#Import Libraries

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

from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

#Importing the Dataset

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

     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           ...       ...     ...
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         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
..                                                 ...     ...   ...
...
886                              Montvila, Rev. Juozas    male  27.0
0
887                       Graham, Miss. Margaret Edith  female  19.0
0
888           Johnston, Miss. Catherine Helen "Carrie"  female   NaN
1
889                              Behr, Mr. Karl Howell    male  26.0
```

```
0
890                                Dooley, Mr. Patrick    male  32.0
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
..     ...               ...      ...   ...      ...
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]
```

#Checking for Null values

```
dataset.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

dataset.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
```

we have few null values as we can see

```
dataset = dataset.drop(columns=['Cabin'])

dataset = dataset.fillna(dataset["Age"].mean())

dataset = dataset.fillna(dataset["Embarked"].mode())

dataset.isnull().any()
```

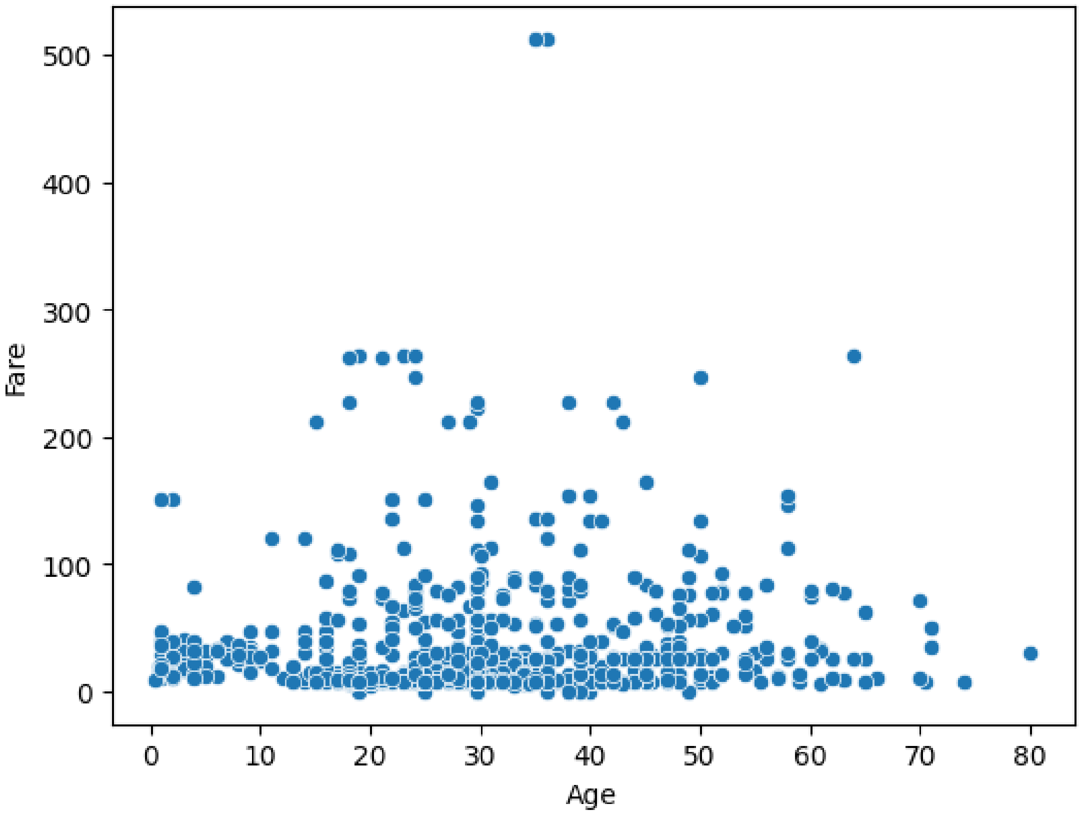
```
PassengerId    False
Survived       False
Pclass         False
Name           False
Sex            False
Age            False
SibSp          False
Parch          False
Ticket         False
Fare           False
Embarked       False
dtype: bool
```

we have removed all the null values

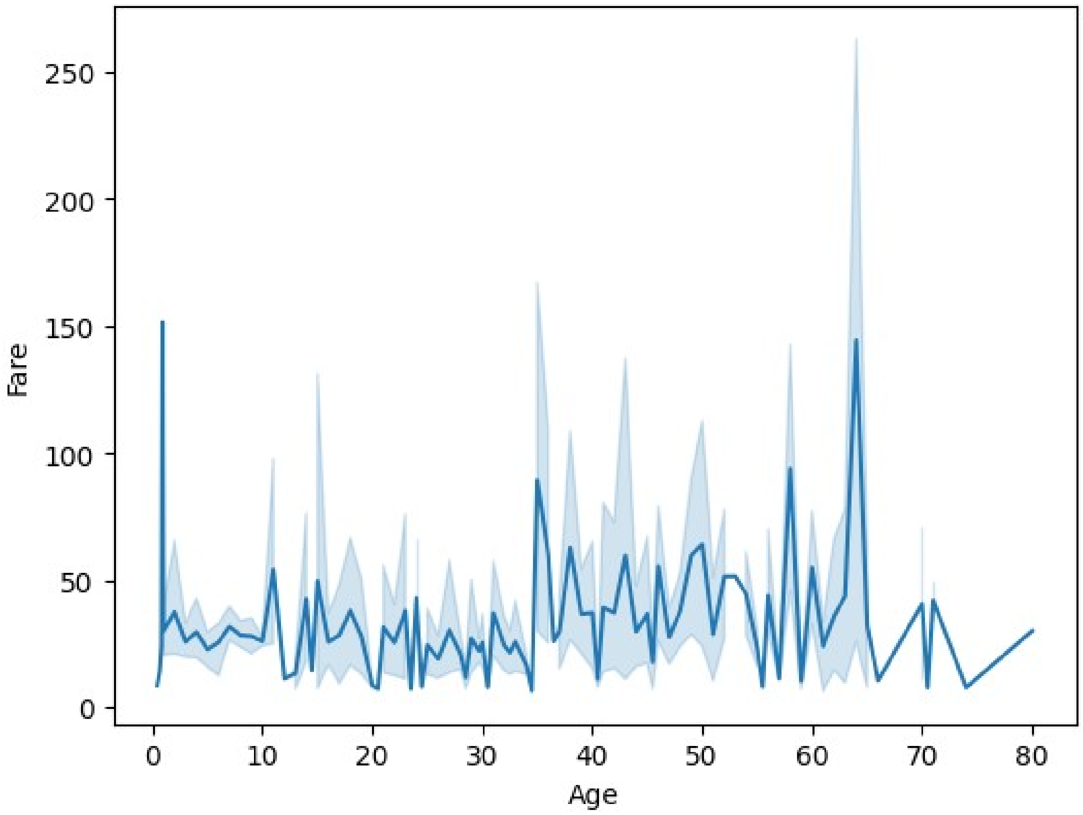
#DataVisualization

```
sns.scatterplot(data=dataset, x='Age', y='Fare')
#Shows the relationship between Age and Fare, which can help
#identify any patterns or trends in the data.
```

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

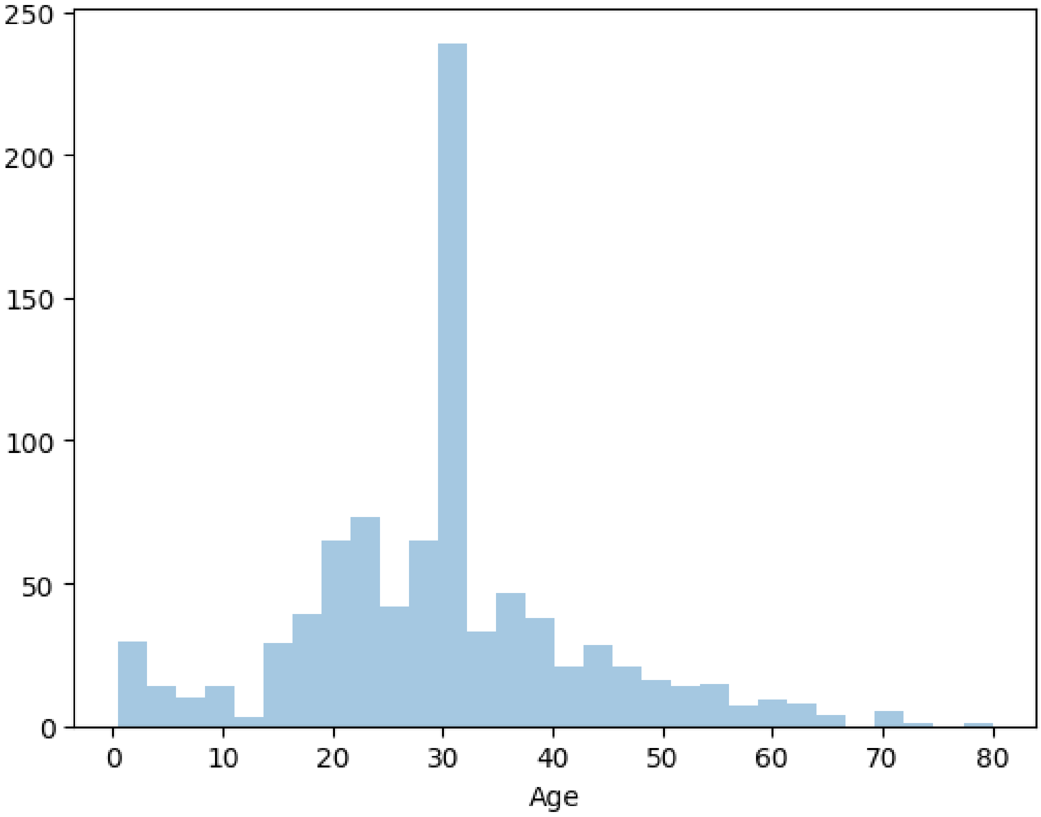


```
sns.lineplot(data=dataset, x='Age', y='Fare')
#Displays trends in Age and Fare over a continuous range of values.

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



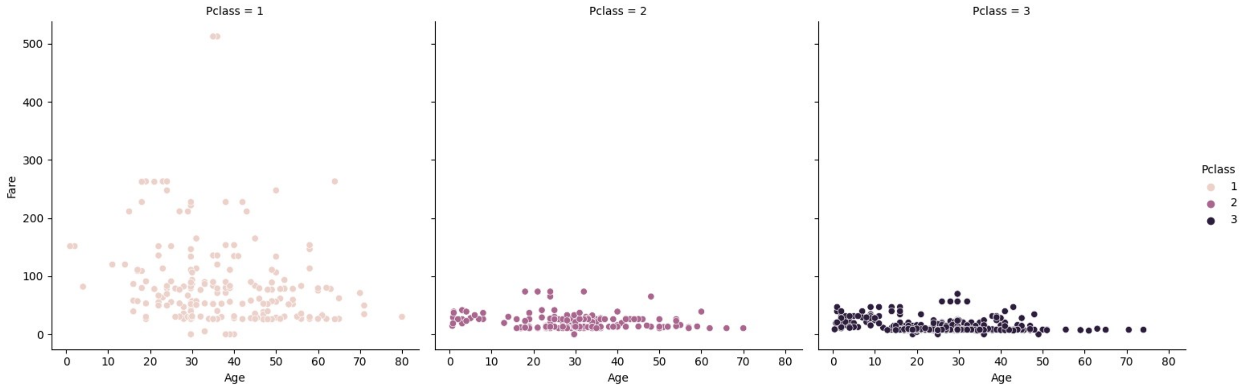
```
sns.distplot(dataset['Age'].dropna(), kde=False)
#Illustrates the distribution of passenger ages, giving insight
#into the age distribution on the Titanic.
```

```
<ipython-input-15-6dc85876cec3>: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(dataset['Age'].dropna(), kde=False)

<Axes: xlabel='Age'>
```



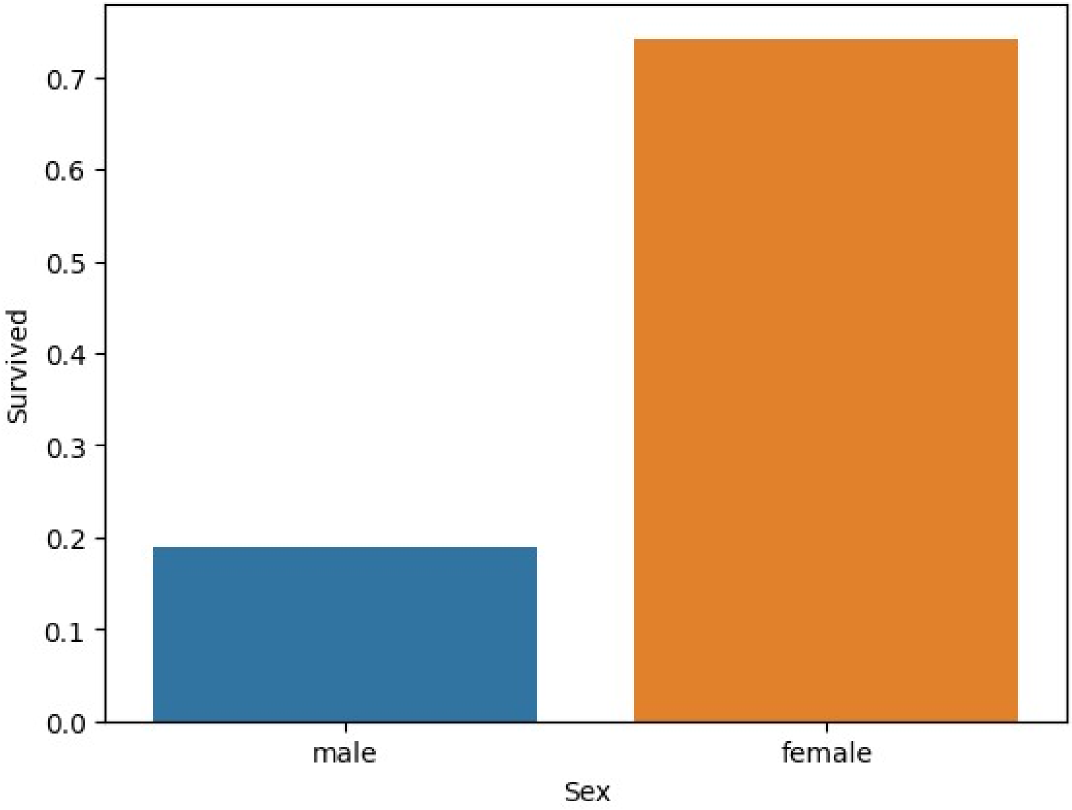
```
sns.relplot(data=dataset, x='Age', y='Fare', hue='Pclass',
col='Pclass')
#Examines the relationship between Age and Fare, with separate plots
#for each passenger class.
```

```
<seaborn.axisgrid.FacetGrid at 0x783e9bd5d9c0>
```

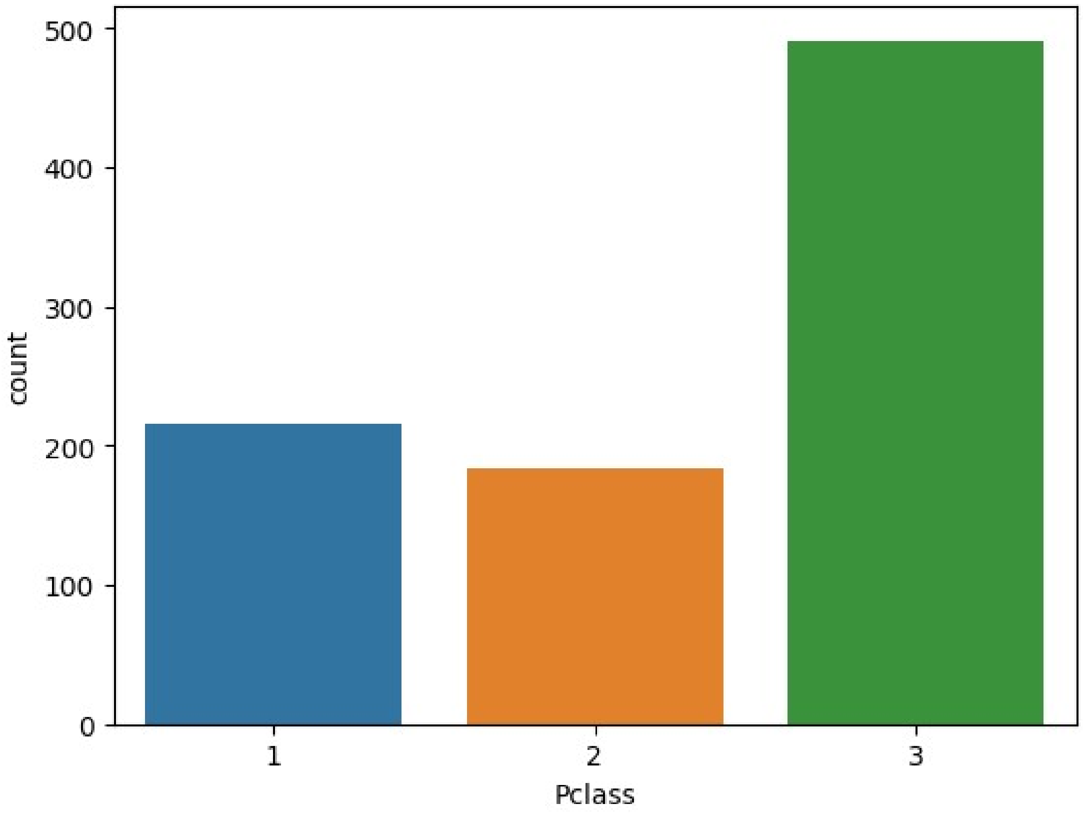


```
sns.barplot(data=dataset, x='Sex', y='Survived', ci=None)
#Displays the survival rate by gender, helping to compare the survival
#rates of male and female passengers.
```

```
<ipython-input-17-11e6d7ce1e6c>:1: FutureWarning:

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

  sns.barplot(data=dataset, x='Sex', y='Survived', ci=None)

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

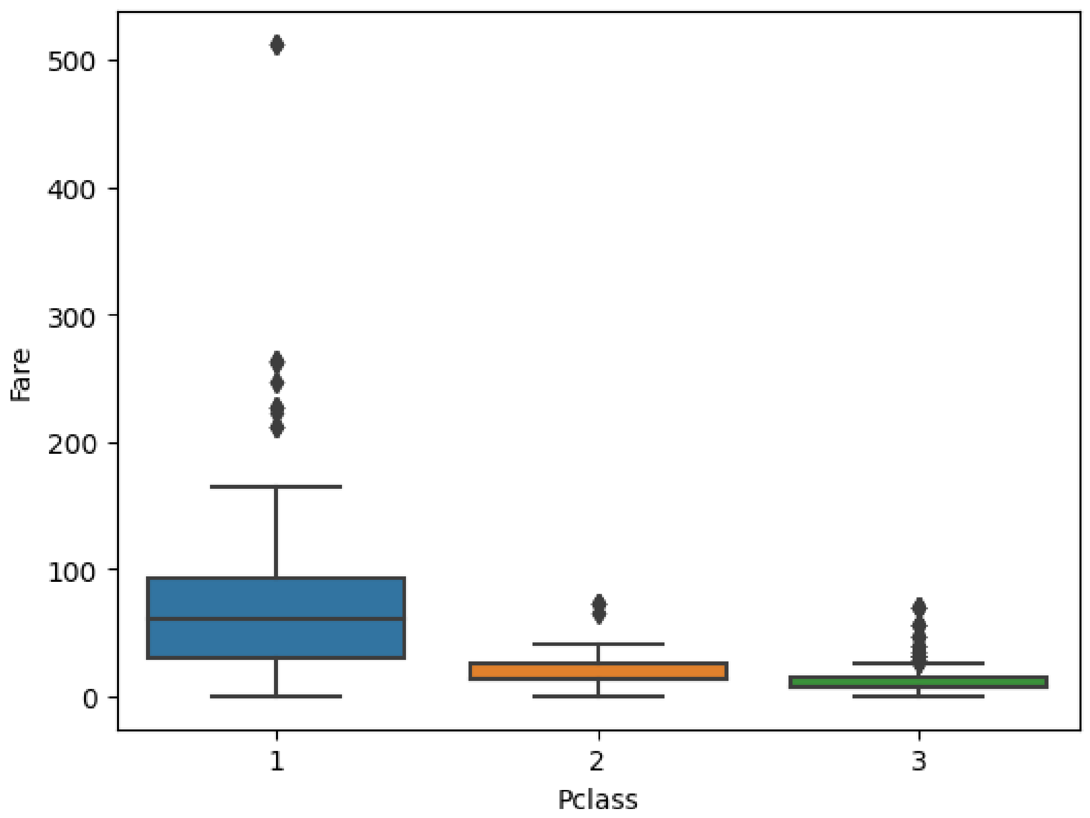


```
sns.countplot(data=dataset, x='Pclass')
#Shows the count of passengers in each passenger class, providing
#a simple visualization of class distribution.

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



```
sns.boxplot(data=dataset, x='Pclass', y='Fare')
#Highlights the distribution of fares for each passenger class,
#including information about outliers and quartiles.
```

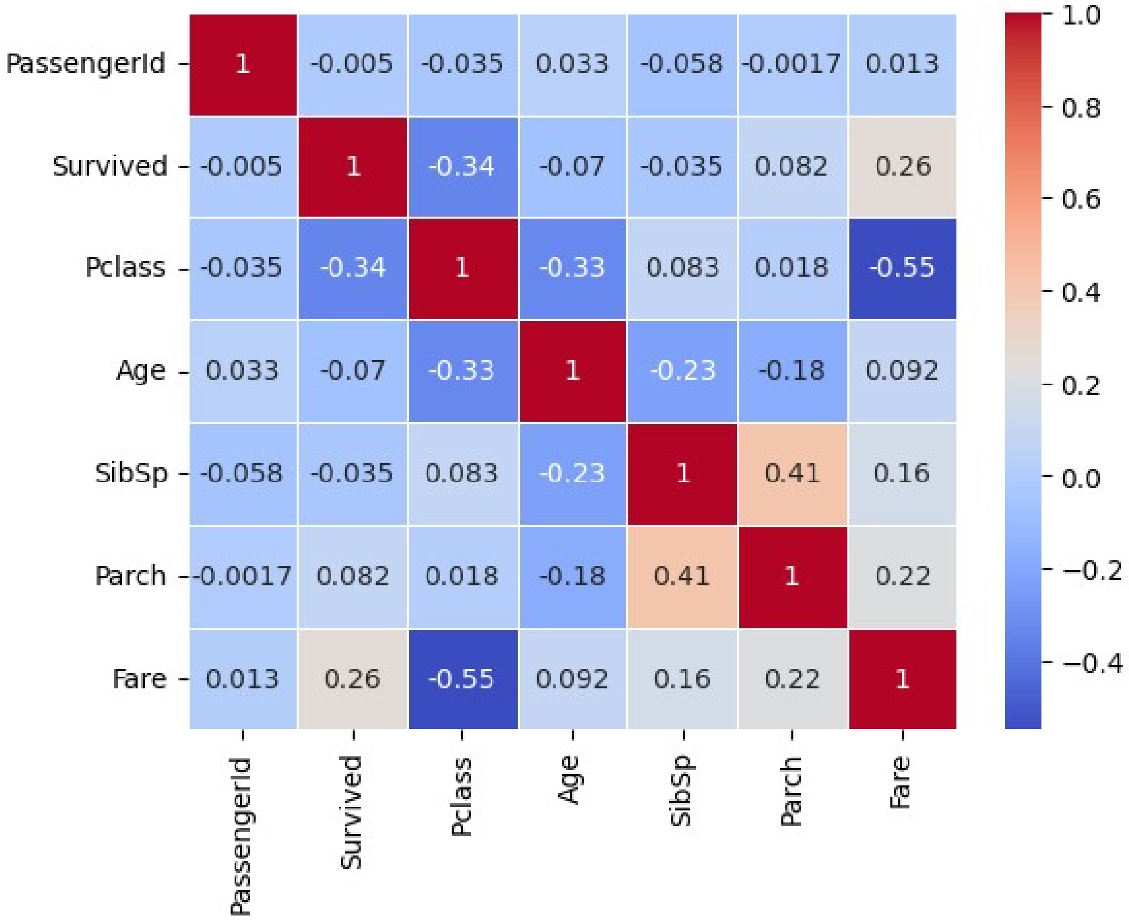
```
<Axes: xlabel='Pclass', ylabel='Fare'>
```



```
corr_matrix = dataset.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
#Illustrates the correlation matrix of numerical features, allowing
#you to see how variables are related to each other.
```

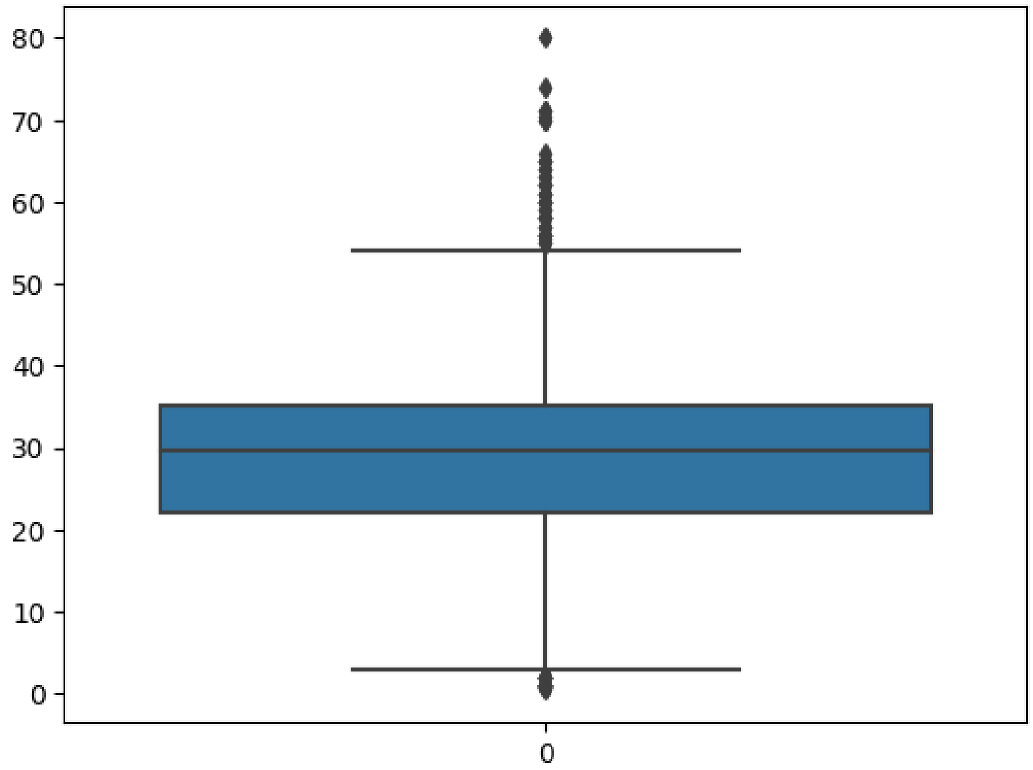
```
<ipython-input-20-b9e496d0e847>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  corr_matrix = dataset.corr()
```

```
<Axes: >
```



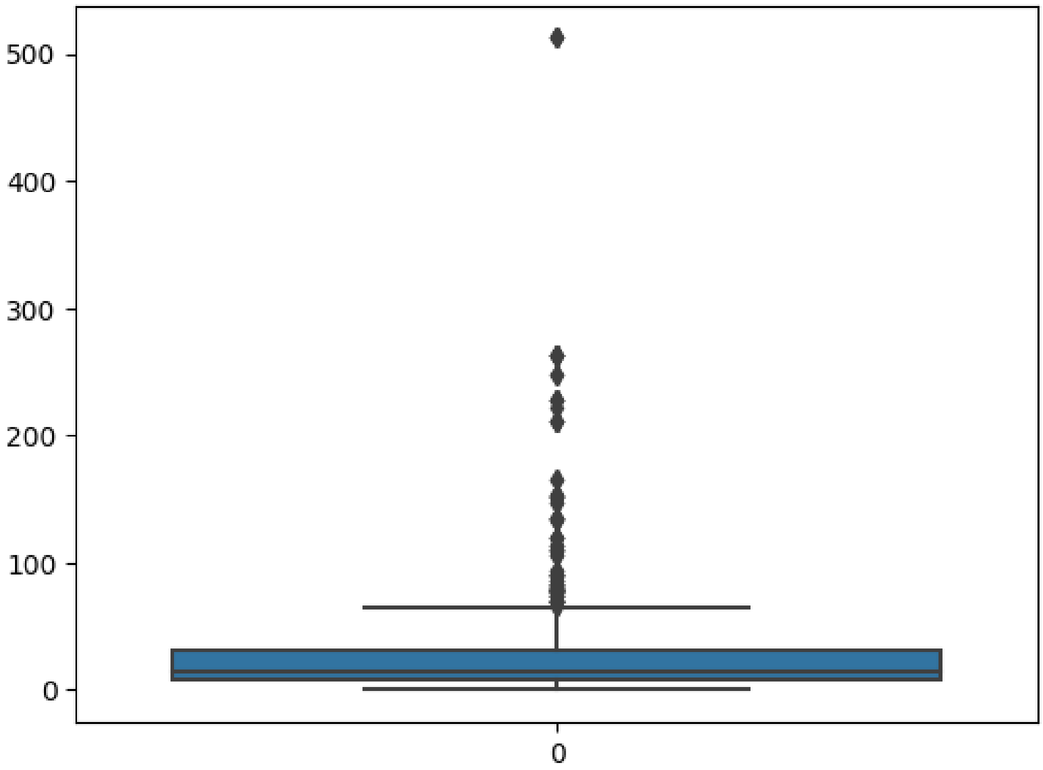
#Outlier Detection

```
#outlier detection on Age and also on Fare

sns.boxplot(dataset['Age'])

<Axes: >
```



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

```
<Axes: >
```



We can visually se that there are outliers in both Age and Fare. Any data points outside of the whiskers are considered outliers.

#Splitting Dependent and Independent Variables

```
dataset.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   7.2500        S
1      0          PC 17599  71.2833        C
2      0  STON/O2. 3101282   7.9250        S
3      0            113803  53.1000        S
4      0            373450   8.0500        S

x = dataset.iloc[:, 2:] #Independent Variable
y = dataset.iloc[:, 1:2]  #Dependent Variable - Survived
```

#Encoding

```
dt = pd.get_dummies(dataset, columns=['Embarked'], prefix='Embarked')

dt

     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           ...       ...     ...
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                  Name     Sex
Age  \
0                              Braund, Mr. Owen Harris    male
22.000000
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female
38.000000
2                               Heikkinen, Miss. Laina  female
26.000000
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female
35.000000
4                             Allen, Mr. William Henry    male
35.000000
..                                                 ...     ...
...
886                              Montvila, Rev. Juozas    male
27.000000
887                       Graham, Miss. Margaret Edith  female
19.000000
888           Johnston, Miss. Catherine Helen "Carrie"  female
29.699118
```

```
889                              Behr, Mr. Karl Howell    male
26.000000
890                                Dooley, Mr. Patrick    male
32.000000

     SibSp  Parch            Ticket     Fare
Embarked_29.69911764705882  \
0        1      0         A/5 21171   7.2500
0
1        1      0          PC 17599  71.2833
0
2        0      0  STON/O2. 3101282   7.9250
0
3        1      0            113803  53.1000
0
4        0      0            373450   8.0500
0
..     ...    ...               ...      ...                         .
..
886      0      0            211536  13.0000
0
887      0      0            112053  30.0000
0
888      1      2        W./C. 6607  23.4500
0
889      0      0            111369  30.0000
0
890      0      0            370376   7.7500
0

     Embarked_C  Embarked_Q  Embarked_S
0             0           0           1
1             1           0           0
2             0           0           1
3             0           0           1
4             0           0           1
..          ...         ...         ...
886           0           0           1
887           0           0           1
888           0           0           1
889           1           0           0
890           0           1           0

[891 rows x 14 columns]
```

```
labels = [row[0] for row in dt]
encoded_labels = le.fit_transform(labels)
unique_classes = list(le.classes_)
print(unique_classes)
```

```
['A', 'E', 'F', 'N', 'P', 'S', 'T']

mapping=dict(zip(le.classes_,range(len(le.classes_))))
mapping

{'A': 0, 'E': 1, 'F': 2, 'N': 3, 'P': 4, 'S': 5, 'T': 6}
```

#Feature Scaling

```
from sklearn.preprocessing import MinMaxScaler
ms=MinMaxScaler()

numeric_columns = x.select_dtypes(include=['number'])
X_scaled = ms.fit_transform(numeric_columns)

X_Scaled=pd.DataFrame(ms.fit_transform(numeric_columns),columns=numeri
c_columns.columns)

X_Scaled.head()

   Pclass       Age  SibSp  Parch      Fare
0     1.0  0.271174  0.125    0.0  0.014151
1     0.0  0.472229  0.125    0.0  0.139136
2     1.0  0.321438  0.000    0.0  0.015469
3     0.0  0.434531  0.125    0.0  0.103644
4     1.0  0.434531  0.000    0.0  0.015713
```

#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_Scaled,y,test_size
=0.2,random_state =0)

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

(712, 5) (179, 5) (712, 1) (179, 1)

from sklearn.linear_model import LinearRegression

lr=LinearRegression()

lr.fit(x_train,y_train)

LinearRegression()

y_pred=lr.predict(x_test)

y_pred

array([[ 0.24070021],
       [ 0.23634822],
       [ 0.30254507],
       [ 0.68118897],
       [ 0.34283675],
       [ 0.65405547],
       [ 0.64597137],
       [ 0.82373277],
       [ 0.68528767],
       [ 0.23614613],
       [ 0.22130452],
       [ 0.35586011],
       [ 0.23647421],
       [ 0.73409818],
       [ 0.73455809],
       [ 0.36691703],
       [ 0.29303788],
       [ 0.54718308],
       [ 0.19264073],
       [ 0.52803004],
       [ 0.53761955],
       [ 0.79425197],
       [ 0.23646634],
       [ 0.65437046],
       [ 0.30172848],
       [ 0.63598731],
       [ 0.21745782],
       [ 0.30041342],
       [ 0.53653805],
       [ 0.43716988],
       [ 0.20963607],
       [ 0.27123774],
       [ 0.25168739],
       [ 0.6572053 ],
       [ 0.19588102],
       [ 0.63494723],
       [ 0.12519482],
       [ 0.45394213],
       [ 0.48043128],
       [ 0.24022364],
       [ 0.43505534],
       [ 0.27516846],
       [ 0.24072648],
       [ 0.25106586],
       [ 0.59744984],
       [ 0.23656606],
       [ 0.23656606],
       [ 0.90121016],
       [ 0.36416485],
       [ 0.50177678],
       [ 0.65374049],
       [ 0.77830675],
       [ 0.66309224],
       [ 0.32541951],
       [ 0.6535515 ],
       [ 0.43493779],
       [ 0.38331652],
       [ 0.31859774],
       [ 0.37073192],
       [ 0.15432755],
       [ 0.30008534],
       [ 0.44234339],
       [ 0.43637864],
       [ 0.65978816],
       [ 0.35685551],
       [ 0.21736597],
       [ 0.49088712],
       [ 0.51603357],
       [ 0.68159162],
       [ 0.74259394],
       [ 0.45097001],
       [ 0.46966   ],
       [ 0.65476145],
       [ 0.2366632 ],
       [ 0.27984055],
       [ 0.30608059],
       [ 0.84734358],
       [ 0.56675847],
       [ 0.19996638],
       [ 0.44269774],
       [ 0.20085008],
       [ 0.41547099],
       [ 0.48227774],
       [ 0.35025764],
       [ 0.31422118],
       [ 0.7561965 ],
       [ 0.67601867],
       [-0.03892356],
       [ 0.40411366],
       [ 0.75464737],
       [ 0.63484154],
       [ 0.32541951],
       [ 0.67328688],
       [ 0.60051845],
       [ 0.37903666],
       [ 0.23614613],
       [ 0.58551542],
       [ 0.23445637],
       [ 0.39992601],
       [ 0.70559374],
       [ 0.25757115],
       [ 0.20182766],
       [ 0.32526466],
       [ 0.33565971],
       [ 0.24595065],
       [ 0.20228796],
       [ 0.45529656],
       [ 0.4023289 ],
       [ 0.44266526],
       [ 0.19540088],
       [ 0.0845557 ],
       [ 0.78134605],
       [ 0.24807413],
       [ 0.2366632 ],
       [ 0.70798902],
       [ 0.45551705],
       [ 0.24295109],
       [ 0.82937699],
       [ 0.15101986],
       [ 0.32464508],
       [ 0.26718237],
       [ 0.23692042],
       [ 0.31759776],
       [ 0.56590482],
       [ 0.20963084],
       [ 0.41851989],
       [ 0.23614348],
       [ 0.22955937],
       [ 0.43321676],
       [ 0.26420868],
       [ 0.30871817],
       [ 0.30890716],
       [ 0.27505821],
       [ 0.27237453],
       [ 0.36636055],
       [ 0.30890716],
       [ 0.19051865],
       [ 0.10605914],
       [ 0.28487511],
       [ 0.49704077],
       [ 0.45551705],
       [ 0.61816985],
       [ 0.2366632 ],
       [ 0.33300244],
       [ 0.45400561],
       [ 0.2844969 ],
       [ 0.43890756],
       [ 0.3014846 ],
       [ 0.67009459],
       [ 0.23704906],
       [ 0.51440239],
       [ 0.23695205],
       [ 0.17699981],
       [ 0.24229596],
       [ 0.54623928],
       [ 0.47055162],
       [ 0.16154492],
       [ 0.1715808 ],
       [ 0.23647421],
       [ 0.44014518],
       [ 0.23656606],
       [ 0.31669483],
       [ 0.42449327],
       [ 0.07629725],
       [ 0.36018021],
       [ 0.27543619],
       [ 0.05913749],
       [ 0.30044228],
       [ 0.28389578],
       [ 0.34251199],
       [ 0.05976481],
       [ 0.56485526],
       [ 0.23615923],
       [ 0.24072648],
       [ 0.37482804],
       [ 0.2366632 ],
       [ 0.64944178],
       [ 0.29217432],
       [ 0.32551665]])
```

```
y_test
```

```
     Survived
495         0
648         0
278         0
31          1
255         1
..        ...
780         1
837         0
215         1
833         0
372         0

[179 rows x 1 columns]
```

```
dataset.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   7.2500        S
1      0          PC 17599  71.2833        C
2      0  STON/O2. 3101282   7.9250        S
3      0            113803  53.1000        S
4      0            373450   8.0500        S
```

```python
from sklearn import metrics

# R- Square
# evaluating testing accuracy
print(metrics.r2_score(y_test,y_pred))
```

```
0.19578250502082062
```

```python
print(metrics.mean_squared_error(y_test,y_pred))
```

```
0.1905062509563363
```

```python
# RMSE (Root Mean Square Error)
print(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
0.43647021771976185
```