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

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

```
td.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
```

```
td.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.(
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.2
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.6
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.(
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.9
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.4
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.(
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.3

Saving...

```
td.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs	female	38.0	1	0	PC 17599

```
# Check for null values
td.isnull().any() # As we can see the output below tells us that the dataset has null values
```

```
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
```

```
td.isnull().sum() # This returns the number of null values in each column
```

```
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
```

```
# Dropping all the irrevelant columns -- Cabin is alos dropped as 77% of the data has null values
td.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis = 1, inplace = True)
```

```
td["Age"] = td["Age"].fillna(td["Age"].mode()[0]) # Impute the mean value of age in place of null values
```

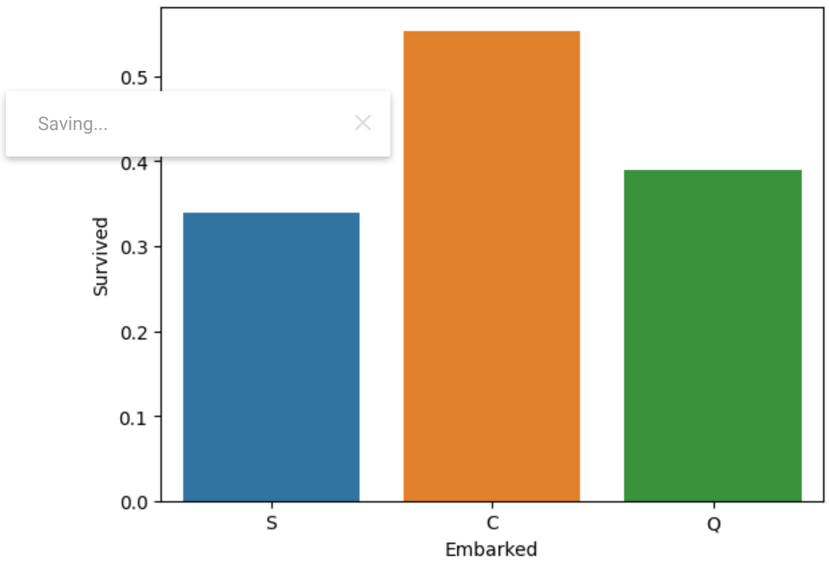
```
td["Embarked"] = td["Embarked"].fillna(td["Embarked"].mode()[0]) # As this is also categorical value -- impute it with mode of the value
```

```
td.isnull().any() # To check any null values after imputation
```

```
Survived    False
Pclass      False
Sex         False
Age         False
SibSp       False
Parch       False
Fare        False
Embarked    False
dtype: bool
```

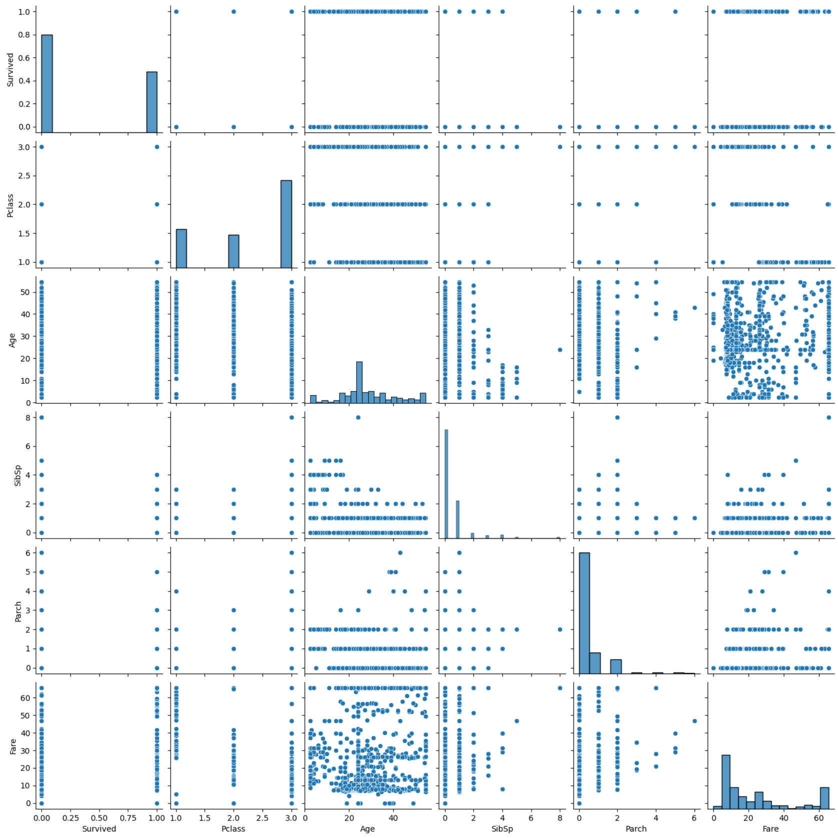
```
# Data Visualization --
sns.barplot(x = "Embarked", y = "Survived", data = td, ci = None)
```

```
<ipython-input-79-37addb60ae57>:2: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x = "Embarked", y = "Survived", data = td, ci = None)
<Axes: xlabel='Embarked', ylabel='Survived'>
```




```
sns.pairplot(td)
```

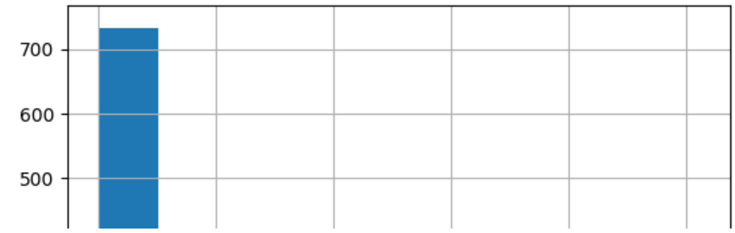
```
<seaborn.axisgrid.PairGrid at 0x7e315b0f7f40>
```



Saving...

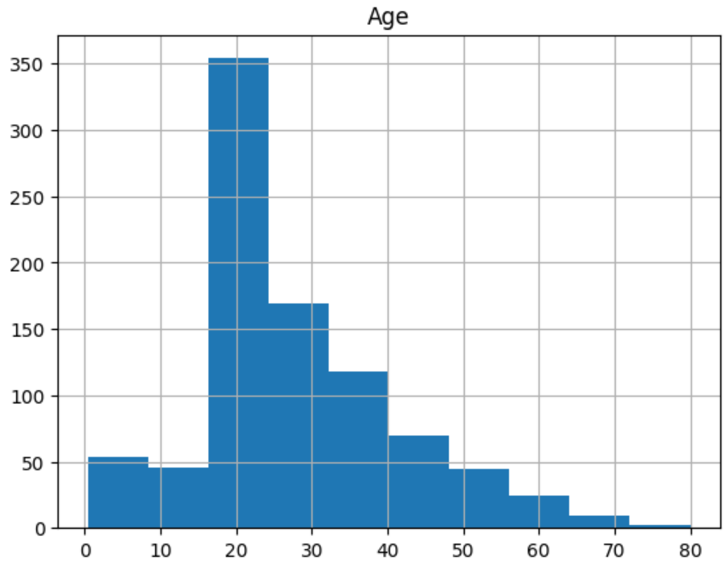
```
td['Fare'].hist()
```

```
<Axes: >
```



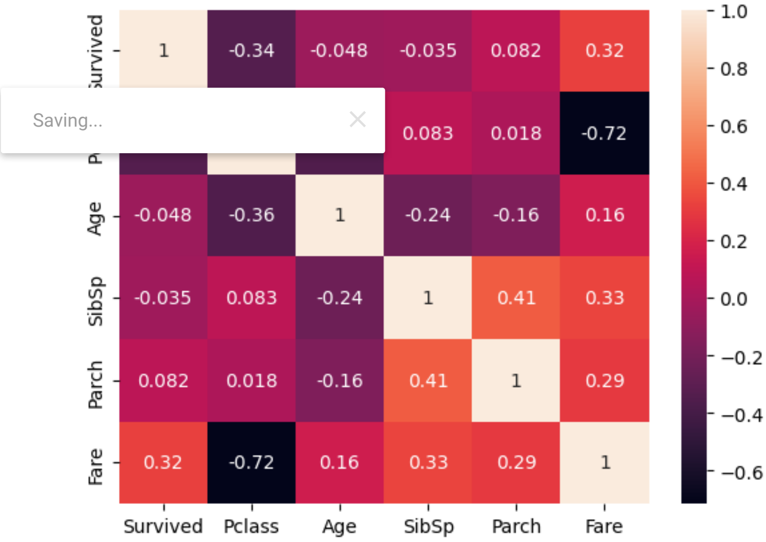

```
td[['Age']].hist()
```

```
array([[<Axes: title={'center': 'Age'}>]], dtype=object)
```




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

```
<ipython-input-95-fa2f6bb73c7d>:1: FutureWarning: The default value of numeric_onl
  sns.heatmap(td.corr(), annot = True)
<Axes: >
```

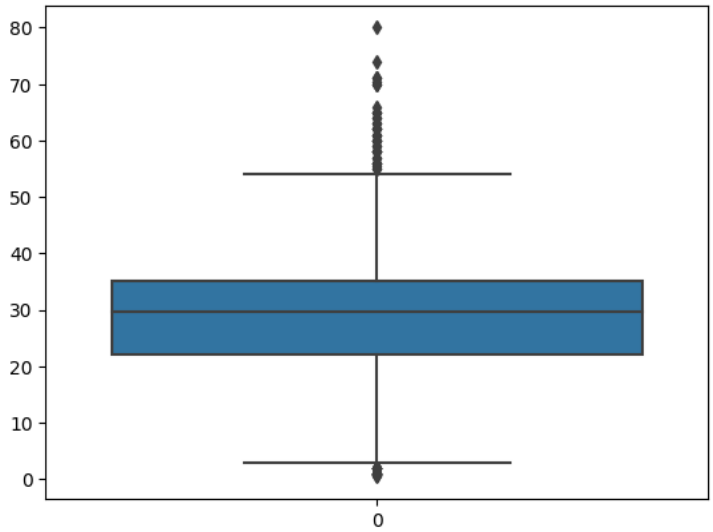



```
sns.boxplot(td.Fare)
```

```
<Axes: >
```


```
sns.boxplot(td.Age)
```

```
<Axes: >
```



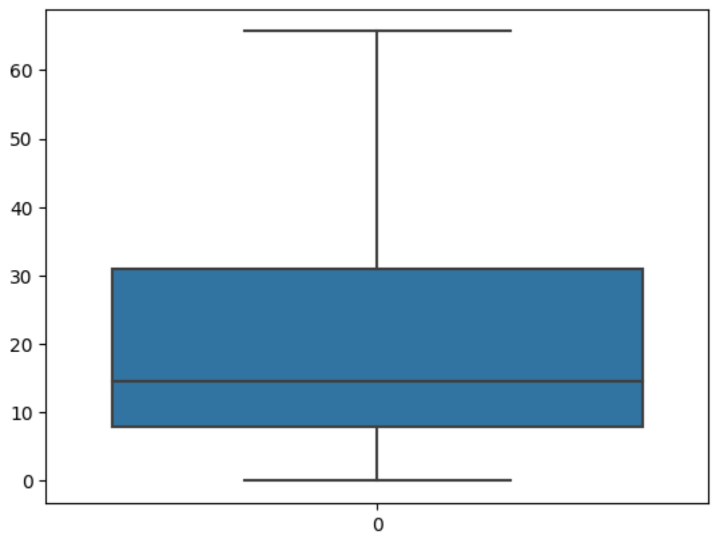

```
# Outlier detection
# As we see from the above two boxplots, fare and age columns have outliers that needs to be deleted so as to increase accuracy of predi
```

```
# FOR OUTLIERS IN AGE --- Using IQR method as it given better performance
Q1 = td['Fare'].quantile(0.25)
Q3 = td['Fare'].quantile(0.75)
IQR = Q3 - Q1
d = 1.5
lower_lim = Q1 - (d*IQR)
r_lim,upper_lim,np.where(td['Fare']<lower_lim,lower_lim,td['Fare']))
```



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

```
<Axes: >
```

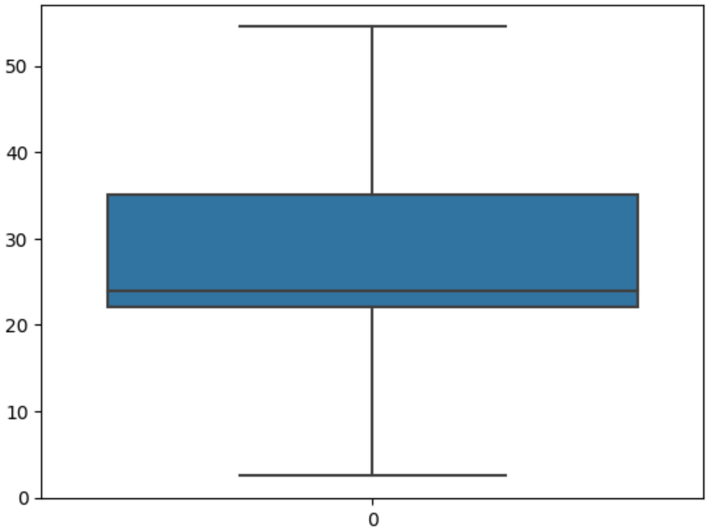



```
# Using the same technique for removing outliers in Age column
q1 = td['Age'].quantile(0.25)
q3 = td['Age'].quantile(0.75)
iqr = q3 - q1
```

```
D = 1.5
lower_lim1 = q1 -(D*iqr)
upper_lim1 = q3 + (D*iqr)
td['Age']=np.where(td['Age']>upper_lim1,upper_lim1,np.where(td['Age']<lower_lim1,lower_lim1,td['Age']))
```

```
sns.boxplot(td.Age)
```

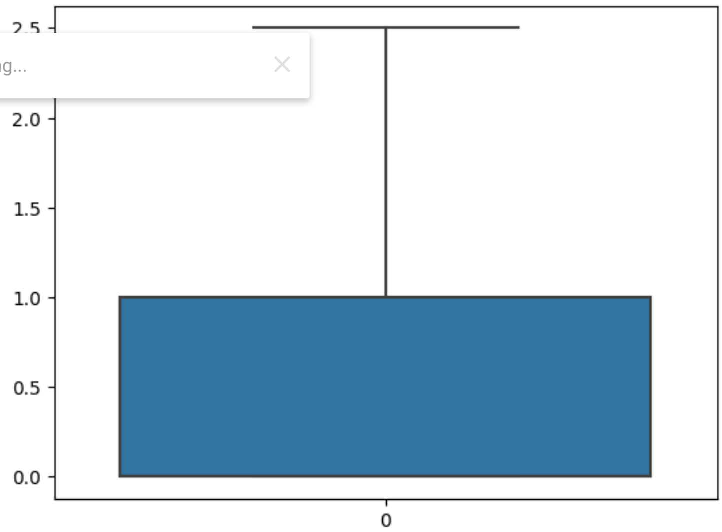
<Axes: >



```
# We also remove the outliers in SibSp and Parch columns
# For SibSp
Q1 = td['SibSp'].quantile(0.25)
Q3 = td['SibSp'].quantile(0.75)
IQR = Q3 - Q1
d = 1.5
lower_lim = Q1 - (d*IQR)
upper_lim = Q3 + (d*IQR)
td['SibSp']=np.where(td['SibSp']>upper_lim,upper_lim,np.where(td['SibSp']<lower_lim,lower_lim,td['SibSp']))
```

```
sns.boxplot(td.SibSp)
```

<Axes: >

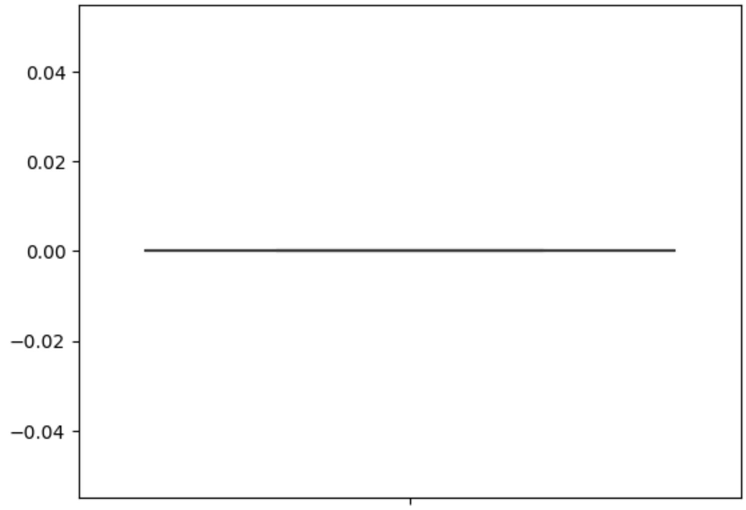


Saving...

```
# For Parch
Q1 = td['Parch'].quantile(0.25)
Q3 = td['Parch'].quantile(0.75)
IQR = Q3 - Q1
d = 1.5
lower_lim = Q1 - (d*IQR)
upper_lim = Q3 + (d*IQR)
td['Parch']=np.where(td['Parch']>upper_lim,upper_lim,np.where(td['Parch']<lower_lim,lower_lim,td['Parch']))
```

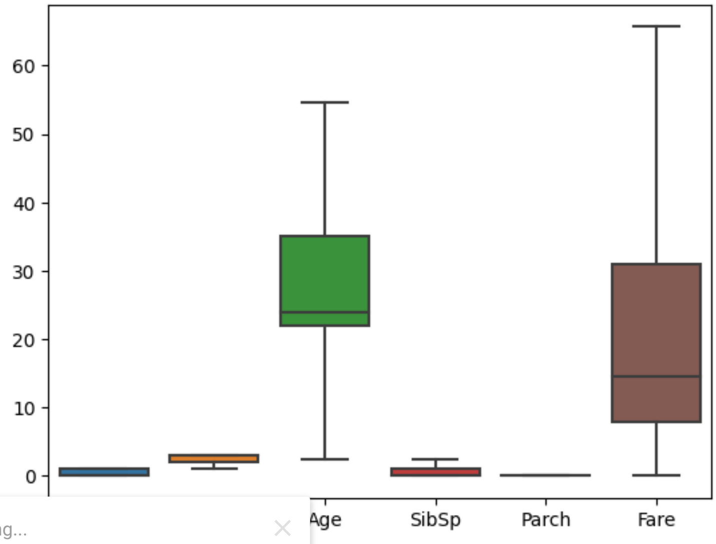
```
sns.boxplot(td.Parch)
```

```
<Axes: >
```




```
# Finally we again check if any columns have outliers
sns.boxplot(td)
# As we dont have any outliers we move to the next step
```

```
<Axes: >
```



Saving...

```
# Splitting the data into dependent and independent variables
x = td.drop(columns = ["Survived"], axis = 1) # Independent variables in the form of 2-D array
y = td["Survived"] # Survived column is the only dependent varaible here
```

```
x.head()
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	male	22.0	1.0	0.0	7.2500	S
1	1	female	38.0	1.0	0.0	65.6344	C
2	3	female	26.0	0.0	0.0	7.9250	S
3	1	female	35.0	1.0	0.0	53.1000	S
4	3	male	35.0	0.0	0.0	8.0500	S

```
y.head()
```

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

```
# Encoding the categorical columns
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
x["Sex"] = le.fit_transform(x["Sex"])
x["Embarked"] = le.fit_transform(x["Embarked"])
x.head()
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	1	22.0	1.0	0.0	7.2500	2
1	1	0	38.0	1.0	0.0	65.6344	0
2	3	0	26.0	0.0	0.0	7.9250	2
3	1	0	35.0	1.0	0.0	53.1000	2
4	3	1	35.0	0.0	0.0	8.0500	2

```
# Feature Scaling -- Bringing all the independent variables in a single scalable format in order to process them
from sklearn.preprocessing import MinMaxScaler
ms = MinMaxScaler()
```

```
x_scaled = pd.DataFrame(ms.fit_transform(x), columns = x.columns)
```

```
# Splitting the data in train test set
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)
```

```
x_train.head()
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
140	1.0	0.0	0.413462	0.0	0.0	0.232284	0.0
439	0.5	1.0	0.548077	0.0	0.0	0.159977	1.0
817	0.5	1.0	0.548077	0.4	0.0	0.563793	0.0
378	1.0	1.0	0.336538	0.0	0.0	0.061134	0.0
491	1.0	1.0	0.355769	0.0	0.0	0.110460	1.0

```
x_test.head()
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
495	1.0	1.0	0.413462	0.0	0.0	0.220285	0.0
[illegible]	[illegible]	[illegible]	[illegible]	0.0	0.0	0.115031	1.0
[illegible]	[illegible]	[illegible]	[illegible]	1.0	0.0	0.443746	0.5
31	0.0	0.0	0.413462	0.4	0.0	1.000000	0.0
255	1.0	0.0	0.509615	0.0	0.0	0.232284	0.0

Saving...

```
y_train.head()
```

```
140    0
439    0
817    0
378    0
491    0
Name: Survived, dtype: int64
```

```
y_test.head()
```

```
495    0
648    0
278    0
31     1
255    1
Name: Survived, dtype: int64
```

```
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
```

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