

▼ Assignment 15 sep

+ Code

+ Text

Perform Data preprocessing on Titanic dataset 1.Data Collection. Please download the dataset from <https://www.kaggle.com/datasets/yasserh/titanic-dataset>

2.Data Preprocessing o Import the Libraries. o Importing the dataset. o Checking for Null Values. o Data Visualization. o Outlier Detection o Splitting Dependent and Independent variables o Perform Encoding o Feature Scaling. o Splitting Data into Train and Test

▼ 1.Data Collection :

Data Set is collected from the kaggle website

▼ 2.Data Preprocessing :

▼ Importing the Libraries

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

▼ Importing the DataSet

```
df=pd.read_csv(r"C:\Users\nitin\Desktop\assignkeents - submissions\Datasets\Titanic-Datase
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0
3	4	1	1	Futrelle, Mrs. Jacques	female	35.0	1	0

```
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	Parch	F
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329

▼ Checking for 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.shape
```

```
(891, 12)
```

```
df["Age"].fillna(df["Age"].median(),inplace=True)
df["Embarked"].fillna(df["Embarked"].mode()[0],inplace=True)
```

```
print(df["Age"].isnull().any())
print(df["Embarked"].isnull().any())
```

```
False
False
```

```
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence	female	38.0	1	0	PC 17599	71.2

```
print(df.shape)
```

```
(891, 12)
```

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

```

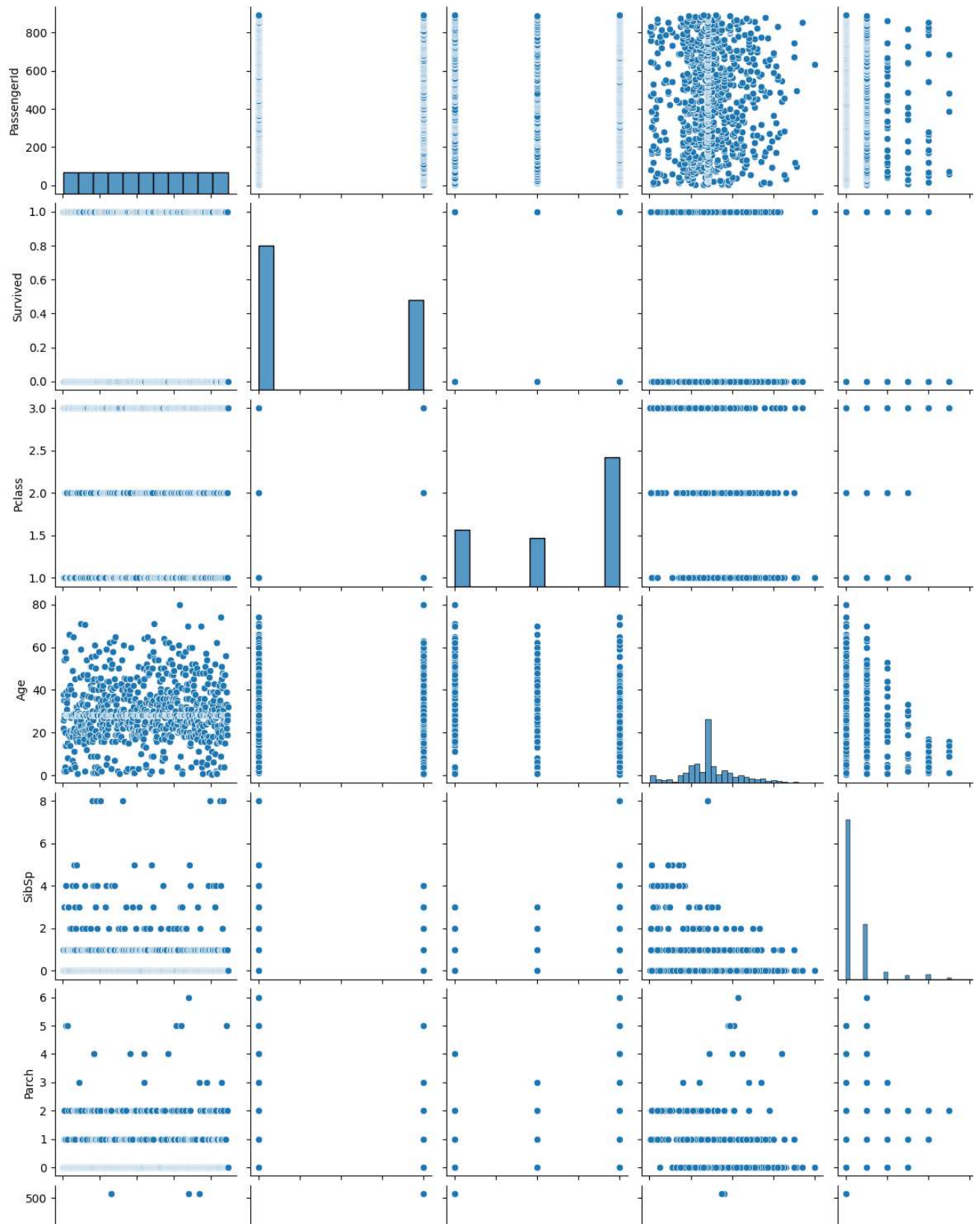
PassengerId    False
Survived        False
Pclass          False
Name            False
Sex             False
Age             False
SibSp           False
Parch           False
Ticket          False
Fare            False
Cabin           True
Embarked        False
dtype: bool

```

▼ Data Visualization

```
sns.pairplot(df)
```

<seaborn.axisgrid.PairGrid at 0x1a667e5ea90>



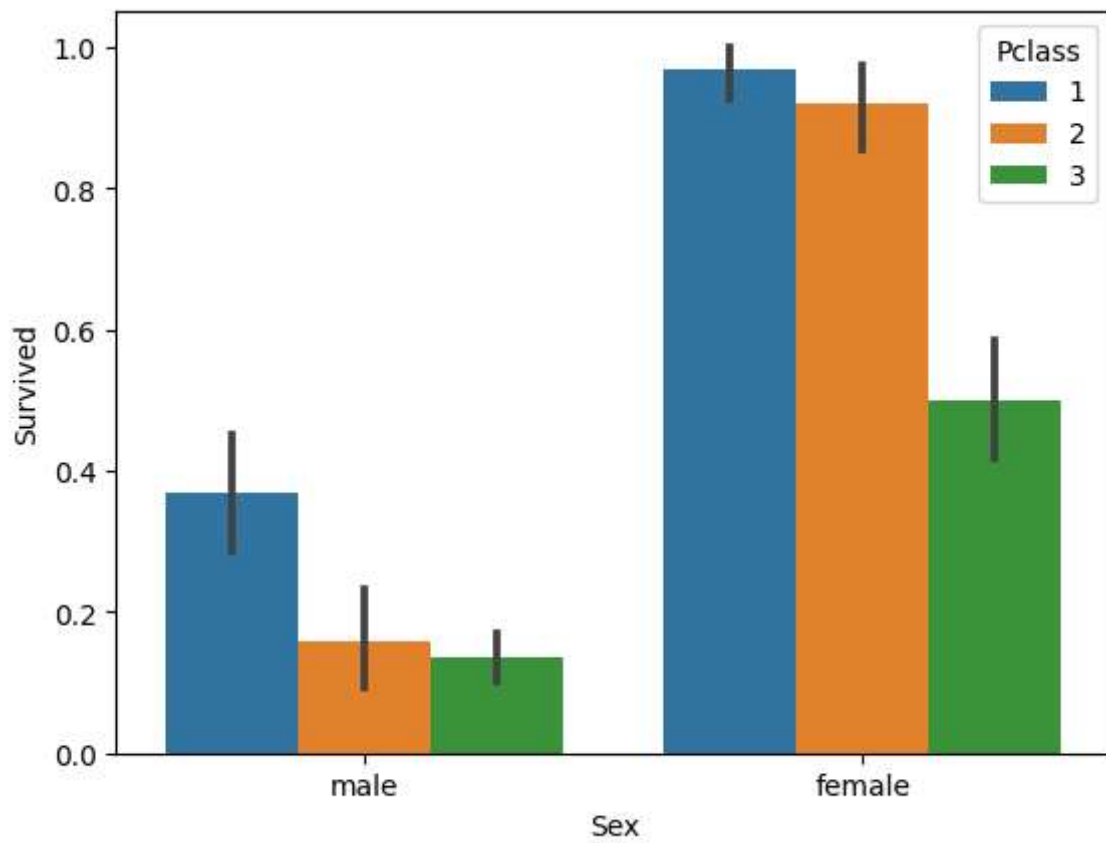
```
sns.scatterplot(x="Embarked",y="Age",data=df)
```

```
<Axes: xlabel='Embarked', ylabel='Age'>
```



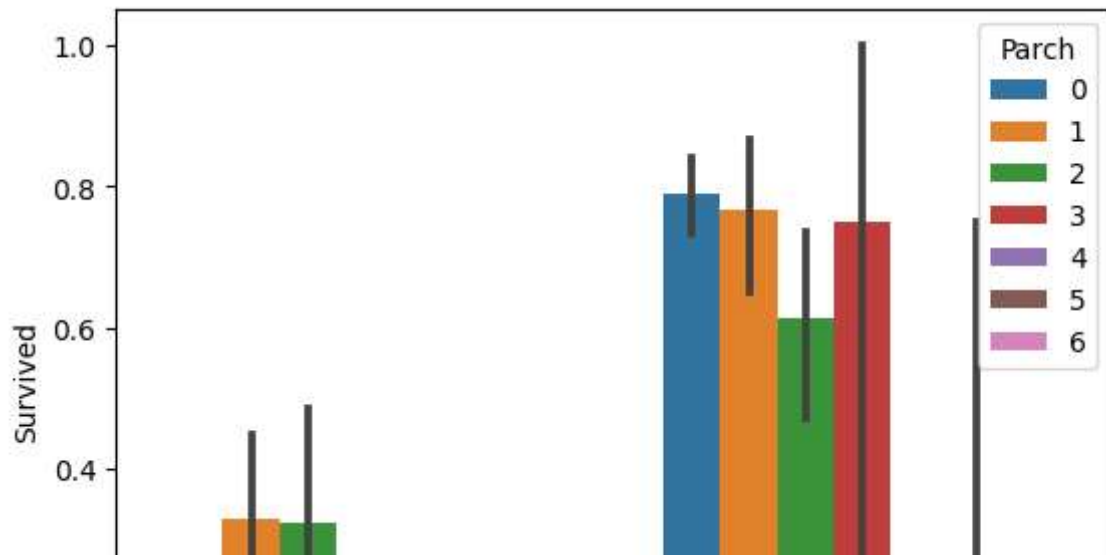
```
sns.barplot(x="Sex",y="Survived",data=df,hue="Pclass")
```

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



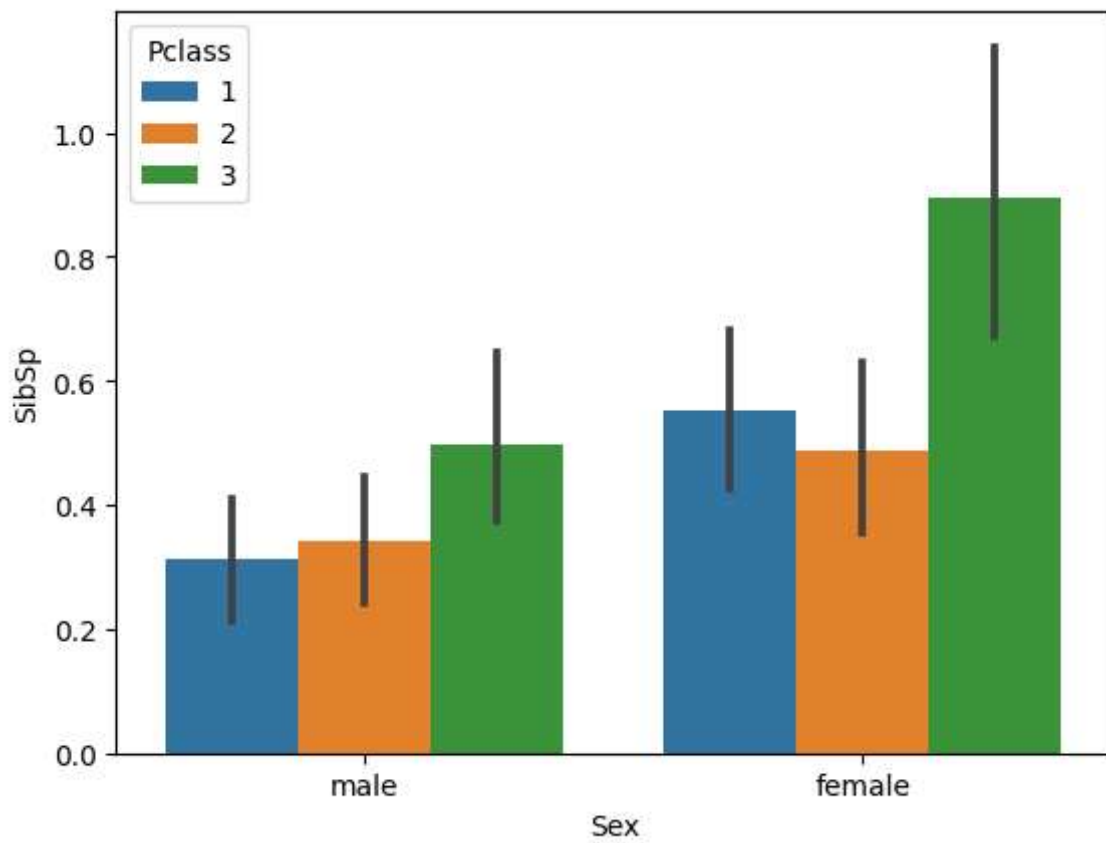
```
sns.barplot(x="Sex",y="Survived",data=df,hue="Parch")
```

<Axes: xlabel='Sex', ylabel='Survived'>



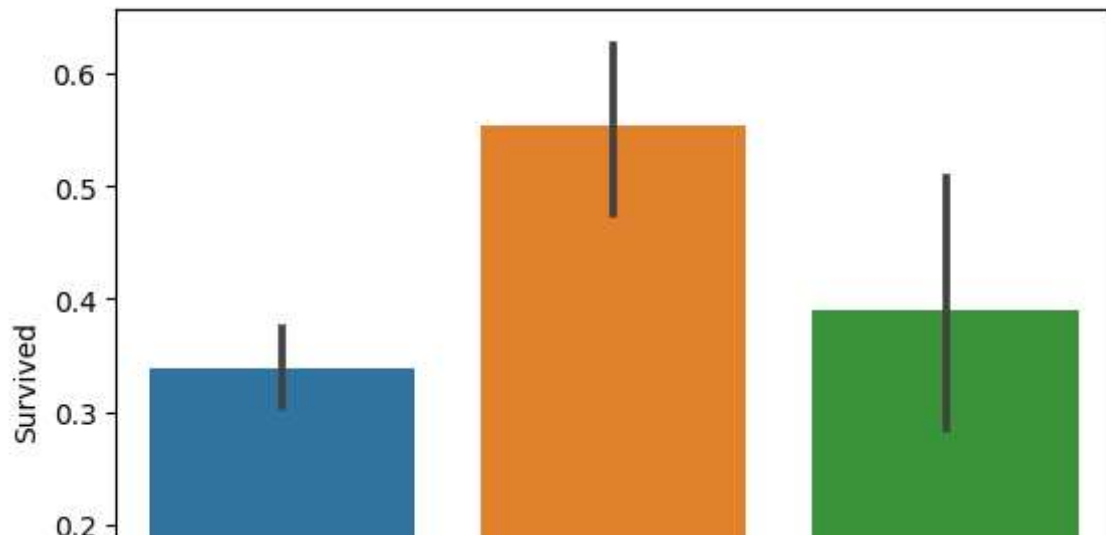
```
sns.barplot(x="Sex",y="SibSp",data=df,hue="Pclass")
```

<Axes: xlabel='Sex', ylabel='SibSp'>



```
sns.barplot(x="Embarked",y="Survived",data=df)
```

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



```
sns.distplot(df["Survived"])
```

C:\Users\nitin\AppData\Local\Temp\ipykernel_17756\41727483.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

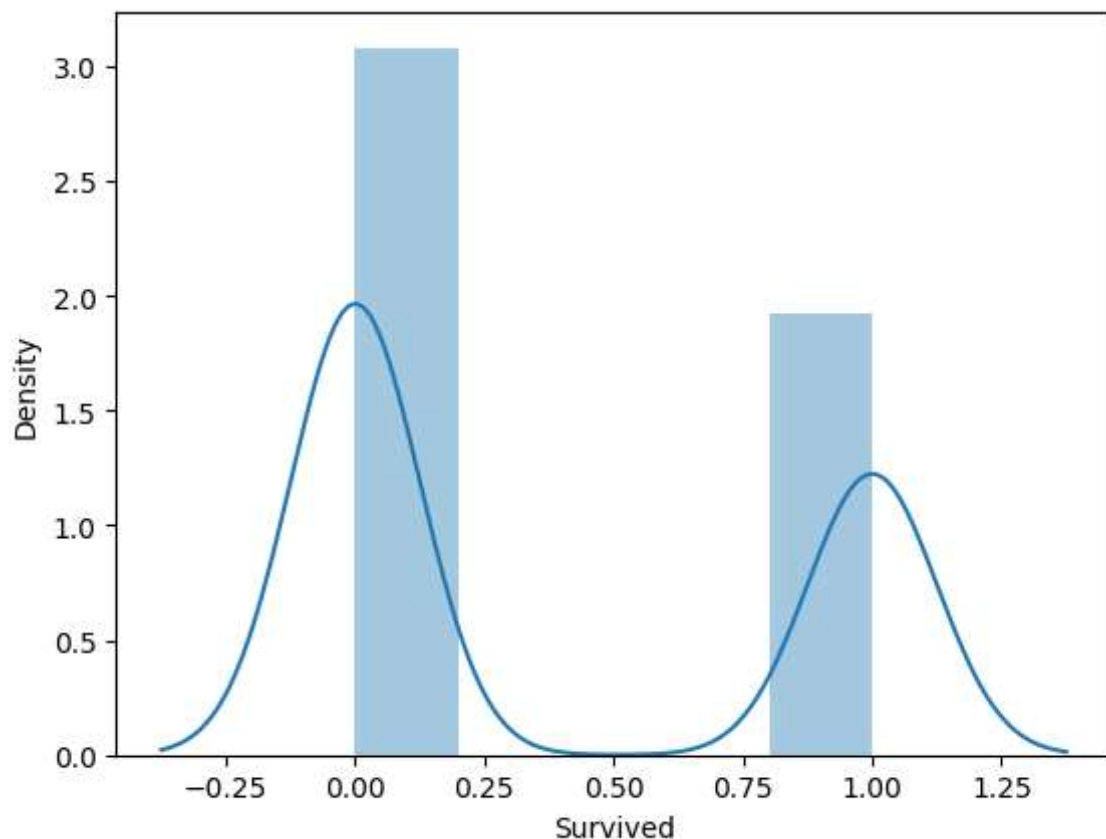
Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

<https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(df["Survived"])
```

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

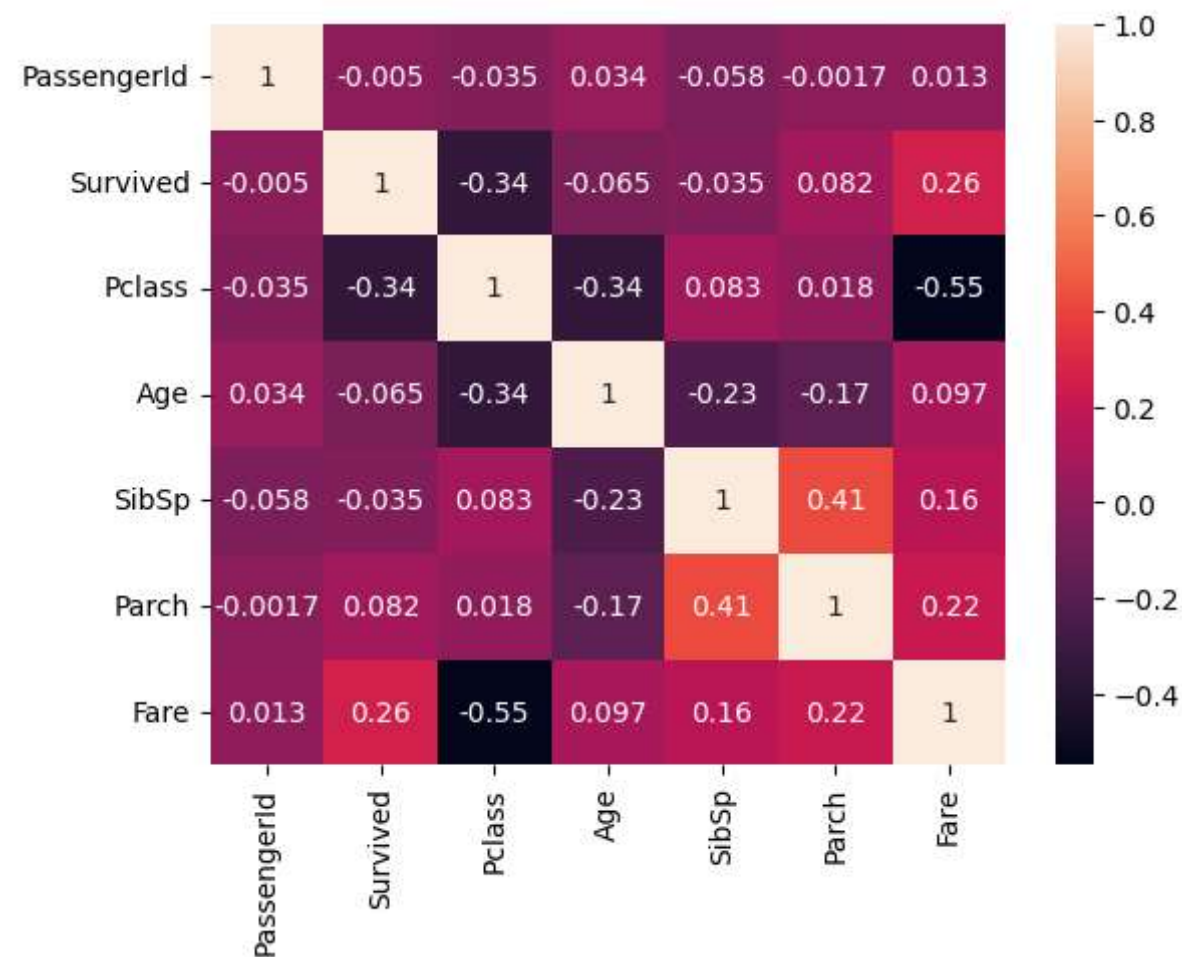



```
corr=df.corr(numeric_only=True)
corr
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.034212	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.064910	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.339898	0.083081	0.018443	-0.549500
Age	0.034212	-0.064910	-0.339898	1.000000	-0.233296	-0.172482	0.096688
SibSp	-0.057527	-0.035322	0.083081	-0.233296	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.172482	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096688	0.159651	0.216225	1.000000

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

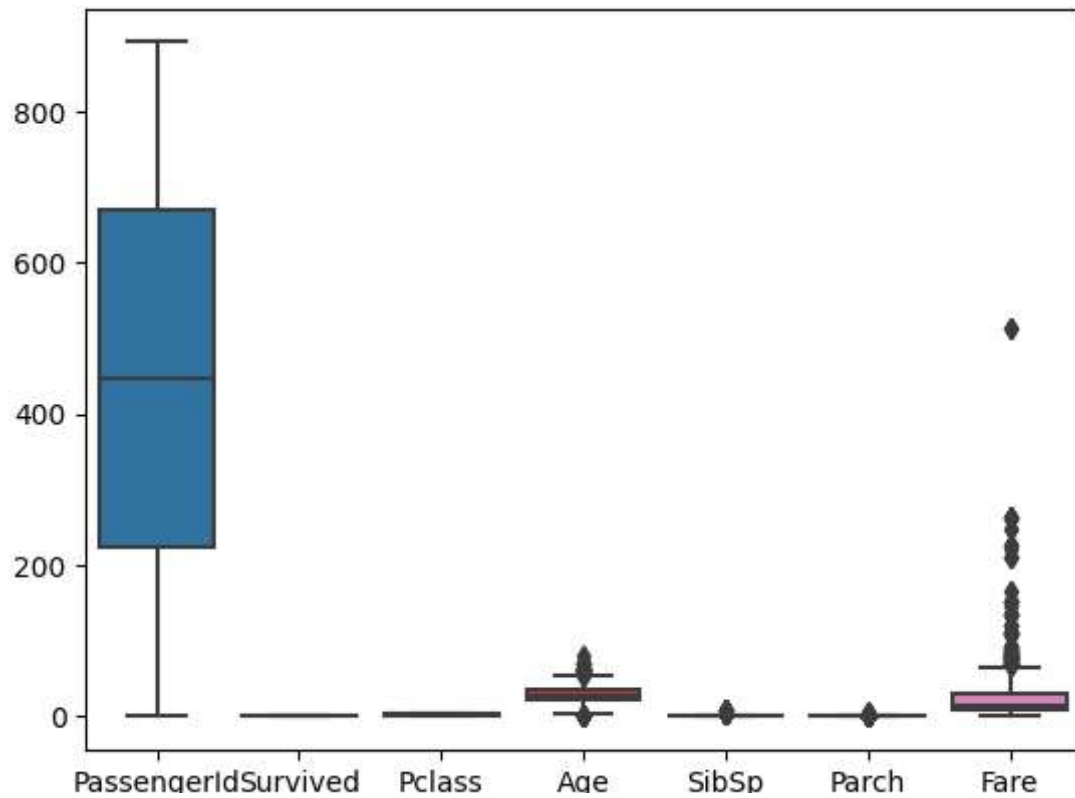
<Axes: >



▼ Outlier Detection

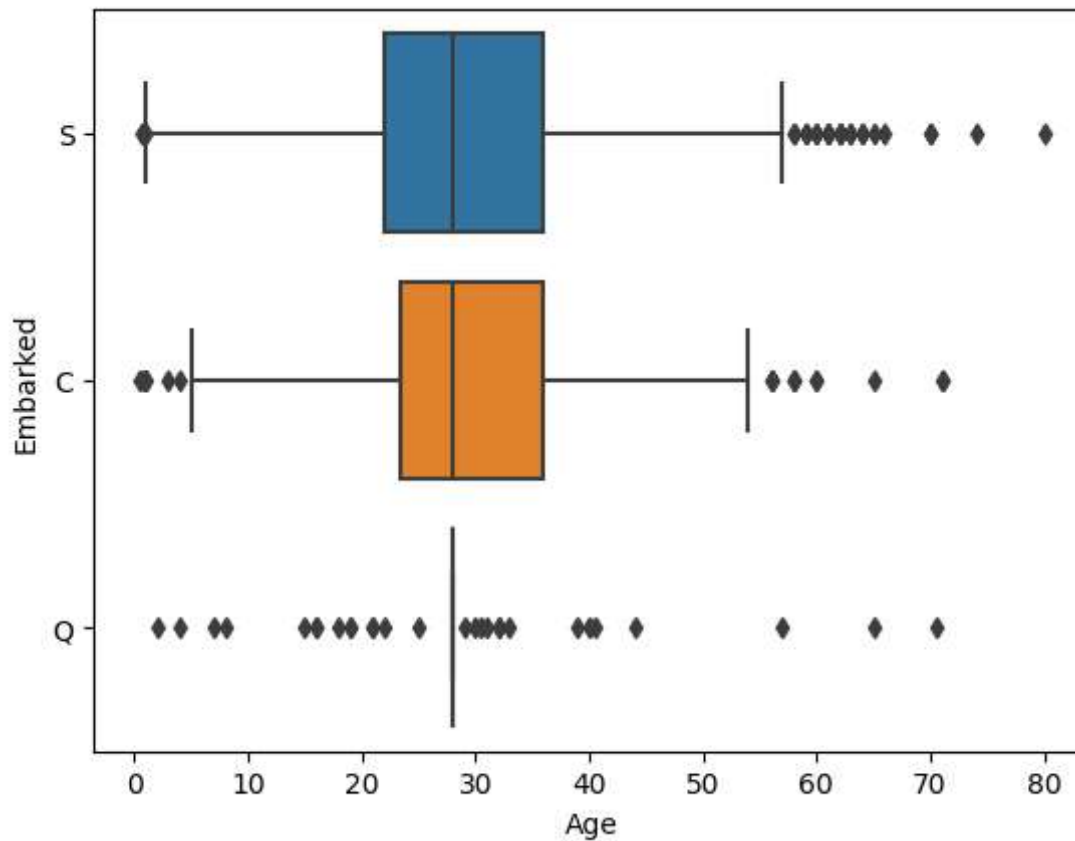
```
sns.boxplot(df)
```

<Axes: >



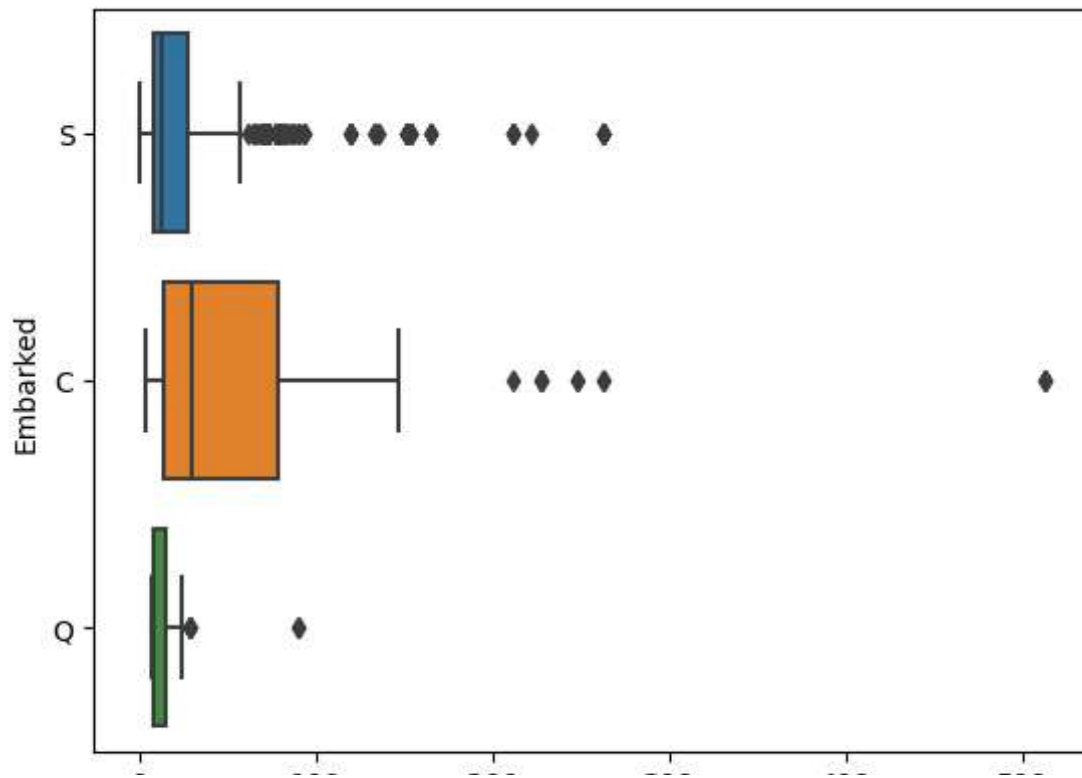
```
sns.boxplot(data=df,x="Age",y="Embarked")
```

<Axes: xlabel='Age', ylabel='Embarked'>



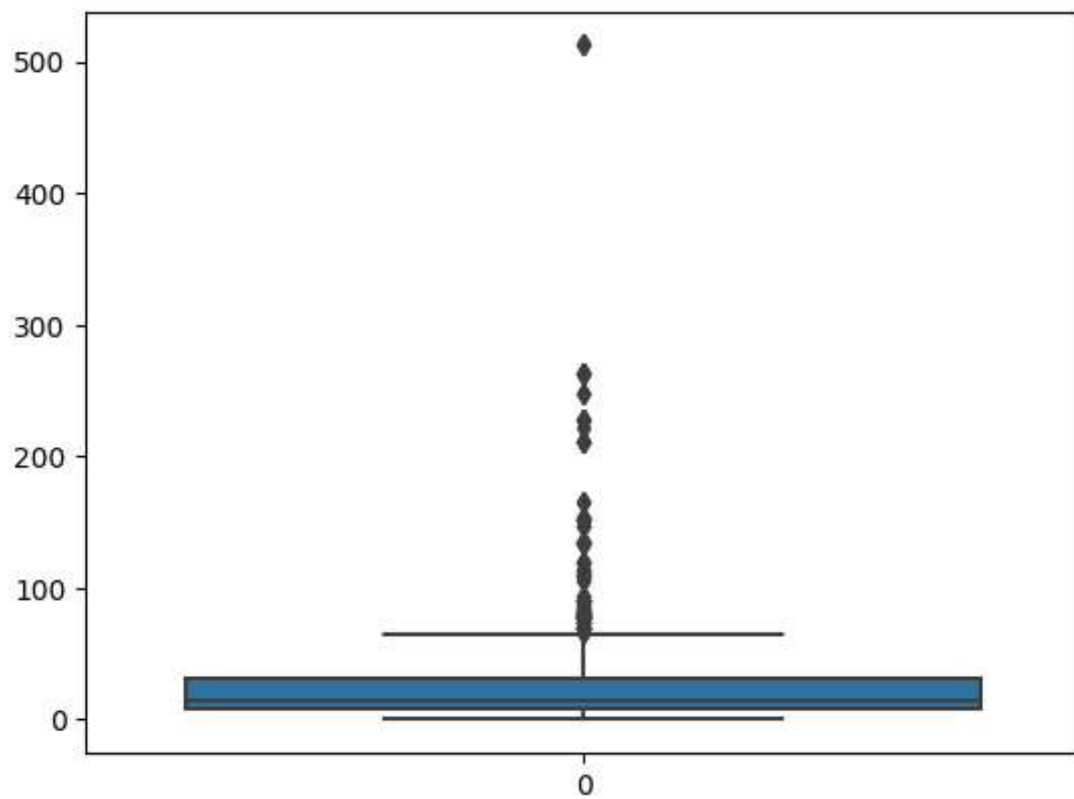
```
sns.boxplot(data=df,x="Fare",y="Embarked")
```

```
<Axes: xlabel='Fare', ylabel='Embarked'>
```



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

```
<Axes: >
```



```
df["Age"].skew()
```

```
0.5102446555756495
```

```
df["Fare"].skew() # as skewnwss should be -1 to +1 is normal range but here we are having
```

4.787316519674893

```
df["Fare"].median()
```

14.4542

```
Q1 = df['Fare'].quantile(0.25)
```

```
Q3 = df['Fare'].quantile(0.75)
```

```
IQR = Q3 - Q1
```

```
width = 1.5
```

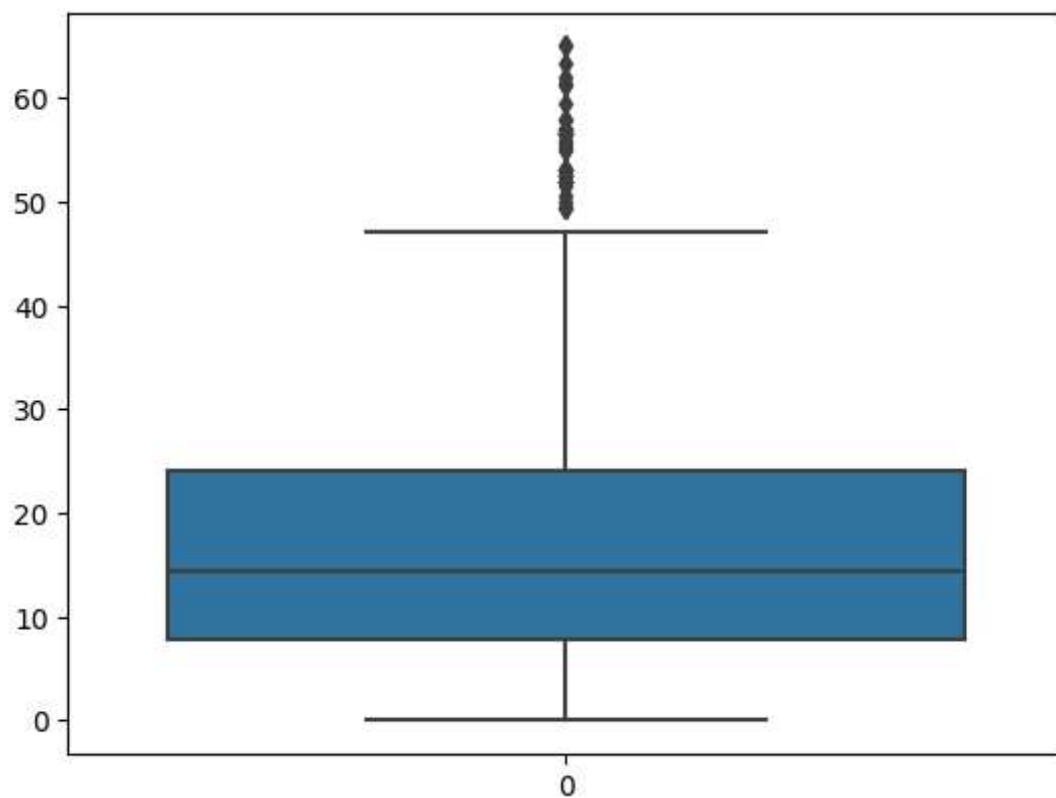
```
lower_limit = Q1 -(width*IQR)
```

```
upper_limit = Q3 + (width*IQR)
```

```
df['Fare']=np.where(df['Fare']>upper_limit,14.4542,np.where(df['Fare']<lower_limit,14.4542
```

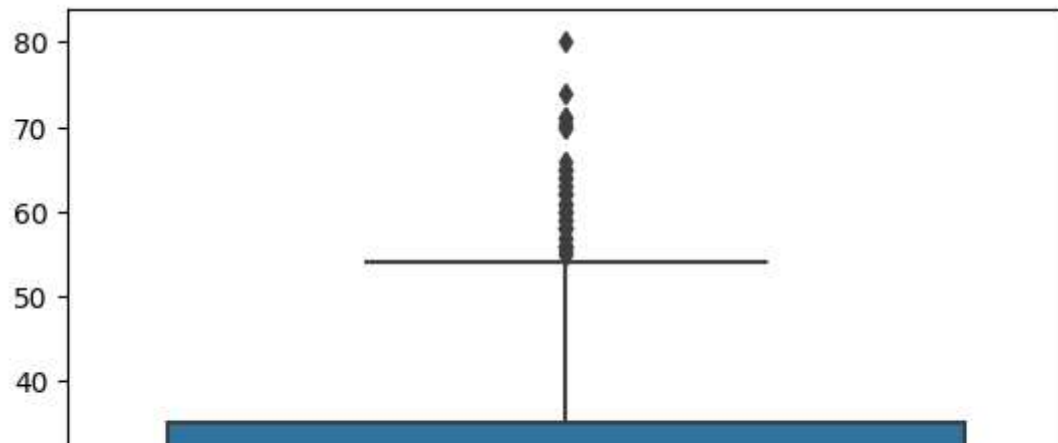
```
sns.boxplot(df["Fare"])
```

<Axes: >



```
sns.boxplot(df.Age)
```

<Axes: >



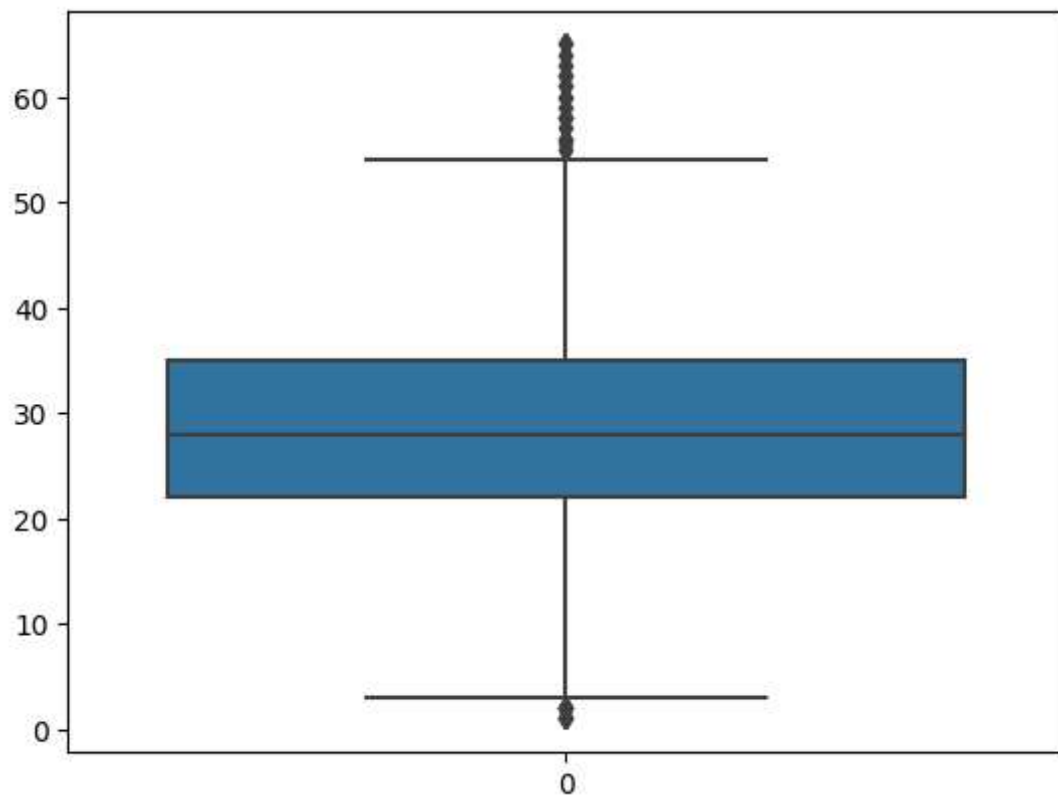
```
print(df.Age.median())  
print(df.Age.shape)
```

```
28.0  
(891,)
```

```
p=df["Age"].quantile(0.99)  
p1=df['Age'].quantile(0.01)  
df=df[df['Age']<=p]  
df=df[df['Age']>=p1]
```

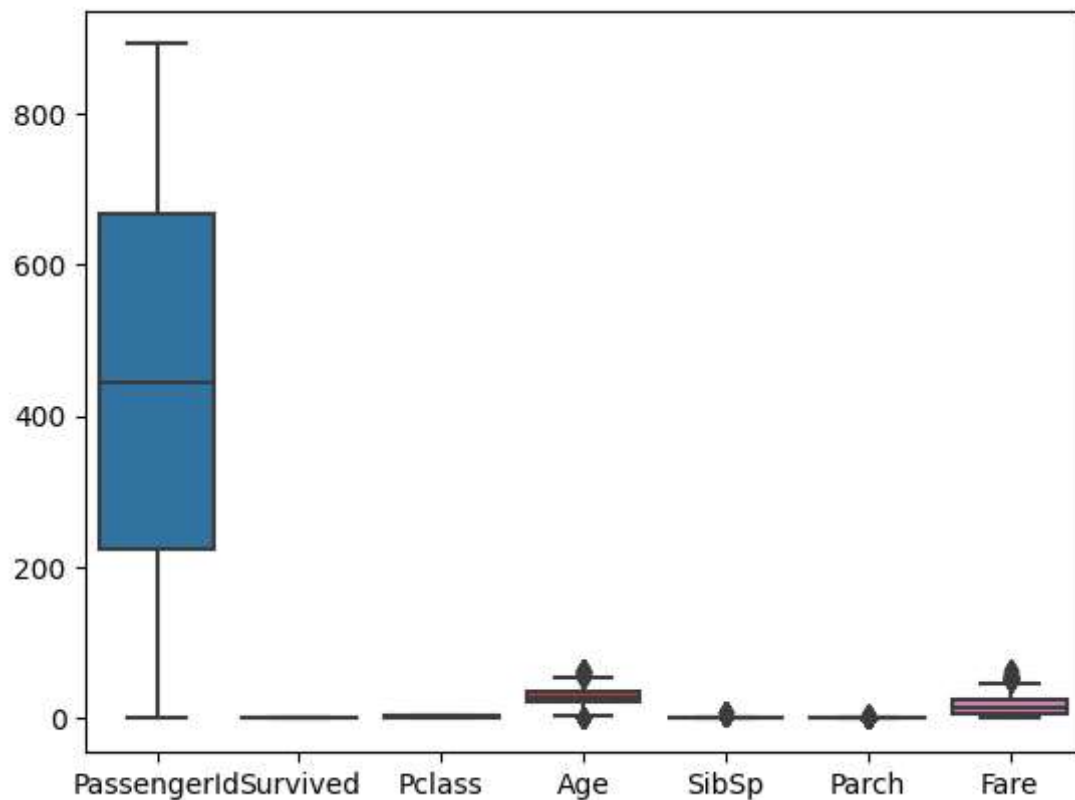
```
sns.boxplot(df.Age)
```

<Axes: >



```
sns.boxplot(df)
```

<Axes: >



```
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.0	1	0
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0
3	4	1	1	Futrelle, Mrs. Jacques	female	35.0	1	0

```
df.shape
```

```
(876, 12)
```

▼ Splitting Dependent and Independent variables

```
df.drop(["PassengerId", "Name", "Ticket", "Cabin"], axis=1, inplace=True)
df.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S

```
X=df.iloc[:,1:]
```

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

0	1	1	female	38.0	1	0	14.4542	C
---	---	---	--------	------	---	---	---------	---

```
X.head()
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	male	22.0	1	0	7.2500	S
1	1	female	38.0	1	0	14.4542	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

```
y.head()
```

	Survived
0	0
1	1
2	1
3	1
4	0

```
y=y.squeeze()
```

```
type(X)
```

```
pandas.core.frame.DataFrame
```

```
type(y)
```

```
pandas.core.series.Series
```

```
y.head()
```

0	0
1	1
2	1
3	1
4	0

Name: Survived, dtype: int64

▼ Perform Encoding

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
X["Sex"]=le.fit_transform(X["Sex"])
mapping1=dict(zip(le.classes_,range(len(le.classes_))))
X["Embarked"]=le.fit_transform(X["Embarked"])
mapping2=dict(zip(le.classes_,range(len(le.classes_))))

print("For Sex Column :",mapping1)
print("For Embarked Column :",mapping2)

For Sex Column : {'female': 0, 'male': 1}
For Embarked Column : {'C': 0, 'Q': 1, 'S': 2}
```

X

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	1	22.0	1	0	7.2500	2
1	1	0	38.0	1	0	14.4542	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
...
886	2	1	27.0	0	0	13.0000	2
887	1	0	19.0	0	0	30.0000	2
888	3	0	28.0	1	2	23.4500	2
889	1	1	26.0	0	0	30.0000	0
890	3	1	32.0	0	0	7.7500	1

876 rows × 7 columns

y

0	0
1	1
2	1
3	1
4	0
...	..
886	0
887	1


```

888      0
889      1
890      0
Name: Survived, Length: 876, dtype: int64

```

▼ Feature Scaling

```

from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
X_Scale=pd.DataFrame(ss.fit_transform(X),columns=X.columns)

```

```
X_Scale.head()
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0.821711	0.743768	-0.589744	0.430836	-0.467137	-0.793895	0.582594
1	-1.573693	-1.344504	0.719906	0.430836	-0.467137	-0.227705	-1.956472
2	0.821711	-1.344504	-0.262332	-0.472065	-0.467137	-0.740846	0.582594
3	-1.573693	-1.344504	0.474346	0.430836	-0.467137	2.809531	0.582594
4	0.821711	0.743768	0.474346	-0.472065	-0.467137	-0.731022	0.582594

```
y.head()
```

```

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

```

```
X_Scale.shape
```

```
(876, 7)
```

```
y.shape
```

```
(876,)
```

▼ Splitting Data into Train and Test

```

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X_Scale,y,test_size=0.2,random_state=0)

print(X_train,"\n",X_test,"\n","\n",y_train,"\n",y_test)

```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
45	0.821711	0.743768	-0.098626	0.430836	-0.467137	-0.145514	-0.686939
172	0.821711	0.743768	-0.098626	2.236638	0.772900	0.637785	0.582594
492	0.821711	0.743768	-0.426038	-0.472065	-0.467137	-0.751000	0.582594
820	0.821711	0.743768	-0.917157	-0.472065	-0.467137	-0.711374	0.582594
651	-1.573693	0.743768	1.702144	1.333737	-0.467137	-0.227705	0.582594
..
835	-1.573693	-1.344504	-0.098626	0.430836	-0.467137	-0.227705	-1.956472
192	0.821711	0.743768	-0.098626	-0.472065	-0.467137	-0.754599	-0.686939
629	-0.375991	0.743768	0.146934	0.430836	0.772900	0.699346	0.582594
559	0.821711	0.743768	-0.835304	-0.472065	-0.467137	-0.743141	0.582594
684	-1.573693	0.743768	2.520675	-0.472065	-0.467137	0.722923	0.582594

[700 rows x 7 columns]

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
141	-0.375991	0.743768	-0.835304	0.430836	0.772900	1.524558	0.582594
113	-0.375991	0.743768	-0.016772	0.430836	-0.467137	0.286740	0.582594
730	-1.573693	0.743768	-0.098626	-0.472065	-0.467137	0.994065	0.582594
294	-1.573693	0.743768	-0.098626	-0.472065	-0.467137	1.033360	0.582594
261	-0.375991	0.743768	0.556200	-0.472065	-0.467137	-0.538472	0.582594
..
578	-1.573693	-1.344504	-0.917157	-0.472065	2.012937	-0.227705	0.582594
773	0.821711	0.743768	-0.344185	-0.472065	-0.467137	-0.793895	0.582594
522	-0.375991	0.743768	-0.507891	1.333737	0.772900	-0.459881	0.582594
780	0.821711	-1.344504	-0.098626	6.751142	2.012937	-0.227705	0.582594
54	-1.573693	0.743768	-0.098626	-0.472065	-0.467137	1.426319	0.582594

[176 rows x 7 columns]

46	0
176	0
499	0
834	0
660	1
..	
849	1
196	0
637	0
566	0
694	0

Name: Survived, Length: 700, dtype: int64

145	0
117	0
740	1
298	1
265	0
..	
585	1
785	0
529	0
792	0
55	1

Name: Survived, Length: 176, dtype: int64

```
print(X_train.shape,X_test.shape,y_train.shape,y_test.shape)
```

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

▼ Preprocessing Done

▼ Testing for accuracy

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

```
▼ LogisticRegression
LogisticRegression()
```

```
y_predict=lr.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_predict)
```

```
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8295454545454546
```