067 0.159651 0.216225
           Fare
PassengerId 0.012658
Survived    0.257307
Pclass      -
           0.549500
Age         0.096067
SibSp       0.159651
Parch       0.216225
Fare        1.000000

```

### DATA VISUALIZATION

```
[10]: plt.scatter(data["Survived"], data["Age"])
```

```
[10]: <matplotlib.collections.PathCollection at 0x7b8530220400>
```

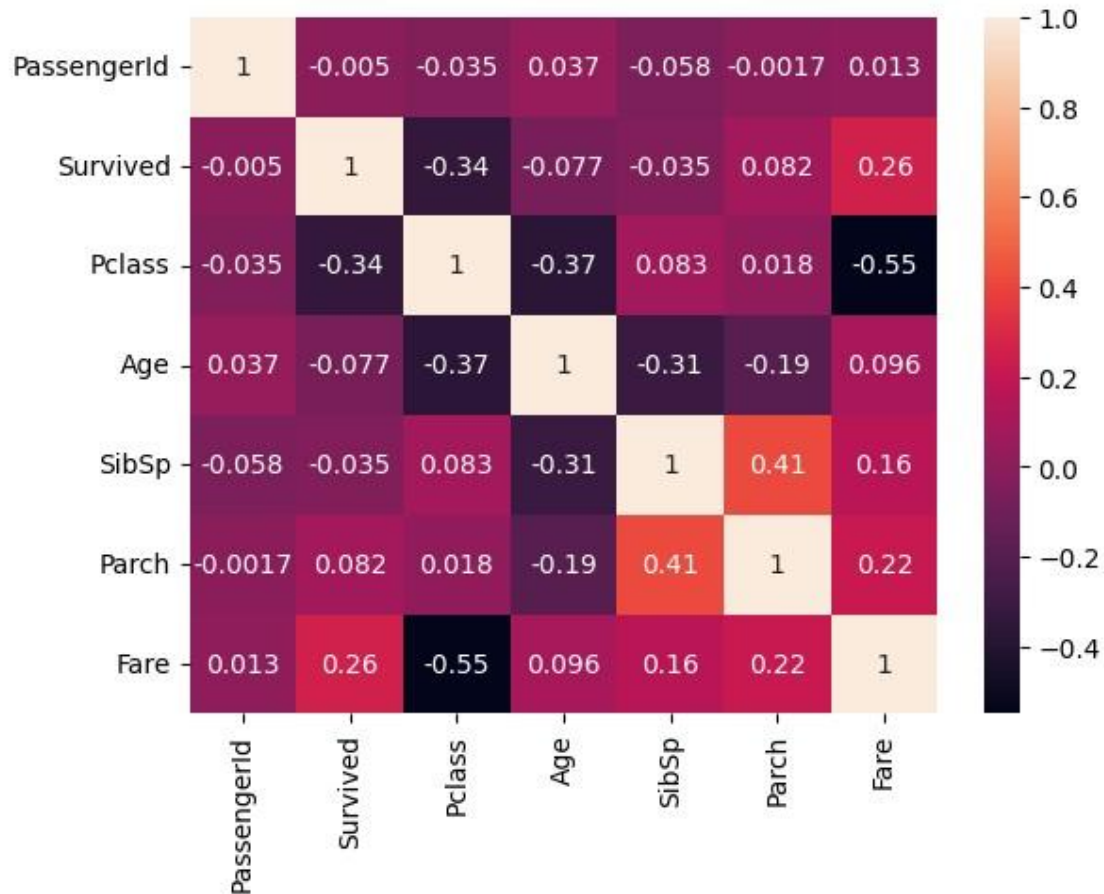


```
[11]: sns.heatmap(data.corr(), annot=True)
```

<ipython-input-11-6c71ac866e2e>:1: FutureWarning: The default value of numeric\_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric\_only to silence this warning.

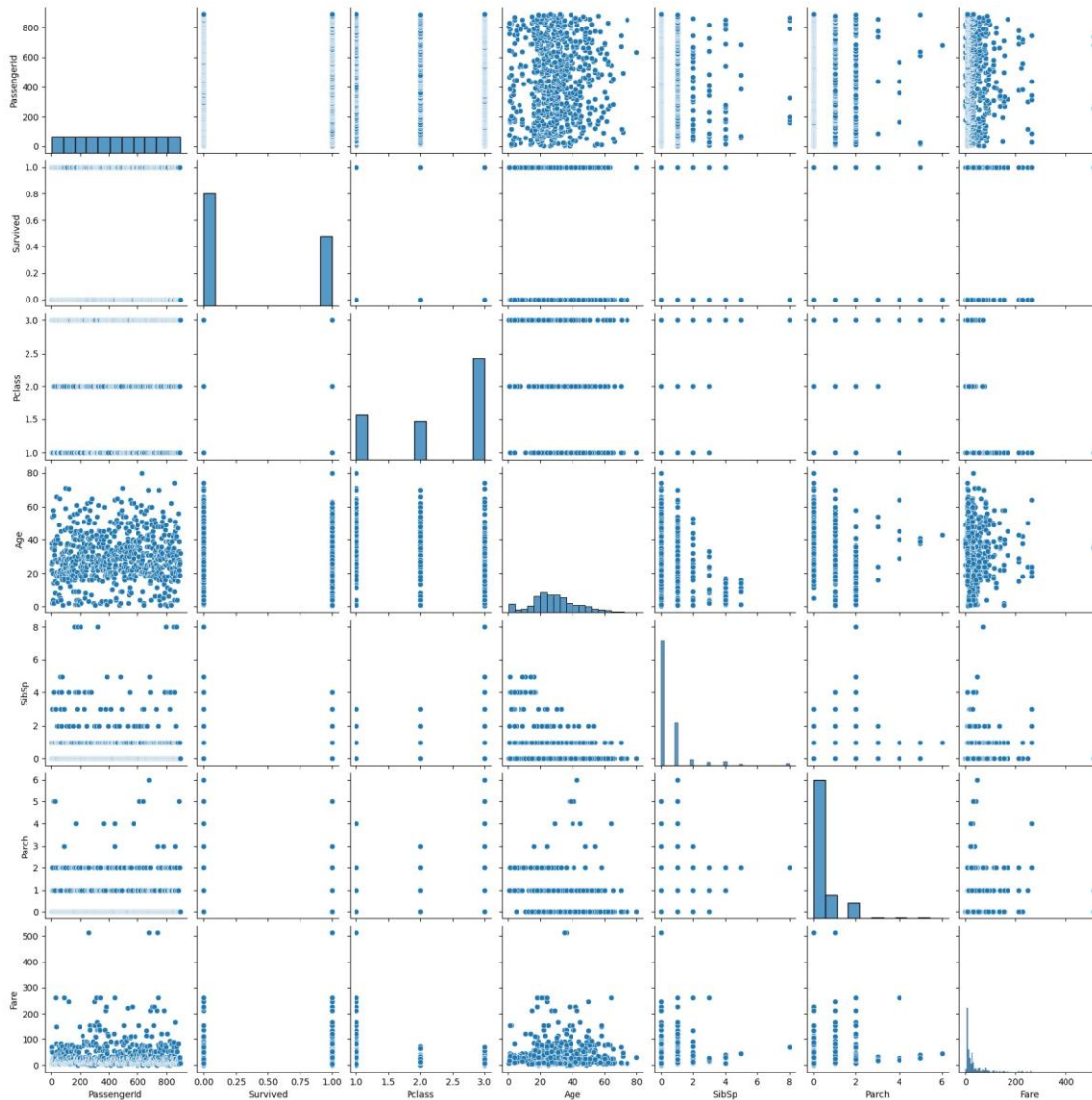
```
sns.heatmap(data.corr(),annot=True)
```

[11]: <Axes: >



```
[12]: sns.pairplot(data)
```

[12]: <seaborn.axisgrid.PairGrid at 0x7b852e0849a0>



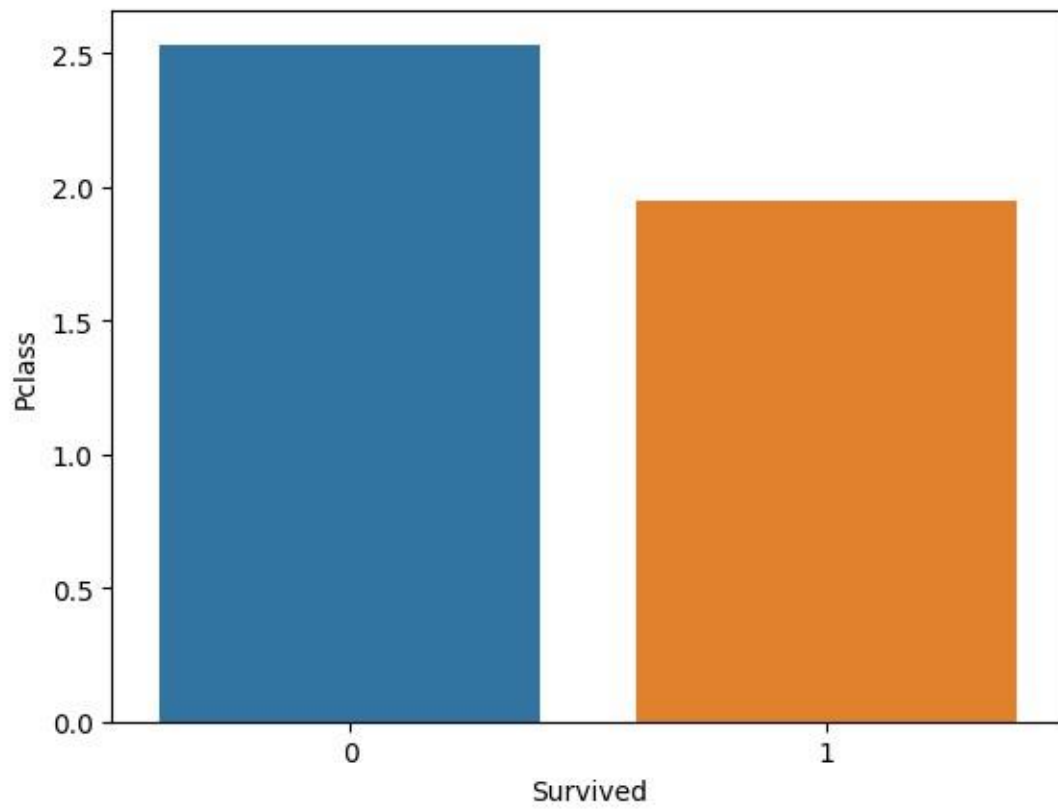
```
[13]: sns.barplot(x=data["Survived"],y=data["Pclass"],ci=0)
```

<ipython-input-13-0a7b02de1ba7>:1: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=('ci', 0)` for the same effect.

```
sns.barplot(x=data["Survived"],y=data["Pclass"],ci=0)
```

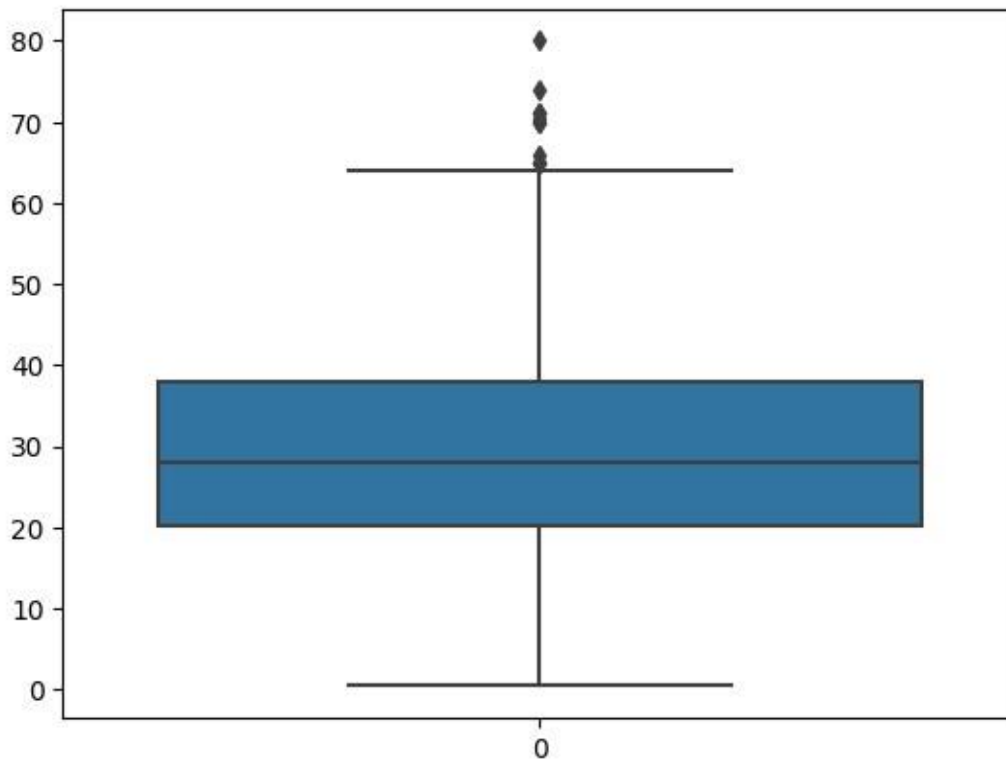
```
[13]: <Axes: xlabel='Survived', ylabel='Pclass'>
```



### OUTLIER DETECTION

```
[14]: sns.boxplot(data.Age)
```

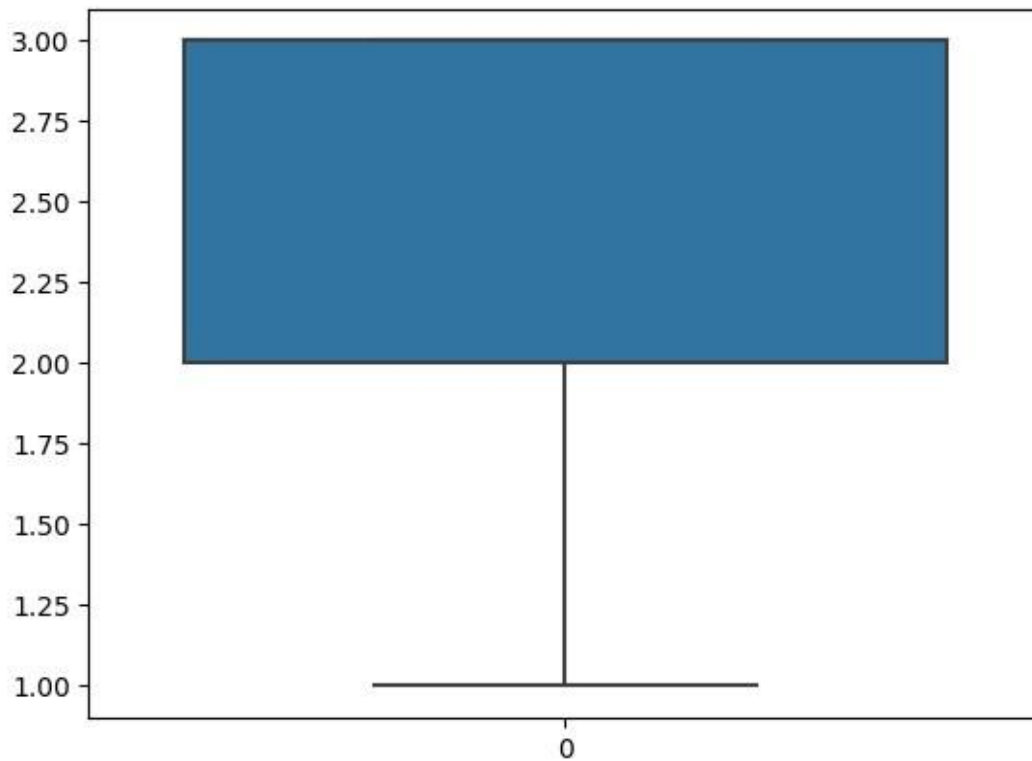
```
[14]: <Axes: >
```



```
[15]: sns.boxplot(data.Pclass)
```

```
[15]: <Axes: >
```





## SPLITTING DEPENDENT AND INDEPENDENT VARIABLES

```
[16]: data.head()
```

```
[16]: 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 \0 Braund, Mr. Owen Harris male 22.0          1
1 Cumings, Mrs. John Bradley (Florence Briggs Th... female 38.0      1
3 Futrelle, Mrs. Jacques Heath (Lily May Peel) female 35.0      1
```

```
2          Heikkinen, Miss. Laina female 26.0          0
4          Allen, Mr. William Henry male 35.0          0
```

```
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

```

```
[18]:
```

```
[17]: x=data.drop(columns=["Survived","PassengerId","Name","Ticket","Cabin"])
```

```
x
```

```
[18]:      Pclass    Sex  Age SibSp Parch    Fare Embarked
```

```

0      3  male  22.0  1    0    7.2500    S
1      1  female 38.01    0 71.2833    C
2      3  female 26.00    0    7.9250    S
3      1  female 35.01    0 53.1000    S
4      3  male  35.0  0    0    8.0500    S
..      ...    ...    ...    ...    ...    ...
886     2  male  27.0  0    0 13.0000    S
887     1  female 19.00    0 30.0000    S
888     3  female    NaN  1    2 23.4500    S
889     1  male  26.0  0    0 30.0000    C
890     3  male  32.0  0    0    7.7500    Q

```

```
[891 rows x 7 columns]
```

```
[19]: x.shape
```

```
[19]: (891, 7)
```

```
[20]: type(x)
```

```
[20]: pandas.core.frame.DataFrame
```

```
[21]: y=data["Survived"]
```

```
[22]: y.head
```

```
[22]: <bound method NDFrame.head of 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>
```

```
[23]: type(y)
```

```
[23]: pandas.core.series.Series
```

## ENCODING

```
[24]: x.head()
```

```
[24]:   Pclass    Sex  Age  SibSp  Parch    Fare Embarked
0        3   male  22.0    1      0  7.2500      S
1        1  female  38.0    1      0 71.2833      C
2        3  female  26.0    0      0  7.9250      S
3        1  female  35.0    1      0 53.1000      S
4        3   male  35.0    0      0  8.0500      S
```

```
[25]: from sklearn.preprocessing import LabelEncoder
      le=LabelEncoder()
```

```
[26]: x["Sex"]=le.fit_transform(x["Sex"])
```

```
[27]: x.head()
```

```
[27]:   Pclass  Sex  Age  SibSp  Parch    Fare Embarked
0        3    1  22.0     1      0  7.2500      S
1        1    0  38.0     1      0 71.2833      C
2        3    0  26.0     0      0  7.9250      S
3        1    0  35.0     1      0 53.1000      S
4        3    1  35.0     0      0  8.0500      S
```

```
[28]: print(le.classes_)
```

```
['female' 'male']
```

```
[29]: mapping=dict(zip(le.classes_,range(len(le.classes_))))
      mapping
```

```
[29]: {'female': 0, 'male': 1}
```

```
[30]: x["Embarked"]=le.fit_transform(x["Embarked"])
```

```
[31]: x.head()
```

```
[31]: Pclass Sex   Age SibSp Parch   Fare Embarked
      0     3    1 22.0     1     0  7.2500     2
      1     1    0 38.0     1     0 71.2833     0
      2     3    0 26.0     0     0  7.9250     2
      3     1    0 35.0     1     0 53.1000     2
      4     3    1 35.0     0     0  8.0500     2
```

```
[32]: print(le.classes_)
```

```
['C' 'Q' 'S' nan]
```

```
[33]: mapping=dict(zip(le.classes_,range(len(le.classes_))))
      mapping
```

```
[33]: {'C': 0, 'Q': 1, 'S': 2, nan: 3}
```

```
[34]: x.head()
```

```
[34]: Pclass Sex   Age SibSp Parch   Fare Embarked
      0     3    1 22.0     1     0  7.2500     2
      1     1    0 38.0     1     0 71.2833     0
      2     3    0 26.0     0     0  7.9250     2
      3     1    0 35.0     1     0 53.1000     2
      4     3    1 35.0     0     0  8.0500     2
```

## Feature Scaling

```
[35]: from sklearn.preprocessing import MinMaxScaler
      ms=MinMaxScaler()
```

```
[36]: x_Scaled=pd.DataFrame(ms.fit_transform(x),columns=x.columns)
```

```
[37]: x_Scaled.head()
```

```
[37]: Pclass Sex   Age SibSp Parch   Fare Embarked
      0  1.0  1.0  0.271174  0.1250.0  0.014151  0.666667
      1  0.0  0.0  0.472229  0.1250.0  0.139136  0.000000
      2  1.0  0.0  0.321438  0.0000.0  0.015469  0.666667
      3  0.0  0.0  0.434531  0.1250.0  0.103644  0.666667
      4  1.0  1.0  0.434531  0.0000.0  0.015713  0.666667
```

## SPLITTING DATA INTO TRAINING AND TESTING

```
[38]: from sklearn.model_selection import train_test_split
      X_train,X_test,y_train,y_test =
      train_test_split(x_Scaled,y,test_size =0.2,random_state =0)
```

```
[39]: print(X_train.shape,X_test.shape,y_train.shape,y_test.shape)

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