ASSIGNMENT-3

M VENKATA SAI

In [2]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [4]:

```
data=pd.read_csv("Titanic-Dataset.csv")
```

In [5]:

```
data.head()
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabir
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN

In [6]:

```
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
```

In [7]:
```
data.describe()
```

Out[7]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	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	14.526497	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	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

HANDALING NULL VALUES

In [8]:
```
data.isnull().any()
```

Out[8]:
```
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
```

In [9]:
```
data.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
```

FILLING NULL VALUES IN AGE COLUMN WITH MEAN

In [11]:
```
mean=data["Age"].mean()
```

In [12]:
```
data["Age"]=data["Age"].fillna(mean)
```

In [13]:
```
data["Age"].tail()
```

Out[13]:
```
886    27.000000
887    19.000000
888    29.699118
889    26.000000
890    32.000000
Name: Age, dtype: float64
```

In [14]:
```
data["Age"].isnull().sum()
```

Out[14]:
```
0
```

FILLING NULL VALUES IN EMBARKED COLUMN WITH MODE

In [15]:
```
em_mode=data["Embarked"].mode()
```

In [16]:
```
data["Embarked"]=data["Embarked"].fillna(em_mode[0])
```

In [17]:
```
data["Embarked"].isnull().sum()
```

Out[17]:
```
0
```

FILLING NULL VALUES IN CABIN COLUMN WITH MODE

In [18]:
```
c_mode=data["Cabin"].mode()
```

In [19]:
```
data["Cabin"]
```

Out[19]:

```
0        NaN
1        C85
2        NaN
3       C123
4        NaN
        ...
886      NaN
887      B42
888      NaN
889     C148
890      NaN
Name: Cabin, Length: 891, dtype: object
```

In [20]:

```
c_mode
```

Out[20]:

```
0        B96 B98
1    C23 C25 C27
2             G6
Name: Cabin, dtype: object
```

In [21]:

```
data["Cabin"]=data["Cabin"].fillna(c_mode[2])
```

In [22]:

```
data["Cabin"].isnull().sum()
```

Out[22]:

```
0
```

In [23]:

```
data["Cabin"]
```

Out[23]:

```
0         G6
1        C85
2         G6
3       C123
4         G6
        ...
886       G6
887      B42
888       G6
889     C148
890       G6
Name: Cabin, Length: 891, dtype: object
```

In [24]:

```
data.isnull().sum()
```

Out[24]:

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

DATA VISUALISATION

In [29]:

```
cor=data.corr()
```

```
C:\Users\venka\AppData\Local\Temp\ipykernel_9632\1426905697.py:1: FutureWarning: T
he default value of numeric_only in DataFrame.corr is deprecated. In a future vers
ion, it will default to False. Select only valid columns or specify the value of n
umeric_only to silence this warning.
  cor=data.corr()
```

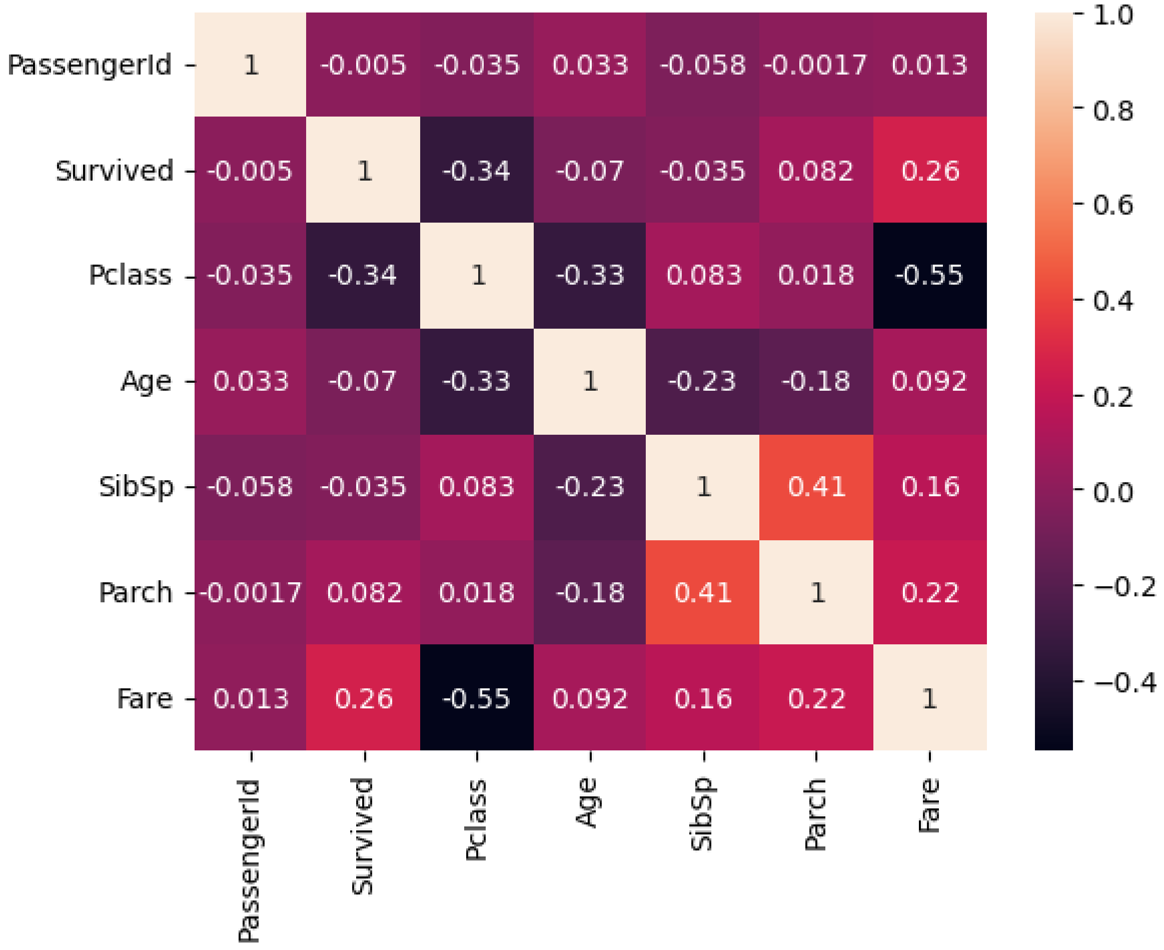
In [30]:
```
cor
```

Out[30]:

	PassengerId	**Survived**	**Pclass**	**Age**	**SibSp**	**Parch**	**Fare**
PassengerId	1.000000	-0.005007	-0.035144	0.033207	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.069809	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.331339	0.083081	0.018443	-0.549500
Age	0.033207	-0.069809	-0.331339	1.000000	-0.232625	-0.179191	0.091566
SibSp	-0.057527	-0.035322	0.083081	-0.232625	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.179191	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.091566	0.159651	0.216225	1.000000

In [31]:
```
sns.heatmap(cor,annot=True)
```

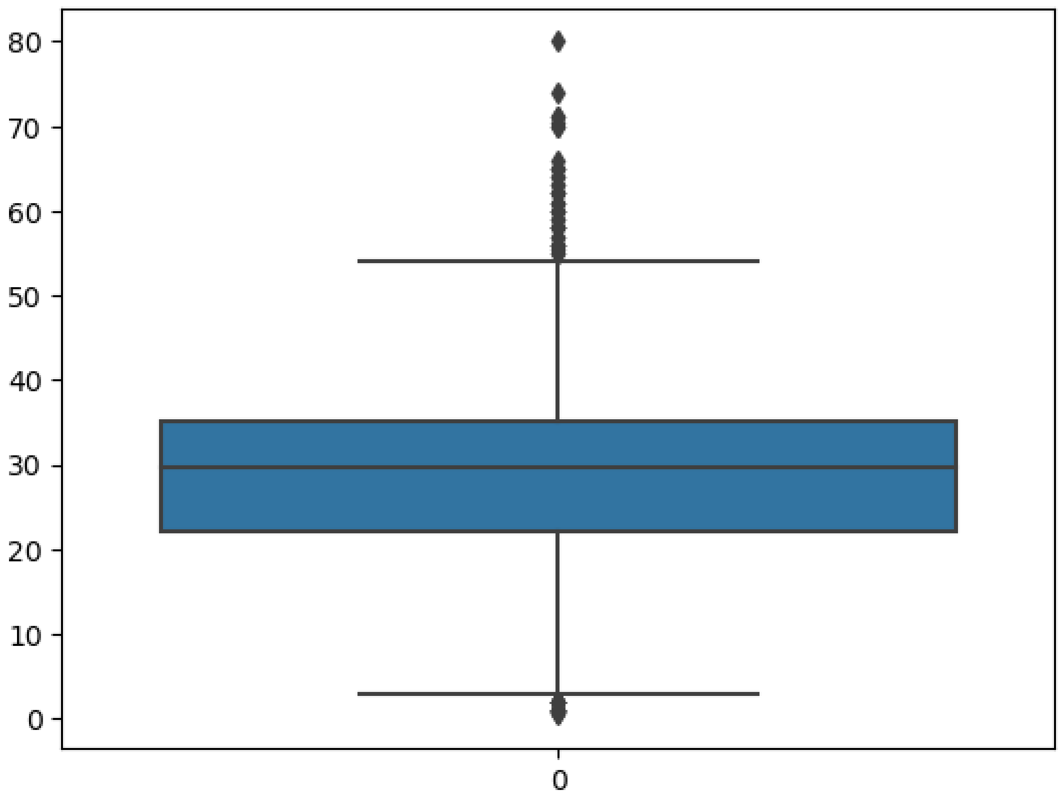
Out[31]:
```
<Axes: >
```



HANDILING WITH OUTLIERS

In [32]:
```
sns.boxplot(data["Age"])
```

```
Out[32]: <Axes: >
```



```
In [33]: Age_q1=data.Age.quantile(0.25)
         Age_q3=data.Age.quantile(0.75)
         print(Age_q1)
         print(Age_q3)

         22.0
         35.0
```

```
In [34]: IQR_Age=Age_q3-Age_q1
         IQR_Age

Out[34]: 13.0
```

```
In [35]: ul_Age=Age_q3+1.5*IQR_Age
         ul_Age

Out[35]: 54.5
```

```
In [37]: ll_Age=Age_q1-1.5*IQR_Age
         ll_Age

Out[37]: 2.5
```

```
In [38]: median_Age=data["Age"].median()
         median_Age

Out[38]: 29.69911764705882
```

```
In [39]: data["Age"]=np.where(data["Age"]>ul_Age,median_Age,data["Age"])
```

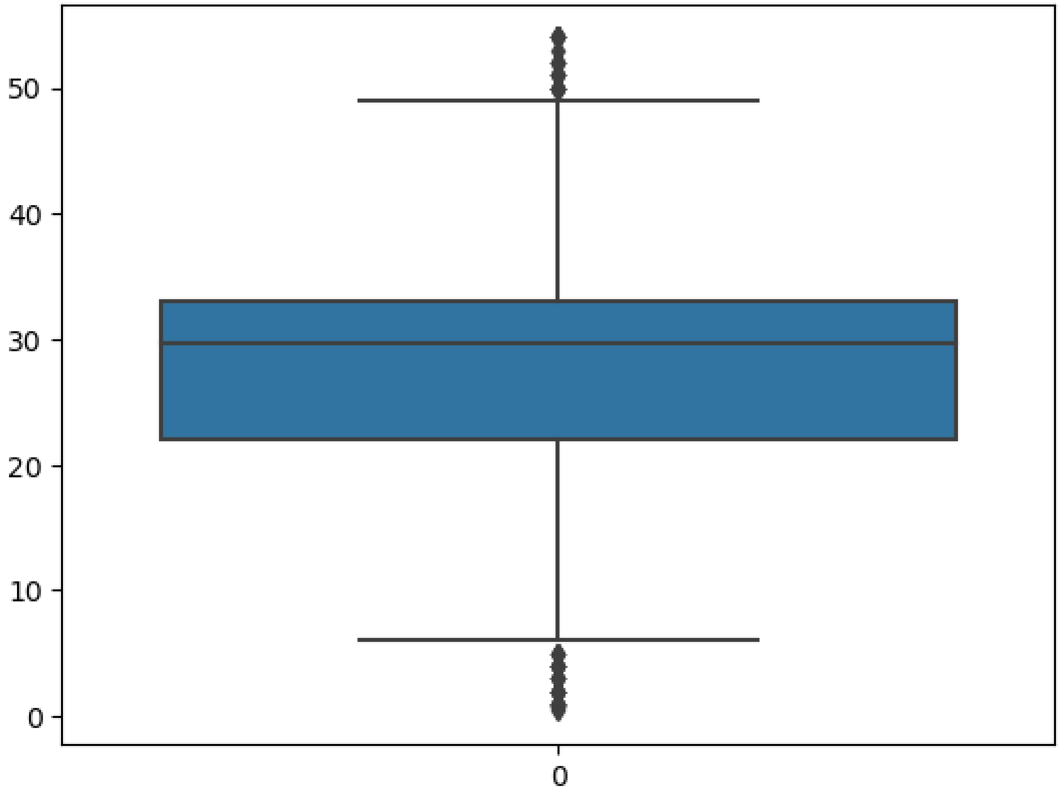
```
In [40]: (data["Age"]>54.5).sum()

Out[40]: 0
```

In [41]:

```
sns.boxplot(data["Age"])
```

Out[41]:

```
<Axes: >
```

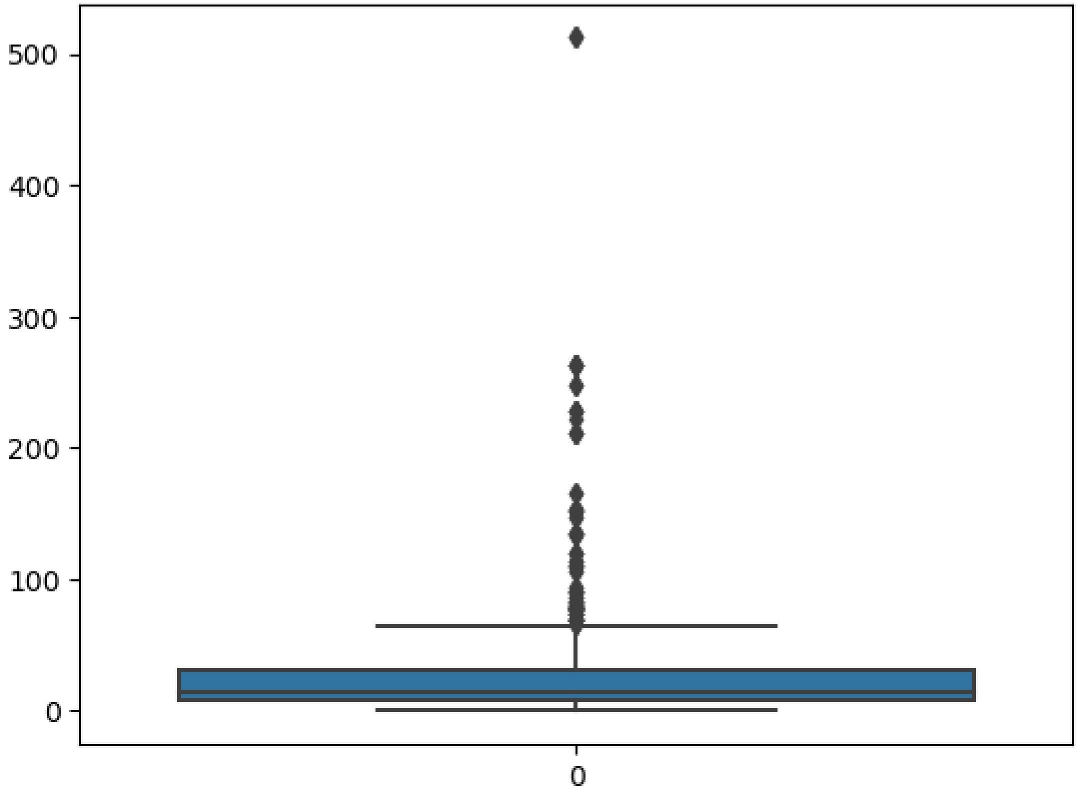


In [42]:

```
sns.boxplot(data["Fare"])
```

Out[42]:

```
<Axes: >
```



In [43]:

```
fare_q1=data.Fare.quantile(0.25)
fare_q3=data.Fare.quantile(0.75)
```

```
print(fare_q1)
print(fare_q3)
```

```
7.9104
31.0
```

In [46]:
```
IQR_Fare=fare_q3-fare_q1
IQR_Fare
```

Out[46]:
```
23.0896
```

In [48]:
```
upperlimit_Fare=fare_q3+1.5*IQR_Fare
upperlimit_Fare
```

Out[48]:
```
65.6344
```

In [49]:
```
lower_limit_Fare=fare_q1-1.5*IQR_Fare
lower_limit_Fare
```

Out[49]:
```
-26.724
```

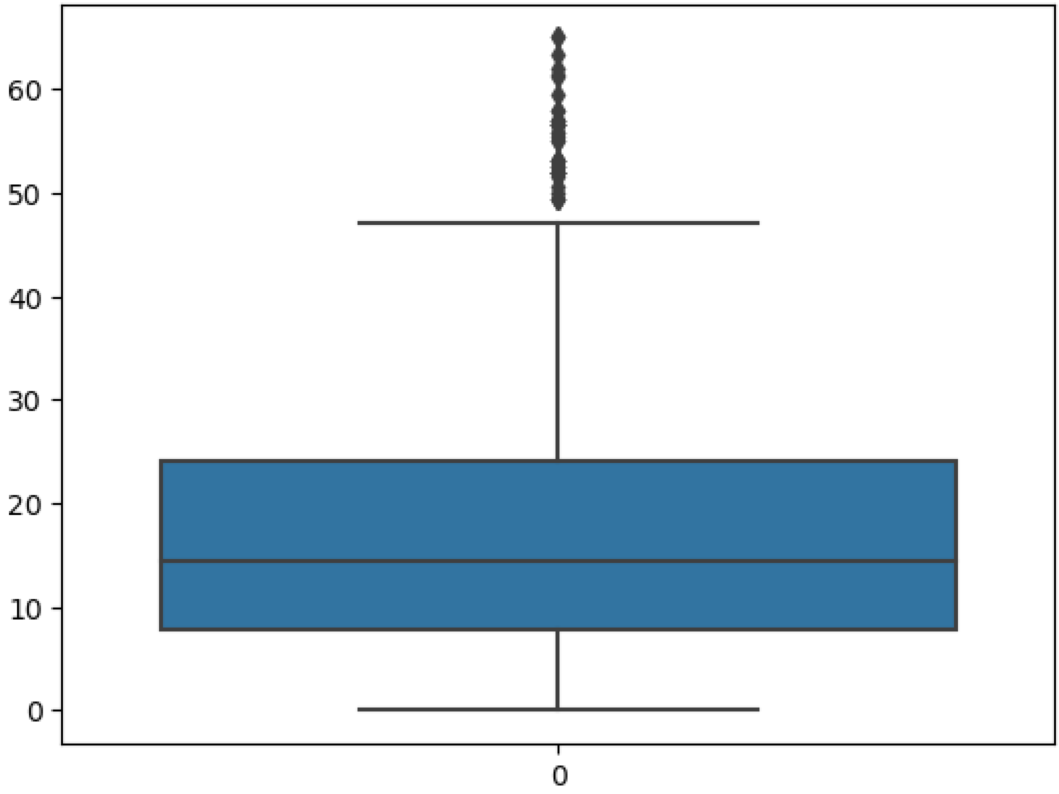
In [50]:
```
median_Fare=data["Fare"].median()
median_Fare
```

Out[50]:
```
14.4542
```

In [51]:
```
data['Fare']=np.where((data['Fare']>upperlimit_Fare),median_Fare,data['Fare'])
```

In [52]:
```
sns.boxplot(data["Fare"])
```

Out[52]:
```
<Axes: >
```



In [53]:
```
(data["Fare"]>65).sum()
```

Out[53]:
```
0
```

dropping the variables

```
In [54]: data.drop(['Name'],axis=1,inplace=True)
```

```
In [55]: data
```

Out[55]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	E
0	1	0	3	male	22.000000	1	0	A/5 21171	7.2500	G6	
1	2	1	1	female	38.000000	1	0	PC 17599	14.4542	C85	
2	3	1	3	female	26.000000	0	0	STON/O2. 3101282	7.9250	G6	
3	4	1	1	female	35.000000	1	0	113803	53.1000	C123	
4	5	0	3	male	35.000000	0	0	373450	8.0500	G6	
...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	male	27.000000	0	0	211536	13.0000	G6	
887	888	1	1	female	19.000000	0	0	112053	30.0000	B42	
888	889	0	3	female	29.699118	1	2	W./C. 6607	23.4500	G6	
889	890	1	1	male	26.000000	0	0	111369	30.0000	C148	
890	891	0	3	male	32.000000	0	0	370376	7.7500	G6	

891 rows × 11 columns

```
In [56]: data.drop(['Ticket'],axis=1,inplace=True)
```

```
In [57]: data
```

Out[57]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	1	0	3	male	22.000000	1	0	7.2500	G6	S
1	2	1	1	female	38.000000	1	0	14.4542	C85	C
2	3	1	3	female	26.000000	0	0	7.9250	G6	S
3	4	1	1	female	35.000000	1	0	53.1000	C123	S
4	5	0	3	male	35.000000	0	0	8.0500	G6	S
...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	male	27.000000	0	0	13.0000	G6	S
887	888	1	1	female	19.000000	0	0	30.0000	B42	S
888	889	0	3	female	29.699118	1	2	23.4500	G6	S
889	890	1	1	male	26.000000	0	0	30.0000	C148	C
890	891	0	3	male	32.000000	0	0	7.7500	G6	Q

891 rows × 10 columns

In [58]:

```
data.drop(["PassengerId"],axis=1,inplace=True)
```

In [59]:

```
data
```

Out[59]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	0	3	male	22.000000	1	0	7.2500	G6	S
1	1	1	female	38.000000	1	0	14.4542	C85	C
2	1	3	female	26.000000	0	0	7.9250	G6	S
3	1	1	female	35.000000	1	0	53.1000	C123	S
4	0	3	male	35.000000	0	0	8.0500	G6	S
...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.000000	0	0	13.0000	G6	S
887	1	1	female	19.000000	0	0	30.0000	B42	S
888	0	3	female	29.699118	1	2	23.4500	G6	S
889	1	1	male	26.000000	0	0	30.0000	C148	C
890	0	3	male	32.000000	0	0	7.7500	G6	Q

891 rows × 9 columns

In [60]:

```
data.drop(["Cabin"],axis=1,inplace=True)
```

In [61]:

```
data
```

Out[61]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.000000	1	0	7.2500	S
1	1	1	female	38.000000	1	0	14.4542	C
2	1	3	female	26.000000	0	0	7.9250	S
3	1	1	female	35.000000	1	0	53.1000	S
4	0	3	male	35.000000	0	0	8.0500	S
...	...	...	...	...	...	...	...	...
886	0	2	male	27.000000	0	0	13.0000	S
887	1	1	female	19.000000	0	0	30.0000	S
888	0	3	female	29.699118	1	2	23.4500	S
889	1	1	male	26.000000	0	0	30.0000	C
890	0	3	male	32.000000	0	0	7.7500	Q

891 rows × 8 columns

Splitting the data

In [62]:

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

In [63]:

```
y.head()
```

Out[63]:

```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

In [64]:

```
data
```

Out[64]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.000000	1	0	7.2500	S
1	1	1	female	38.000000	1	0	14.4542	C
2	1	3	female	26.000000	0	0	7.9250	S
3	1	1	female	35.000000	1	0	53.1000	S
4	0	3	male	35.000000	0	0	8.0500	S
...	...	...	...	...	...	...	...	...
886	0	2	male	27.000000	0	0	13.0000	S
887	1	1	female	19.000000	0	0	30.0000	S
888	0	3	female	29.699118	1	2	23.4500	S
889	1	1	male	26.000000	0	0	30.0000	C
890	0	3	male	32.000000	0	0	7.7500	Q

891 rows × 8 columns

ENCODING

In [65]:

```
from sklearn.preprocessing import LabelEncoder
```

In [66]:

```
le=LabelEncoder()
```

In [67]:

```
data["Sex"]=le.fit_transform(data["Sex"])
```

In [68]:

```
data["Sex"]
```

Out[68]:

```
0      1
1      0
2      0
3      0
4      1
      ..
886    1
887    0
888    0
889    1
890    1
Name: Sex, Length: 891, dtype: int32
```

In [69]:

```
data.head()
```

Out[69]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	22.0	1	0	7.2500	S
1	1	1	0	38.0	1	0	14.4542	C
2	1	3	0	26.0	0	0	7.9250	S
3	1	1	0	35.0	1	0	53.1000	S
4	0	3	1	35.0	0	0	8.0500	S

In [70]:

```
data["Embarked"]=le.fit_transform(data["Embarked"])
```

In [71]:

```
data.head()
```

Out[71]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	22.0	1	0	7.2500	2
1	1	1	0	38.0	1	0	14.4542	0
2	1	3	0	26.0	0	0	7.9250	2
3	1	1	0	35.0	1	0	53.1000	2
4	0	3	1	35.0	0	0	8.0500	2

In [72]:

```
data["Pclass"].nunique()
```

Out[72]:

```
3
```

In [73]:

```
data["Pclass"].unique()
```

Out[73]:

```
array([3, 1, 2], dtype=int64)
```

In [74]:

```
data["Sex"].unique()
```

Out[74]:

```
array([1, 0])
```

In [75]:

```
data["Embarked"].unique()
```

Out[75]:

```
array([2, 0, 1])
```

Spliting the train and test data

In [76]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(data,y,test_size=0.3,random_state=0
```

In [77]:

```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[77]:

```
((623, 8), (268, 8), (623,), (268,))
```

Feature Scaling

In [78]:

```
from sklearn.preprocessing import StandardScaler
```

In [79]:

```
sc=StandardScaler()
```

In [80]:

```
x_train=sc.fit_transform(x_train)
```

In [81]:

```
x_train
```

```
Out[81]: array([[ 1.25474307, -1.5325562 ,  0.72592065, ..., -0.47299765,
                  0.67925137,  0.56710989],
                [ 1.25474307, -1.5325562 , -1.37756104, ..., -0.47299765,
                 -0.26059483, -2.03075381],
                [-0.79697591,  0.84844757,  0.72592065, ...,  1.93253327,
                  2.26045064,  0.56710989],
                ...,
                [-0.79697591,  0.84844757,  0.72592065, ..., -0.47299765,
                 -0.78281017, -0.73182196],
                [ 1.25474307,  0.84844757, -1.37756104, ..., -0.47299765,
                 -0.03170555,  0.56710989],
                [-0.79697591, -0.34205431,  0.72592065, ...,  0.72976781,
                  1.64661898,  0.56710989]])
```

```
In [82]: x_test=sc.fit_transform(x_test)
```

```
In [83]: x_test
```

```
Out[83]: array([[-0.77151675,  0.77963055,  0.76537495, ..., -0.47809977,
                 -0.15813988, -1.76531134],
                [-0.77151675,  0.77963055,  0.76537495, ..., -0.47809977,
                 -0.72165412,  0.63014911],
                [-0.77151675,  0.77963055,  0.76537495, ...,  0.87064484,
                  1.03823178, -0.56758111],
                ...,
                [-0.77151675,  0.77963055,  0.76537495, ..., -0.47809977,
                 -0.15847431, -1.76531134],
                [ 1.29614814,  0.77963055, -1.30654916, ..., -0.47809977,
                 -0.72607524,  0.63014911],
                [-0.77151675, -1.64991582,  0.76537495, ..., -0.47809977,
                  0.92369033, -1.76531134]])
```

```
In [ ]:
```