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

1.Importing Libraries

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

2.Importing Dataset

```
df=pd.read_csv('Titanic-Dataset.csv')
df.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  \
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
```

```
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
```

```
df.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
```

```
    3.Checking Null Values
```

```
df.isnull().any()
```

```
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
```

```
df.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
```

```
print("Null percentage in columns : ")
for i in df.columns:
    c=df[i].count()
    n=df[i].isnull().sum()
    print(i," : ",(n/(n+c)) * 100)
```

```
Null percentage in columns :
PassengerId  :  0.0
Survived  :  0.0
Pclass  :  0.0
Name  :  0.0
Sex  :  0.0
Age  :  19.865319865319865
SibSp  :  0.0
Parch  :  0.0
Ticket  :  0.0
Fare  :  0.0
Cabin  :  77.10437710437711
Embarked  :  0.22446689113355783
```

```
df['Age']=df['Age'].fillna(df['Age'].median())
```

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

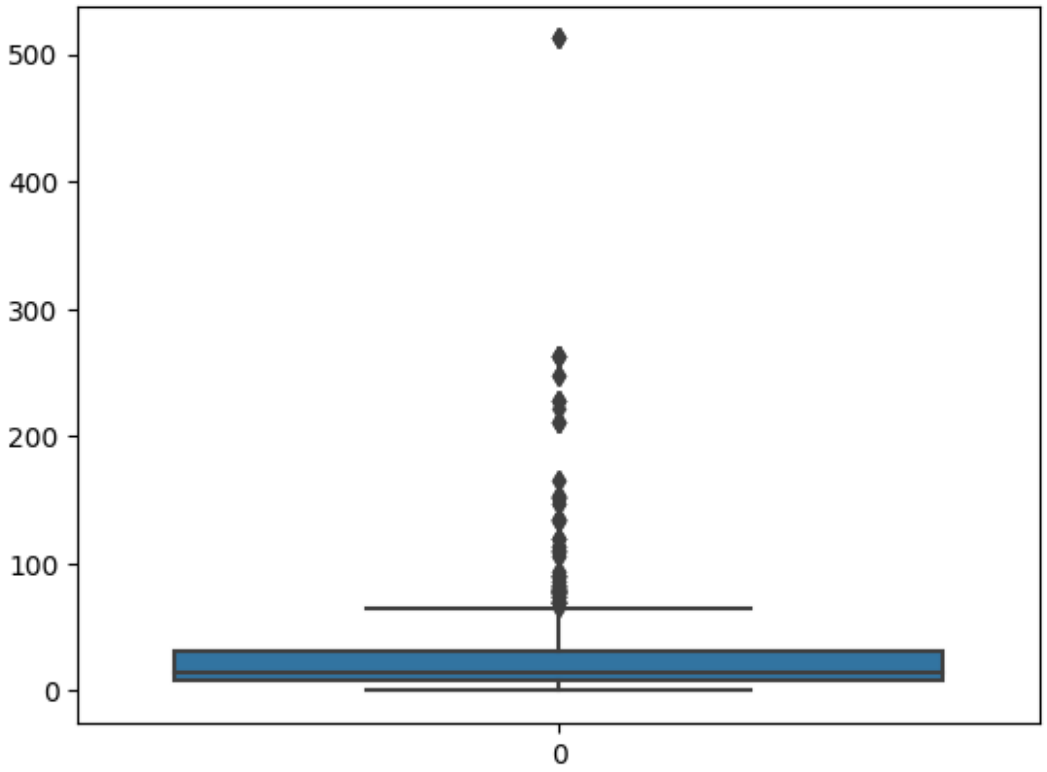
```
df.drop('Cabin',axis=1,inplace=True)
```

```
print("Null percentage in columns : ")
for i in df.columns:
    c=df[i].count()
    n=df[i].isnull().sum()
    print(i," : ",(n/(n+c)) * 100)
```

```
Null percentage in columns :
PassengerId  :  0.0
Survived  :  0.0
Pclass  :  0.0
Name  :  0.0
Sex  :  0.0
Age  :  0.0
SibSp  :  0.0
Parch  :  0.0
Ticket  :  0.0
Fare  :  0.0
Embarked  :  0.0

    4.Data Visualization

    5.Checking for outliers

df.columns

Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age',
'SibSp',
       'Parch', 'Ticket', 'Fare', 'Embarked'],
      dtype='object')

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 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          891 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  Embarked     891 non-null    object
dtypes: float64(2), int64(5), object(4)
memory usage: 76.7+ KB
```

Fare

```
sns.boxplot(df['Fare'])

<Axes: >
```




```
q1=df['Fare'].quantile(0.25)
q2=df['Fare'].quantile(0.5)
q3=df['Fare'].quantile(0.75)
print(q1,q2,q3)
```

```
7.9104 14.4542 31.0
```

```
iqr=q3-q1
upper_lm=q3+1.5*iqr
lower_lm=q1-1.5*iqr
print(upper_lm,lower_lm)
```

```
65.6344 -26.724
```

```
# No.of Outliers
d=df[(df['Fare']>upper_lm)]
d.head()
```

```
    PassengerId  Survived  Pclass  \
1             2         1       1
27           28         0       1
31           32         1       1
34           35         0       1
52           53         1       1

                                                 Name     Sex    Age
```

```
SibSp  \
1   Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
1
27                      Fortune, Mr. Charles Alexander    male  19.0
3
31     Spencer, Mrs. William Augustus (Marie Eugenie)  female  28.0
1
34                               Meyer, Mr. Edgar Joseph    male  28.0
1
52            Harper, Mrs. Henry Sleeper (Myna Haxtun)  female  49.0
1

    Parch    Ticket      Fare Embarked
1       0  PC 17599   71.2833        C
27      2     19950  263.0000        S
31      0  PC 17569  146.5208        C
34      0  PC 17604   82.1708        C
52      0  PC 17572   76.7292        C
```

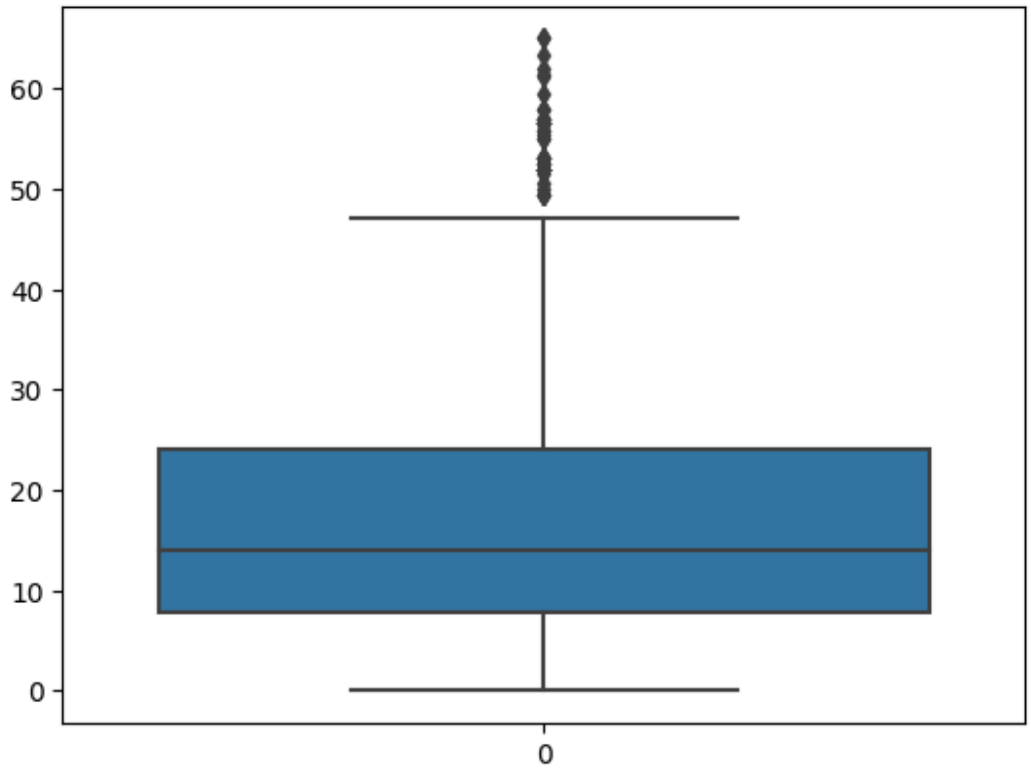
```
d.shape
```

```
(116, 11)
```

```
df['Fare']= np.where(df['Fare']>upper_lm,14,df['Fare'])
```

```
sns.boxplot(df['Fare'])
```

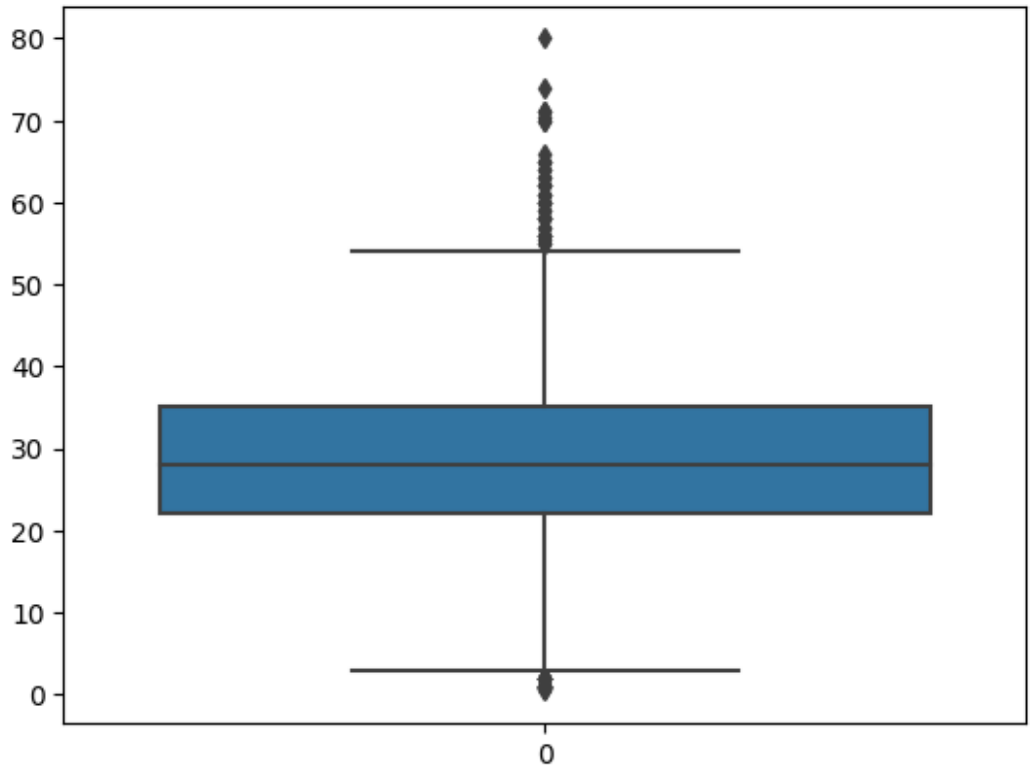
```
<Axes: >
```



Age

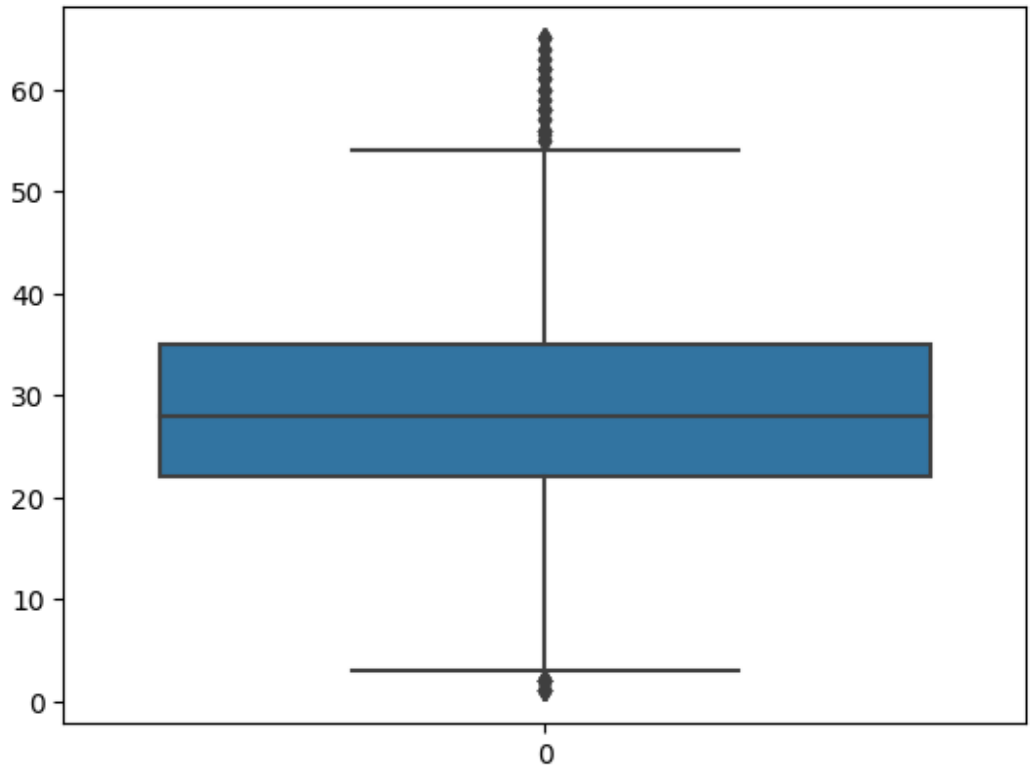
```
sns.boxplot(df['Age'])
```

```
<Axes: >
```




```
from scipy import stats
z=stats.zscore(df['Age'])
p=df["Age"].quantile(0.99)
p1=df['Age'].quantile(0.01)

df=df[df['Age']<=p]
df=df[df['Age']>=p1]

sns.boxplot(df['Age'])
```

```
<Axes: >
```




```
df.shape
```

```
(876, 11)
```

```
    6.Splitting Dependent and Independent variables
```

```
df.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  \
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 Embarked
0      0         A/5 21171   7.250        S
1      0          PC 17599  14.000        C
2      0  STON/O2. 3101282   7.925        S
3      0            113803  53.100        S
4      0            373450   8.050        S
```

```
X=df.drop(columns=['PassengerId','Name','Ticket','Survived'])
X
```

```
     Pclass     Sex   Age  SibSp  Parch    Fare Embarked
0         3    male  22.0      1      0   7.250        S
1         1  female  38.0      1      0  14.000        C
2         3  female  26.0      0      0   7.925        S
3         1  female  35.0      1      0  53.100        S
4         3    male  35.0      0      0   8.050        S
..      ...     ...   ...    ...    ...     ...      ...
886       2    male  27.0      0      0  13.000        S
887       1  female  19.0      0      0  30.000        S
888       3  female  28.0      1      2  23.450        S
889       1    male  26.0      0      0  30.000        C
890       3    male  32.0      0      0   7.750        Q
```

```
[876 rows x 7 columns]
```

```
y=df.iloc[:,1:2]
y
```

```
     Survived
0           0
1           1
2           1
3           1
4           0
..        ...
886         0
887         1
888         0
889         1
890         0
```

```
[876 rows x 1 columns]
```

```
    7.Performing Encoding
```

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
X['Sex'] =le.fit_transform(X['Sex'])
```

```
d1=dict(zip(le.classes_,range(len(le.classes_))))

X['Embarked']=le.fit_transform(X['Embarked'])

d2=dict(zip(le.classes_,range(len(le.classes_))))
d2
```

```
{'C': 0, 'Q': 1, 'S': 2}
```

```
X.head()
```

```
   Pclass  Sex   Age  SibSp  Parch     Fare  Embarked
0       3    1  22.0      1      0   7.250         2
1       1    0  38.0      1      0  14.000         0
2       3    0  26.0      0      0   7.925         2
3       1    0  35.0      1      0  53.100         2
4       3    1  35.0      0      0   8.050         2
```

```
d1
```

```
{'female': 0, 'male': 1}
```

```
d2
```

```
{'C': 0, 'Q': 1, 'S': 2}
```

```
    7.Feature Scaling
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc.fit_transform(X)
```

```
array([[ 0.82171099,  0.74376844, -0.58974449, ..., -0.46713727,
        -0.78835932,  0.58259401],
       [-1.57369276, -1.34450448,  0.71990595, ..., -0.46713727,
        -0.25846421, -1.95647243],
       [ 0.82171099, -1.34450448, -0.26233188, ..., -0.46713727,
        -0.73536981,  0.58259401],
       ...,
       [ 0.82171099, -1.34450448, -0.09862557, ...,  2.01293697,
         0.48338894,  0.58259401],
       [-1.57369276,  0.74376844, -0.26233188, ..., -0.46713727,
         0.99758345, -1.95647243],
       [ 0.82171099,  0.74376844,  0.22878704, ..., -0.46713727,
        -0.74910783, -0.68693921]])
```

```
X.shape
```

```
(876, 7)
```

```
    8.Splitting data in Training and Test
```

```
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,rando
m_state=0)

X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

```
((700, 7), (176, 7), (700, 1), (176, 1))
```

```
from sklearn.linear_model import LogisticRegression
reg=LogisticRegression()
reg.fit(X_train,y_train)
```

```
LogisticRegression()
```

```
y_pred=reg.predict(X_test)

y_pred.shape
```

```
(176,)
```

```
from sklearn.metrics import
accuracy_score,r2_score,classification_report,confusion_matrix
accuracy_score(y_test,y_pred)
```

```
0.8295454545454546
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.84      0.86       107
           1       0.77      0.81      0.79        69

    accuracy                           0.83       176
   macro avg       0.82      0.83      0.82       176
weighted avg       0.83      0.83      0.83       176


confusion_matrix(y_test,y_pred)
```

```
array([[90, 17],
       [13, 56]], dtype=int64)
```