ASSIGNMENT-03

Aniket Chattopadhyay - 21BEC1564

Date of Submission : 21/09/2023

Perform Data preprocessing on Titanic dataset

1) Import the necessary libraries

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

2) Import the dataset

In [ ]:
```
df=pd.read_csv("Titanic-Dataset.csv")
```

In [ ]:
```
df.head()
```

Out[ ]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	E
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 [ ]:
```
df.tail()
```

Out[ ]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Em
886	887	0	2	Montvila, Rev.	male	27.0	0	0	211536	13.00	NaN	

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Em
				Juozas								
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	

In [ ]:
```
df.shape
```

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

In [ ]:
```
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
```

In [ ]:
```
df.describe()
```

Out[ ]:

	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

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
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

In [ ]:
```
df.corr()
```

```
<ipython-input-38-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only
in DataFrame.corr is deprecated. In a future version, it will default to False. Sele
ct only valid columns or specify the value of numeric_only to silence this warning.
  df.corr()
```

Out[ ]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

In [ ]:
```
df.isnull().any()
```

Out[ ]:
```
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 [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
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
```

3) Checking for null values

In [ ]:
```
df["Age"].fillna(df["Age"].mean(),inplace=True)
```

In [ ]:
```
df["Cabin"].fillna(df["Cabin"].mode()[0],inplace=True)
```

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

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
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
```

In [ ]:
```
df.head(10)
```

Out[ ]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450	8.0500	
5	6	0	3	Moran, Mr. James	male	29.699118	0	0	330877	8.4583	

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
6	7	0	1	McCarthy, Mr. Timothy J	male	54.000000	0	0	17463	51.8625	
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.000000	3	1	349909	21.0750	
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.000000	0	2	347742	11.1333	
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.000000	1	0	237736	30.0708	

Data Visualization

In [ ]:
```
df["Sex"].value_counts()
```

Out[ ]:
```
male      577
female    314
Name: Sex, dtype: int64
```

In [ ]:
```
df["Survived"].value_counts()
```

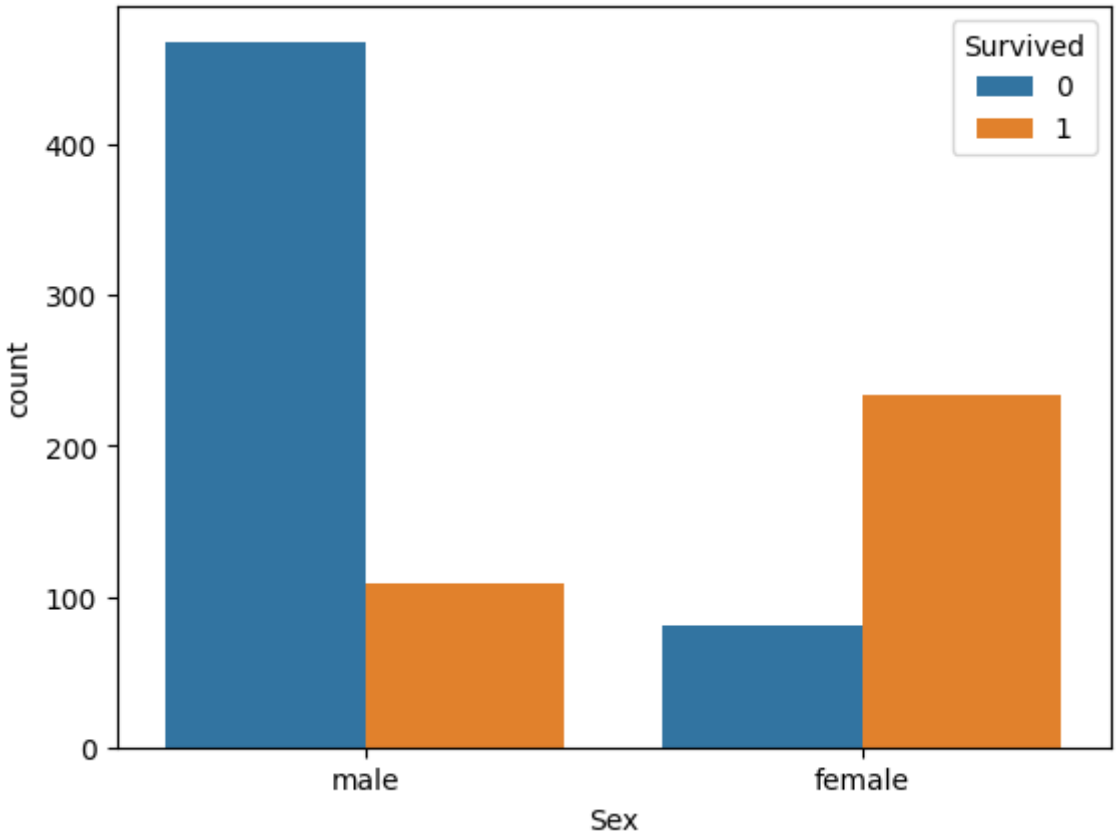
Out[ ]:
```
0    549
1    342
Name: Survived, dtype: int64
```

In [ ]:
```
df["Pclass"].value_counts()
```

Out[ ]:
```
3    491
1    216
2    184
Name: Pclass, dtype: int64
```

In [ ]:
```
sns.countplot(x="Sex",data=df,hue="Survived")
```

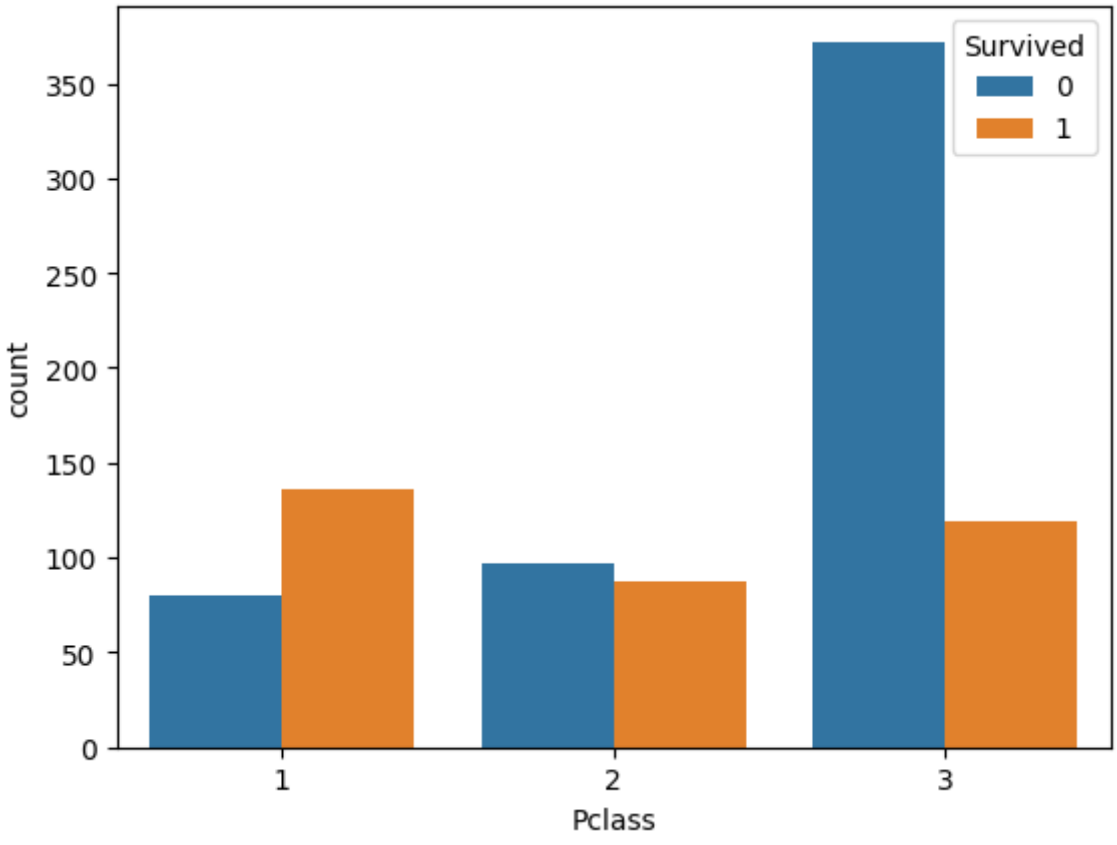
Out[ ]:
```
<Axes: xlabel='Sex', ylabel='count'>
```



Inference: With the help of the coutplot we can see that large number of male did not survived (more than 400).

In [ ]:
```
sns.countplot(x="Pclass",data=df,hue="Survived")
```

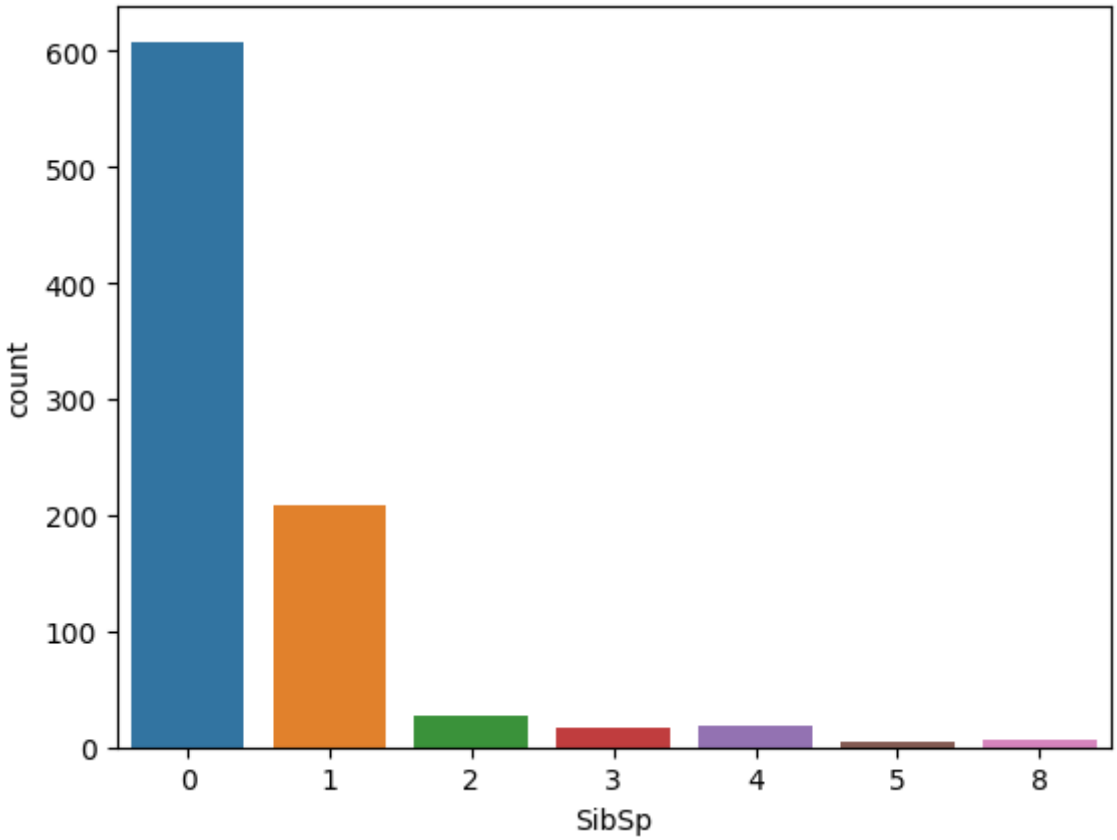
Out[ ]:
```
<Axes: xlabel='Pclass', ylabel='count'>
```



Inference: With help of this graph we can see that the large number of the 3rd class or say lower class people did not survived.

In [ ]:
```
sns.countplot(x="SibSp",data=df)
```

Out[ ]:
```
<Axes: xlabel='SibSp', ylabel='count'>
```

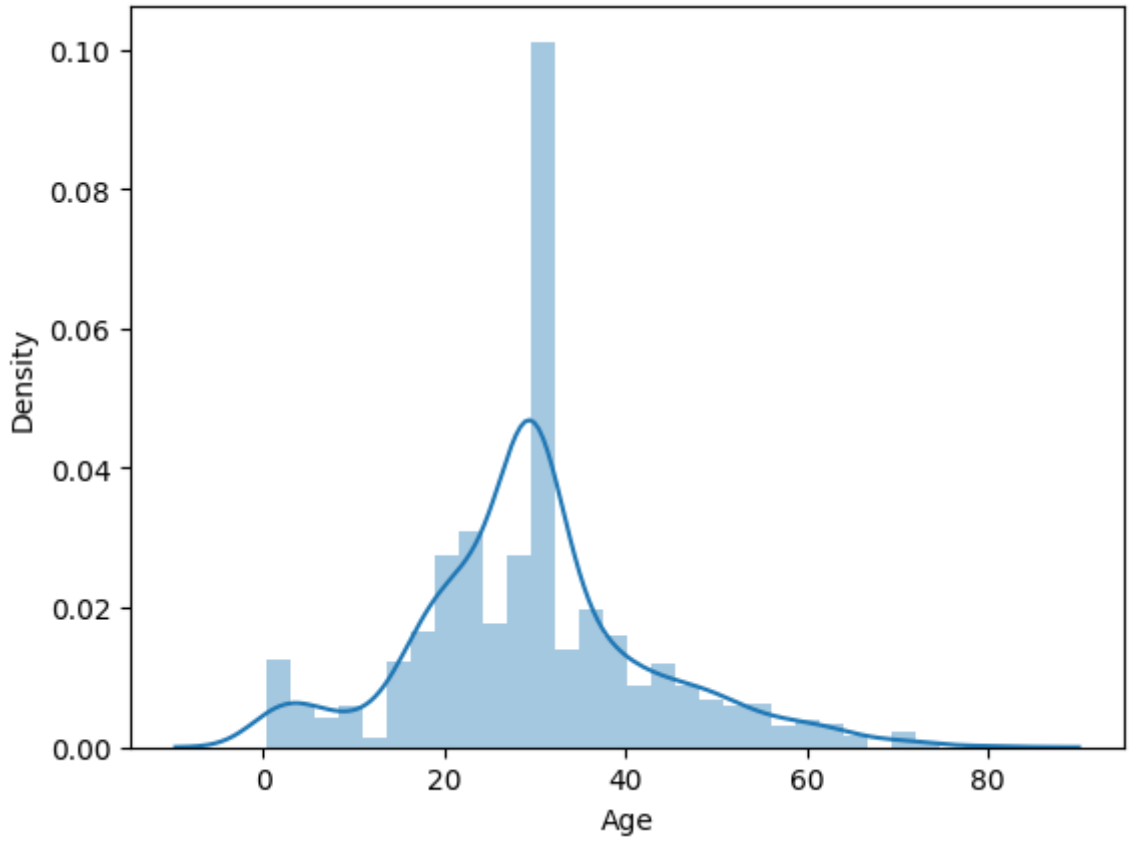


Inference: Here we can see that majority of the passengers were single and second highest were we can say pessengers with their 1 sibling or spouse etc

In [ ]:
```
sns.distplot(df["Age"])
```

```
<ipython-input-54-cf0334540b62>: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["Age"])
```

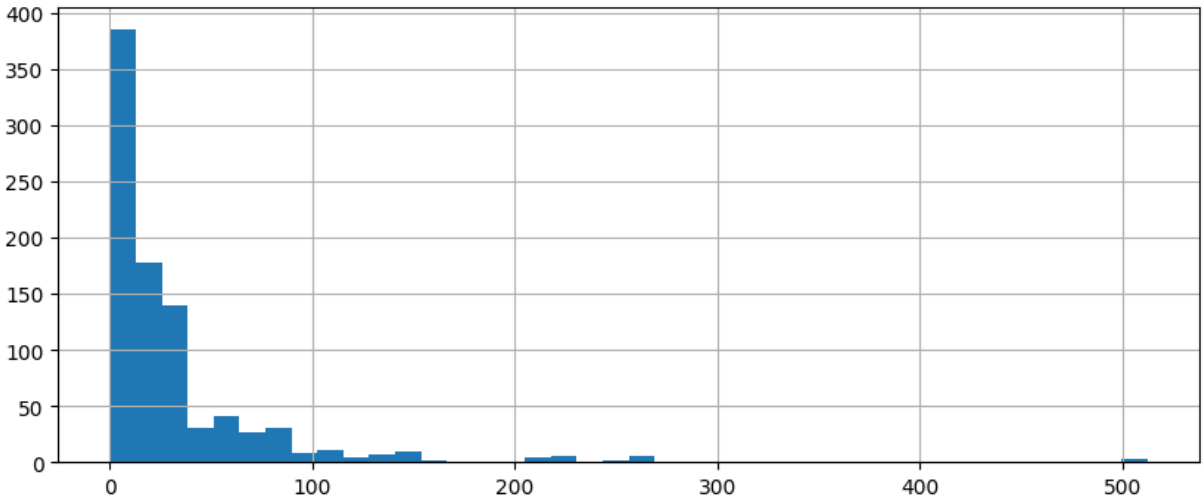
Out[ ]:
```
<Axes: xlabel='Age', ylabel='Density'>
```



Inference: This histogram helps us to understand that many of the passengers that were present in the titanic were of age range 28-30 years.

In [ ]:
```
df['Fare'].hist(bins=40,figsize=(10,4))
```

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

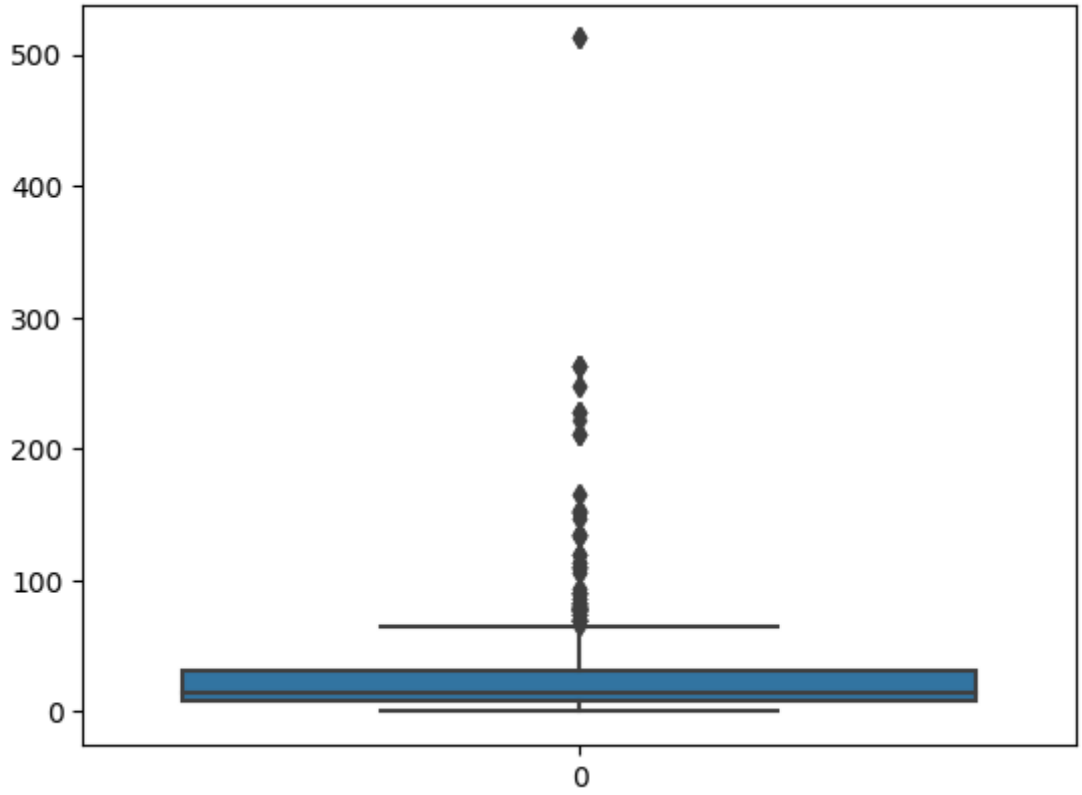


Inference: With this histogram this we can find that cheaper tickets were sold, most passengers were lower class.

5) Outlier detection

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

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



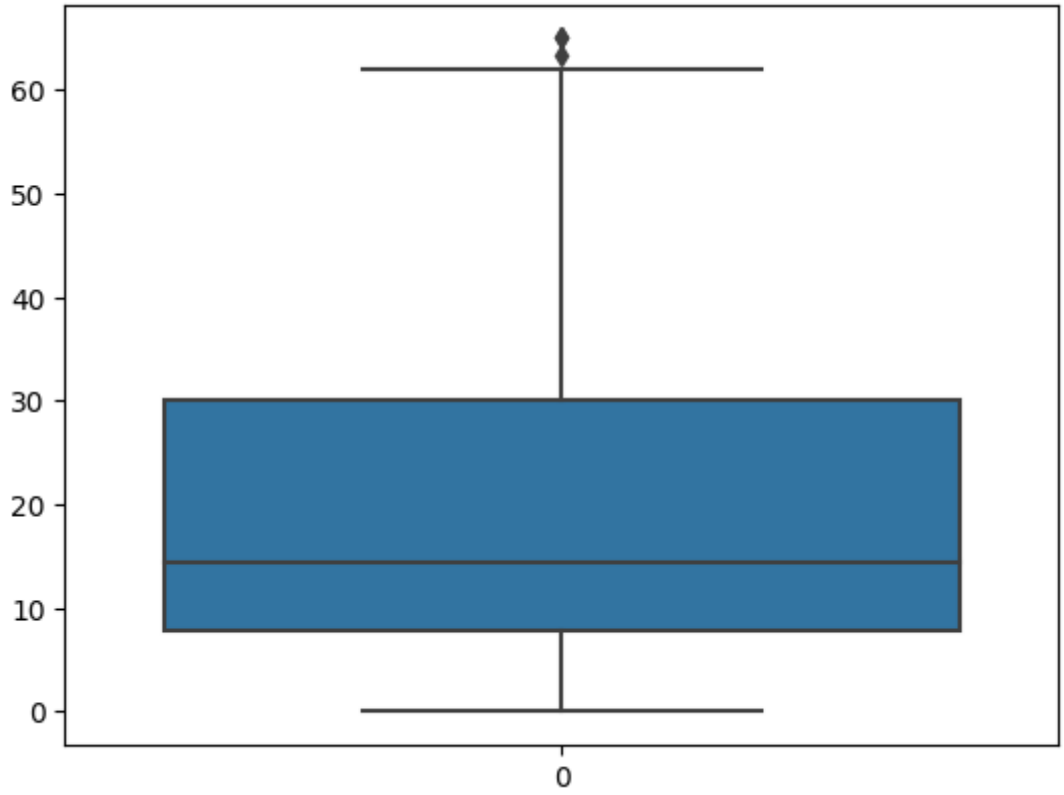
In [ ]:
```
q1=df.Fare.quantile(0.25)
q3=df.Fare.quantile(0.75)
print(q1)
print(q3)
IQR=q3-q1
print(IQR)
upper_limit = q3+1.5*IQR
print(upper_limit)
lower_limit = q1-1.5*IQR
print(lower_limit)
df.median()
```

```
7.9104
31.0
23.0896
65.6344
-26.724
<ipython-input-58-e75a8fb19795>:11: FutureWarning: The default value of numeric_only in DataFrame.median is deprecated. In a future version, it will default to False. In addition, specifying 'numeric_only=None' is deprecated. Select only valid columns or specify the value of numeric_only to silence this warning.
  df.median()
```

Out[ ]:
```
PassengerId    446.000000
Survived         0.000000
Pclass           3.000000
Age             29.699118
SibSp            0.000000
Parch            0.000000
Fare            14.454200
dtype: float64
```

In [ ]:
```
df['Fare']= np.where(df['Fare']>upper_limit,30,df['Fare'])
sns.boxplot(df["Fare"])
```

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



6) Seperate Dependent and Independent variables

In [ ]:
```
df.head()
```

Out[ ]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	E
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250	B96 B98	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	30.000	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925	B96 B98	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100	C123	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.050	B96 B98	

In [ ]:
```
x=df.drop(labels='Survived',axis=1)
x.head()
```

Out[ ]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250	B96 B98	S
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	30.000	C85	C
2	3	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925	B96 B98	S
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100	C123	S
4	5	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.050	B96 B98	S

In [ ]:

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

Out[ ]:

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

891 rows × 1 columns

7) Encoding

In [ ]:

```
df.head()
```

Out[ ]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	E
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250	B96 B98	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	30.000	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925	B96 B98	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100	C123	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.050	B96 B98	

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

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

Out[ ]:
```
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: int64
```

In [ ]:
```
x["Sex"].value_counts()
```

Out[ ]:
```
1    577
0    314
Name: Sex, dtype: int64
```

In [ ]:
```
x["Sex"].nunique()
```

Out[ ]:
```
2
```

In [ ]:
```
x.Embarked.value_counts()
```

Out[ ]:
```
S    646
C    168
Q     77
Name: Embarked, dtype: int64
```

In [ ]:
```
x.shape
```

Out[ ]:
```
(891, 11)
```

In [ ]:
```
embarked=pd.get_dummies(x["Embarked"],drop_first=True)
embarked
```

Out[ ]:

	Q	S
0	0	1
1	0	0
2	0	1
3	0	1
4	0	1
...	...	...
886	0	1
887	0	1
888	0	1
889	0	0
890	1	0

891 rows × 2 columns

In [ ]:
```
x=pd.concat([x,embarked],axis=1)
```

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

Out[ ]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Q
0	1	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.250	B96 B98	S	0
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	30.000	C85	C	0
2	3	3	Heikkinen, Miss. Laina	0	26.0	0	0	STON/O2. 3101282	7.925	B96 B98	S	0
3	4	1	Futrelle, Mrs.	0	35.0	1	0	113803	53.100	C123	S	0

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Q
			Jacques Heath (Lily May Peel)									
4	5	3	Allen, Mr. William Henry	1	35.0	0	0	373450	8.050	B96 B98	S	0

In [ ]:
```
x.drop(["Embarked"],axis=1,inplace=True)
```

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

Out[ ]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Q	S
0	1	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.250	B96 B98	0	1
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	30.000	C85	0	0
2	3	3	Heikkinen, Miss. Laina	0	26.0	0	0	STON/O2. 3101282	7.925	B96 B98	0	1
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	1	0	113803	53.100	C123	0	1
4	5	3	Allen, Mr. William Henry	1	35.0	0	0	373450	8.050	B96 B98	0	1

In [ ]:
```
x.drop(["Name"],axis=1,inplace=True)
```

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

Out[ ]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Q	S
0	1	3	1	22.0	1	0	A/5 21171	7.250	B96 B98	0	1
1	2	1	0	38.0	1	0	PC 17599	30.000	C85	0	0
2	3	3	0	26.0	0	0	STON/O2. 3101282	7.925	B96 B98	0	1
3	4	1	0	35.0	1	0	113803	53.100	C123	0	1
4	5	3	1	35.0	0	0	373450	8.050	B96 B98	0	1

In [ ]:
```
x.drop(["Ticket"],axis=1,inplace=True)
```

In [ ]:
```
x.drop(["Cabin"],axis=1,inplace=True)
```

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

Out[ ]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Q	S
0	1	3	1	22.0	1	0	7.250	0	1
1	2	1	0	38.0	1	0	30.000	0	0
2	3	3	0	26.0	0	0	7.925	0	1
3	4	1	0	35.0	1	0	53.100	0	1
4	5	3	1	35.0	0	0	8.050	0	1

Inference: I have dropped the Name, Ticket and Cabin columns because they are all object type and can't be converted from categorical to numerical type.

8) Splitting into training and testing set

We need to split a dataset into train and test sets to evaluate how well our machine learning model performs. The train set is used to fit the model, and the statistics of the train set are known. The second set is called the test data set, this set is solely used for predictions.

In [ ]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

In [ ]:
```
x_train.shape,x_test.shape
```

Out[ ]:
```
((623, 9), (268, 9))
```

In [ ]:
```
y_train.shape,y_test.shape
```

Out[ ]:
```
((623, 1), (268, 1))
```

9) Perform feature scaling

Feature Scaling is a technique to standardize the independent features present in the data in a fixed range.

In [ ]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
```

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

In [ ]:
```
x_train
```

Out[ ]:
```
array([[ 1.59014094, -1.5325562 ,  0.72592065, ...,  0.49567463,
        -0.31426968,  0.59774449],
       [-1.52952238, -1.5325562 , -1.37756104, ...,  0.75298828,
        -0.31426968, -1.67295561],
```

```
       [-0.23515275,  0.84844757,  0.72592065, ...,  2.01345222,
        -0.31426968,  0.59774449],
       ...,
       [ 0.70655928,  0.84844757,  0.72592065, ..., -0.90774382,
         3.18198052, -1.67295561],
       [ 0.43528421,  0.84844757, -1.37756104, ..., -0.1867659 ,
        -0.31426968,  0.59774449],
       [ 0.91970398, -0.34205431,  0.72592065, ...,  1.42424126,
        -0.31426968,  0.59774449]])
```

In [ ]:

```
x_test
```

Out[ ]:

```
array([[ 0.21119888,  0.77963055,  0.76537495, ..., -0.29235767,
        -0.29158231, -1.51942159],
       [ 0.8106727 ,  0.77963055,  0.76537495, ..., -0.82457025,
        -0.29158231,  0.65814518],
       [-0.63903523,  0.77963055,  0.76537495, ...,  0.83755883,
         3.42956335, -1.51942159],
       ...,
       [ 0.70096507,  0.77963055,  0.76537495, ..., -0.29267353,
        -0.29158231, -1.51942159],
       [ 1.35137458,  0.77963055, -1.30654916, ..., -0.82874579,
        -0.29158231,  0.65814518],
       [-1.47751496, -1.64991582,  0.76537495, ...,  0.72937985,
        -0.29158231, -1.51942159]])
```