Name: Gongati Sree Nidhitha Reddy

Reg.No: 21BCE2409

1. Import necessary libraries

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

2. Import Dataset

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

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emba
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	
				Allen, Mr								

```
ds.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embark
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	

```
ds.tail()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
				Johnston								

```
ds.shape
```

```
(891, 12)
```

```
ds.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
```

```
ds.describe()
```

	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

```
ds.Survived.value_counts()
```

```
0    549
1    342
Name: Survived, dtype: int64
```

```
ds.Pclass.value_counts()
```

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

```
ds.SibSp.value_counts()
```

```
0    608
1    209
2     28
4     18
3     16
8      7
5      5
Name: SibSp, dtype: int64
```

```
ds.Parch.value_counts()
```

```
0    678
1    118
2     80
5      5
3      5
4      4
6      1
Name: Parch, dtype: int64
```

3. Handling Null Values

```
ds.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
```

```
ds.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
```

```
ds["Age"].fillna(ds["Age"].mean(),inplace=True)
ds
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin

```
ds["Cabin"].fillna(ds["Cabin"].mode()[0],inplace=True)
ds["Embarked"].fillna(ds["Embarked"].mode()[0],inplace=True)
ds
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.2500	B96 B98
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282	7.9250	B96 B98
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803	53.1000	C123
				Allen, Mr							

```
ds.isnull().any()
```

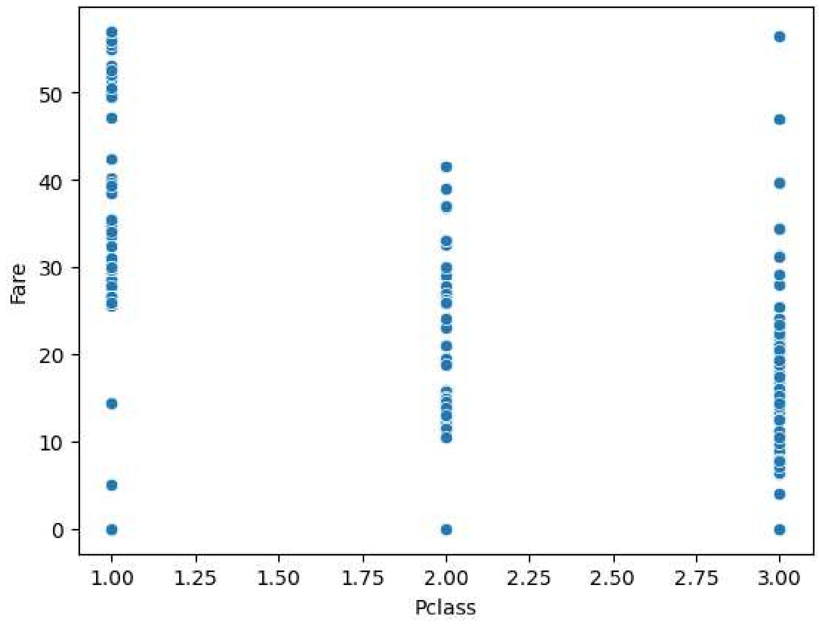
```
PassengerId    False
Survived       False
Pclass         False
Name           False
Sex            False
Age            False
SibSp          False
Parch          False
Ticket         False
Fare           False
Cabin          False
Embarked       False
dtype: bool
```

4. Data Visualization

1. Scatter Plot

```
sns.scatterplot(x="Pclass", y="Fare", data=ds)
```

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

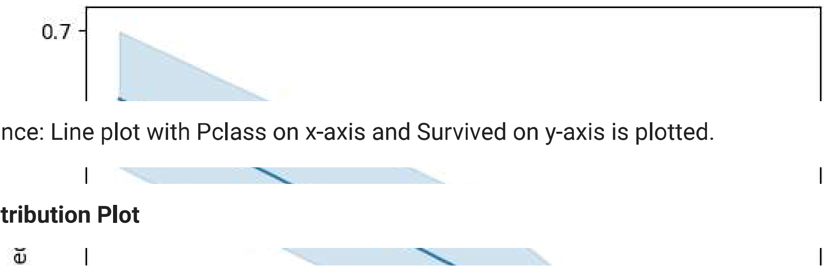


Inference: In the above graph Pclass is taken on x-axis and Fare is taken on y-axis. This is to analize the Fare of different Pclass

2. Line Plot

```
sns.lineplot(x="Pclass",y="Survived",data=ds)
```

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



Inference: Line plot with Pclass on x-axis and Survived on y-axis is plotted.

3. Distribution Plot

```
sns.distplot(ds["Fare"])
```

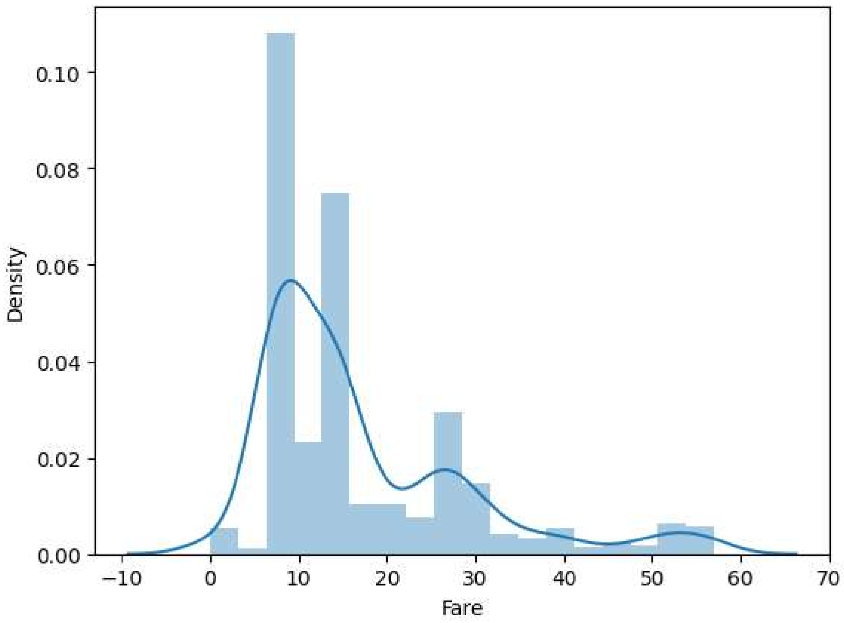
```
<ipython-input-77-76aa0978ee3c>:1: UserWarning:

`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(ds["Fare"])
<Axes: xlabel='Fare', ylabel='Density'>
```

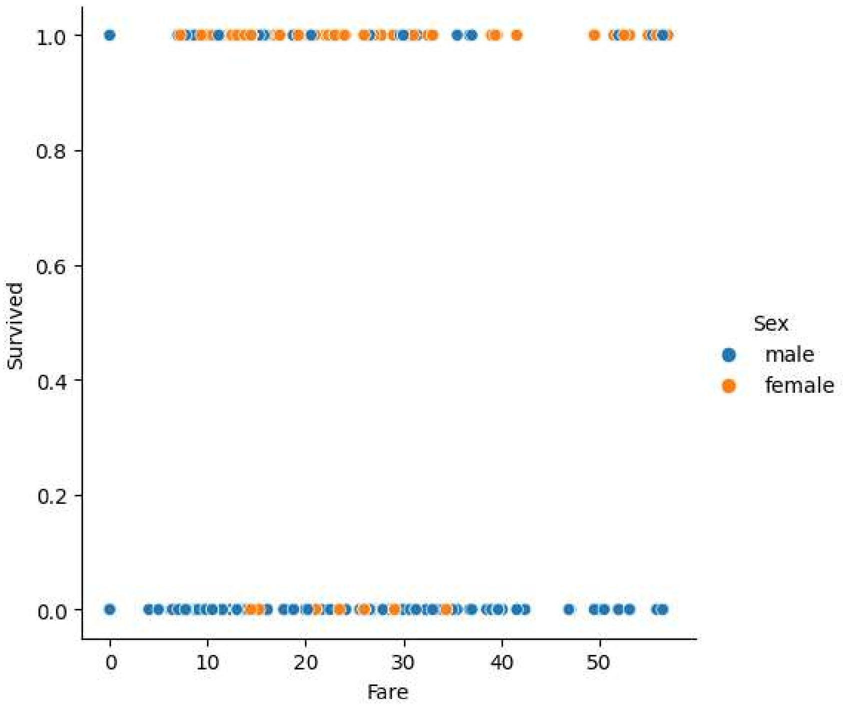


Inference: In this graph x-axis represents Fare and y-axis represents the probability density. The density of previous Fare increases until 10 and later decreases.

4. Relational plot

```
sns.relplot(x="Fare", y="Survived", data=ds, hue="Sex")
```

```
<seaborn.axisgrid.FacetGrid at 0x78958395afe0>
```



Inference: Relational plot with Fare on x-axis and Survived on y-axis with hue as Sex is plotted. It allows us to visualise how variables on a dataset relate to each other.

5. Bar Plot

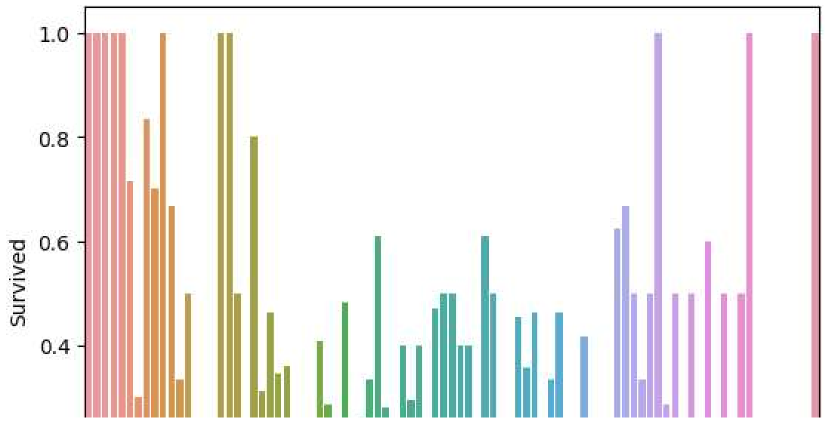
Inference: Bar graph is plotted with not_distracted on x-axis and ins_losses on y-axis

```
sns.barplot(data=ds, x="Age",y="Survived",errorbar=None)
sns.figure(figsize(17,8))
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-88-d01b1be0db06> in <cell line: 2>()
      1 sns.barplot(data=ds, x="Age",y="Survived",errorbar=None)
----> 2 sns.figure(figsize(17,8))

AttributeError: module 'seaborn' has no attribute 'figure'
```

SEARCH STACK OVERFLOW

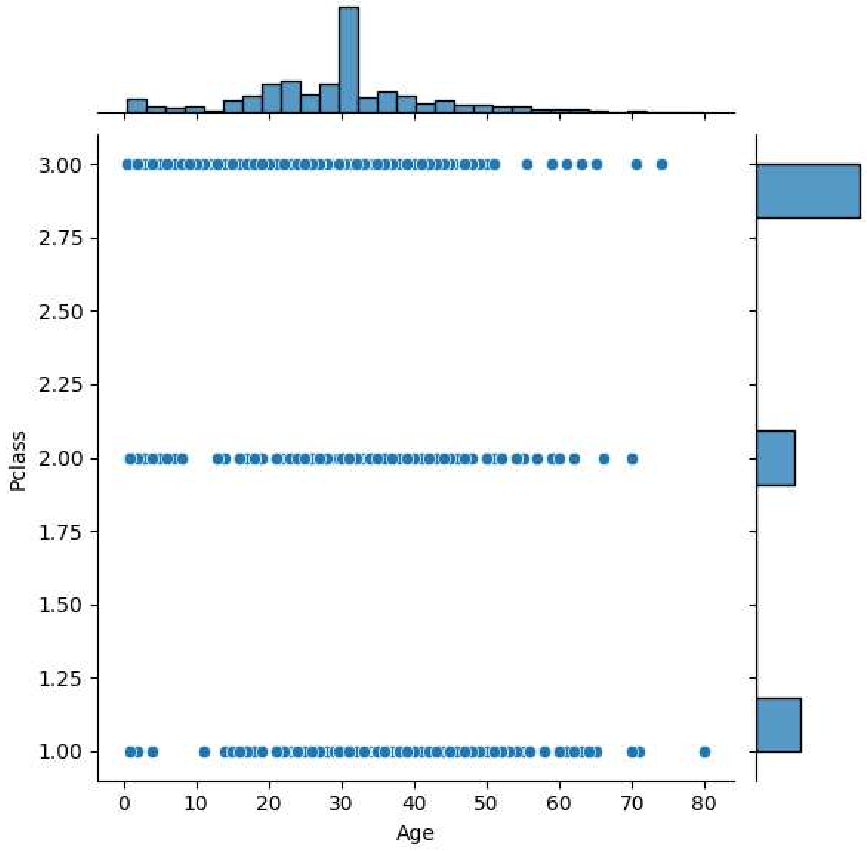


Inference: Bar graph is plotted with Age on x-axis and Survived on y-axis.

6. Joint Plot

```
sns.jointplot(x="Age", y="Pclass", data=ds)
```

```
<seaborn.axisgrid.JointGrid at 0x7895839eb400>
```

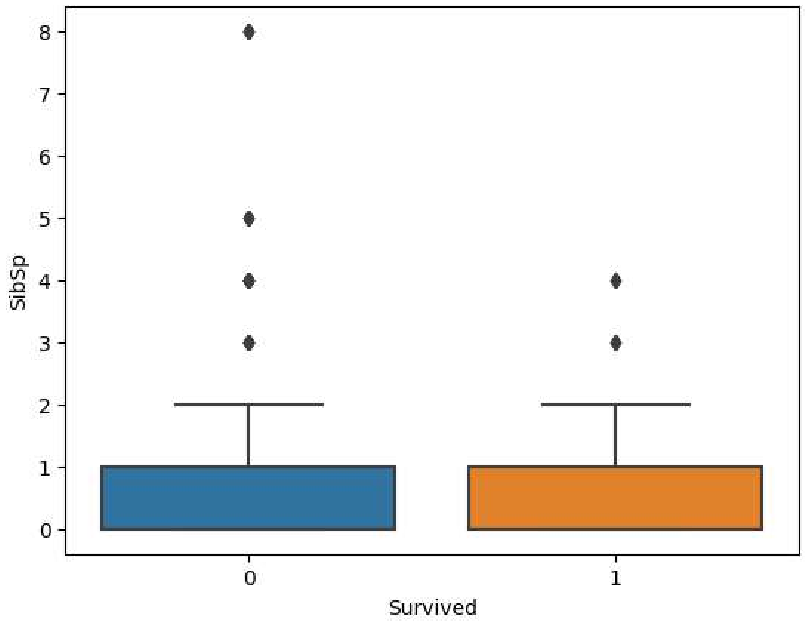


Inference: Joint plot with Age on x axis and Pclass on y-axis is plotted. This graph shows the Pclass preference of certain ages.

7. Box Plot

```
sns.boxplot(x="Survived",y="SibSp",data=ds)
```

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

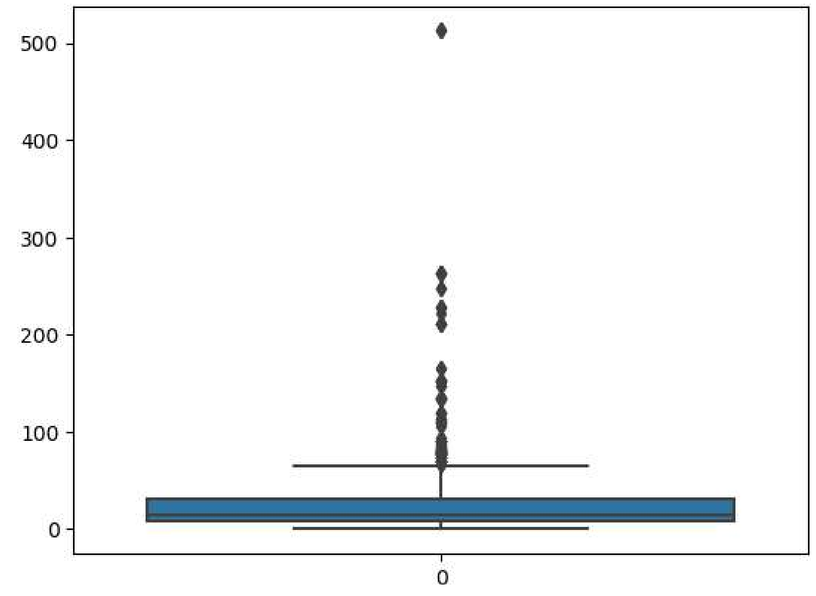


Inference: Box plot with Survived on x axis and SibSp on y-axis is plotted.

5. Outlier Detection

```
sns.boxplot(ds.Fare)
```

```
<Axes: >
```



Outlier replacement with median

```
qu1=ds.Fare.quantile(0.25)
qu3=ds.Fare.quantile(0.75)
print(qu1)
print(qu3)
```

```
7.9104
31.0
```

```
IQR=qu3-qu1
IQR
```

```
23.0896
```

```
up_limit=qu3+1.5*IQR
up_limit
```

```
65.6344
```

```
low_limit=qu1-1.5*IQR
IQR
```

```
23.0896
```

```
ds.Fare.median()
```

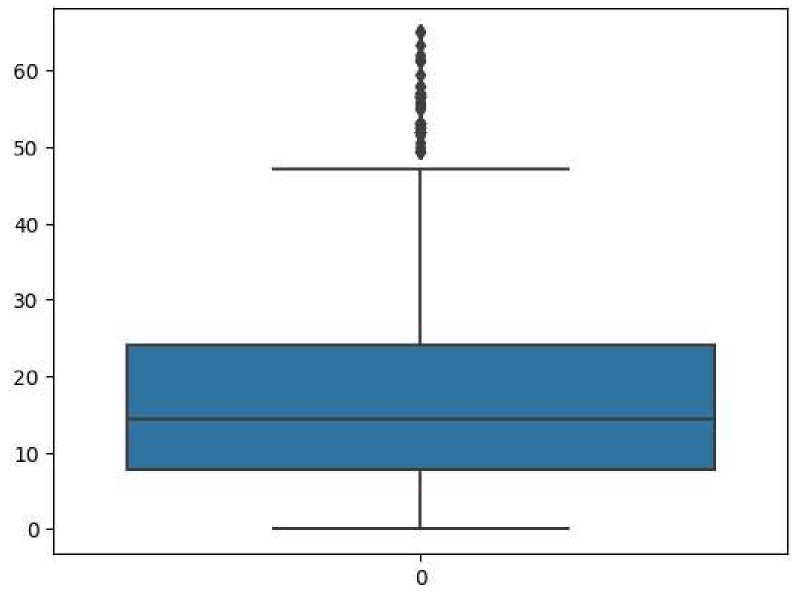
```
14.4542
```

```
ds['Fare']= np.where(ds['Fare']>up_limit,14.4542,ds['Fare'])
ds['Fare']
```

```
0       7.2500
1      14.4542
2       7.9250
3      53.1000
4       8.0500
        ...
886    13.0000
887    30.0000
888    23.4500
889    30.0000
890     7.7500
Name: Fare, Length: 891, dtype: float64
```

```
sns.boxplot(ds.Fare)
```

```
<Axes: >
```



```
ds.shape  #No loss of data
```

```
(891, 12)
```

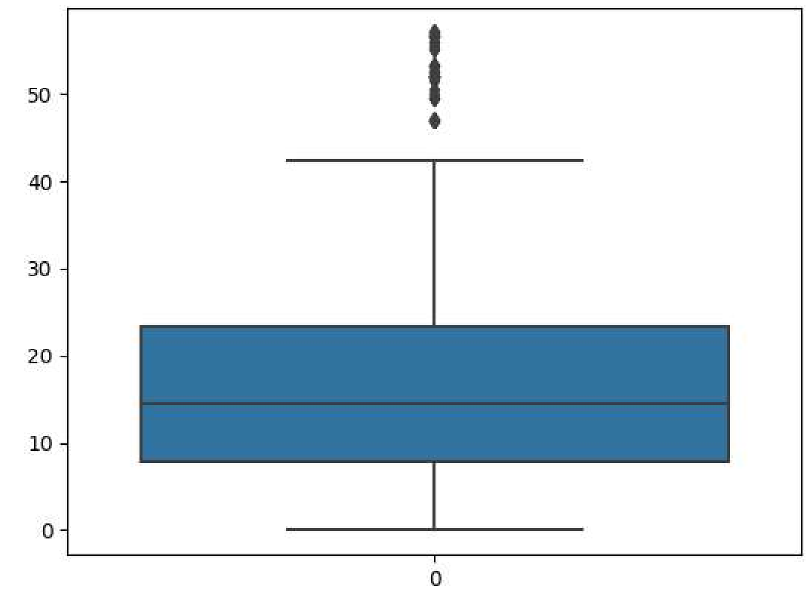
Percentile

```
per=ds.Fare.quantile(0.99)
per
```

```
    57.09792000000002
```

```
ds=ds[ds.Fare<=per]
sns.boxplot(ds.Fare)
```

```
    <Axes: >
```



6. Splitting Dependent and Independent variables

```
#Independent Variables
x=ds.drop(columns=['Survived'], inplace=False)
x
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.2500	B96 B98	S
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599	14.4542	C85	C
2	3	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282	7.9250	B96 B98	S
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803	53.1000	C123	S
			Allen, Mr								

```
#Dependent variable
y=ds.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

882 rows × 1 columns

```
ds.shape
```

```
    (882, 12)
```

```
x.shape
```

```
    (882, 11)
```

```
y.shape
```

```
    (882, 1)
```

7. Encoding

```
from sklearn.preprocessing import LabelEncoder
l=LabelEncoder()
x["Sex"]=l.fit_transform(x["Sex"])
x["Sex"]
```

```
0      1
1      0
2      0
3      0
4      1
      ..
886    1
887    0
888    0
889    1
890    1
Name: Sex, Length: 882, dtype: int64
```

```
x.head()
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	B96 B98	S
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	14.4542	C85	C
			Heikkinen					STON/O2		B96	

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

```
1    573
0    309
Name: Sex, dtype: int64
```

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

```
2
```

Label Encoding on Embarked, Cabin & Ticket

```
x["Embarked"]=l.fit_transform(x["Embarked"])
x["Embarked"]
```

```
0      2
1      0
2      2
3      2
4      2
      ..
886    2
887    2
888    2
889    0
890    1
Name: Embarked, Length: 882, dtype: int64
```

```
x["Embarked"].value_counts()
```

```
2    643
0    162
1     77
Name: Embarked, dtype: int64
```

```
x["Embarked"].nunique()
```

```
3
```

```
x.head()
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	B96 B98	2
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	14.4542	C85	0
			Heikkinen					STON/O2		B96	

```
x["Ticket"]=l.fit_transform(x["Ticket"])
x["Ticket"]
```

```
0      520
1      592
2      663
3       47
4      469
      ...
886     99
887     13
888    669
889      7
890    463
Name: Ticket, Length: 882, dtype: int64
```

```
x["Cabin"]=l.fit_transform(x["Cabin"])
x["Cabin"]
```

```
0      45
1      79
2      45
3      53
4      45
       ..
886    45
887    28
888    45
889    58
890    45
Name: Cabin, Length: 882, dtype: int64
```

```
x["Name"]=l.fit_transform(x["Name"])
x["Name"]
```

```
0      108
1      189
2      351
3      271
4       15
      ...
886    542
887    302
888    407
889     81
890    219
Name: Name, Length: 882, dtype: int64
```

```
x
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	3	108	1	22.000000	1	0	520	7.2500	45	2
1	2	1	189	0	38.000000	1	0	592	14.4542	79	0
2	3	3	351	0	26.000000	0	0	663	7.9250	45	2
3	4	1	271	0	35.000000	1	0	47	53.1000	53	2
4	5	3	15	1	35.000000	0	0	469	8.0500	45	2
...	...	...	...	...	...	...	...	...	...	...	...
886	887	2	542	1	27.000000	0	0	99	13.0000	45	2
887	888	1	302	0	19.000000	0	0	13	30.0000	28	2
888	889	3	407	0	29.699118	1	2	669	23.4500	45	2
889	890	1	81	1	26.000000	0	0	7	30.0000	58	0
890	891	3	219	1	32.000000	0	0	463	7.7500	45	1

882 rows × 11 columns

Correlation after encoding

```
corr=ds.corr()
corr
```

```
<ipython-input-61-5fb269adebca>:1: FutureWarning: The default value of numeric_only in DataFrame.corr i
  corr=ds.corr()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.009548	-0.044338	0.038705	-0.060494	-0.002870	0.016764
Survived	-0.009548	1.000000	-0.338136	-0.069492	-0.035868	0.075182	0.183125
Pclass	-0.044338	-0.338136	1.000000	-0.325355	0.081677	0.025713	-0.419194
Age	0.038705	-0.069492	-0.325355	1.000000	-0.235436	-0.186432	0.032520
SibSp	-0.060494	-0.035868	0.081677	-0.235436	1.000000	0.418549	0.283286
Parch	-0.002870	0.075182	0.025713	-0.186432	0.418549	1.000000	0.284996
Fare	0.016764	0.183125	-0.419194	0.032520	0.283286	0.284996	1.000000

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

```
<Axes: >
```

8. Feature Scaling

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

```
array([[-1.6948399 ,  0.85675649,  1.09700401, ..., -0.76182423,
        -0.29079717,  0.59293918],
       [-1.113136  , -0.33921354,  1.55529655, ..., -0.1248214 ,
        -0.29079717,  0.59293918],
       [-1.23416164,  0.85675649, -0.83326203, ..., -0.77458083,
        -0.29079717, -1.93011926],
       ...,
       [ 0.74519324, -0.33921354, -1.03522145, ...,  0.74381884,
        -0.29079717,  0.59293918],
       [ 0.46800547,  0.85675649,  1.31838261, ..., -0.77458083,
        -0.29079717,  0.59293918],
       [ 0.95991616,  0.85675649,  0.02894936, ...,  3.24598252,
        -0.29079717,  0.59293918]])
```

9. 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,y,test_size=0.3,random_state=0)
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

```
((617, 11), (265, 11), (617, 1), (265, 1))
```