Data Preprocessing

1. Importing necessary libraries

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

2. Importing the dataset

```
dataset = pd.read_csv("TitanicDataset.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]
```

```
dataset.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
```

```
dataset.describe()
```

```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
```

```
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000
```

```
            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

```
dataset.shape
```

```
(891, 12)
```

3. 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
```

There are 177 null values in age and 687 null values in cabin

- as number of null values are more than 50% we cannot delete data
- instead we try imputing via either MEAN/ MEDIAN/ MODE

```
# Handling null values in Age column (numerical) ---> MEAN
dataset["Age"].fillna(dataset["Age"].mean(), inplace = True)

# Handling null values in Cabin column (Categorical) ---> MODE
dataset["Cabin"].fillna(dataset["Cabin"].mode()[0], inplace = True)

# Handling null values in Embarked column (Categorical) ---> MODE
dataset["Embarked"].fillna(dataset["Embarked"].mode()[0], inplace =
True)

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    Cabin Embarked
0      0         A/5 21171   7.2500  B96 B98        S
1      0          PC 17599  71.2833      C85        C
2      0  STON/O2. 3101282   7.9250  B96 B98        S
3      0            113803  53.1000     C123        S
4      0            373450   8.0500  B96 B98        S
```

```
dataset.isnull().sum()
```

```
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
```

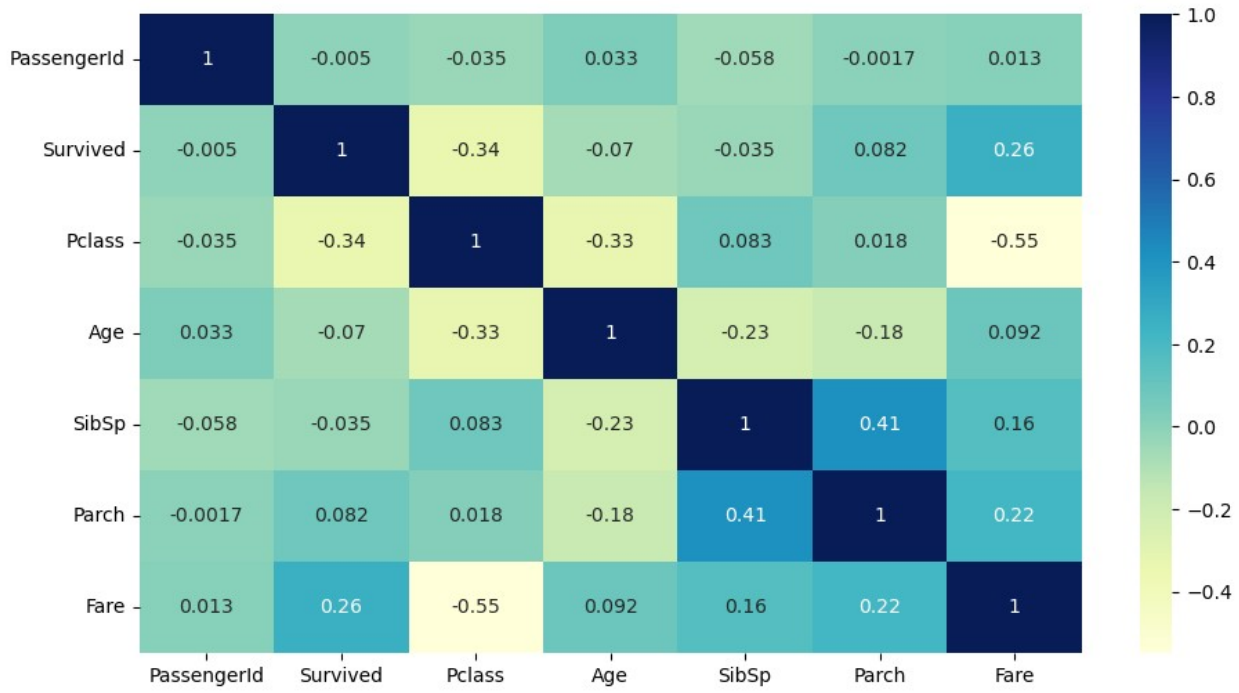
4. Data visualisation

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

```
             PassengerId  Survived    Pclass       Age     SibSp
Parch  \
PassengerId     1.000000 -0.005007 -0.035144  0.033207 -0.057527 -
0.001652
Survived       -0.005007  1.000000 -0.338481 -0.069809 -0.035322
0.081629
Pclass         -0.035144 -0.338481  1.000000 -0.331339  0.083081
0.018443
Age             0.033207 -0.069809 -0.331339  1.000000 -0.232625 -
0.179191
SibSp          -0.057527 -0.035322  0.083081 -0.232625  1.000000
0.414838
Parch          -0.001652  0.081629  0.018443 -0.179191  0.414838
1.000000
Fare            0.012658  0.257307 -0.549500  0.091566  0.159651
0.216225

                 Fare
PassengerId  0.012658
Survived     0.257307
Pclass      -0.549500
Age          0.091566
SibSp        0.159651
Parch        0.216225
Fare         1.000000
```

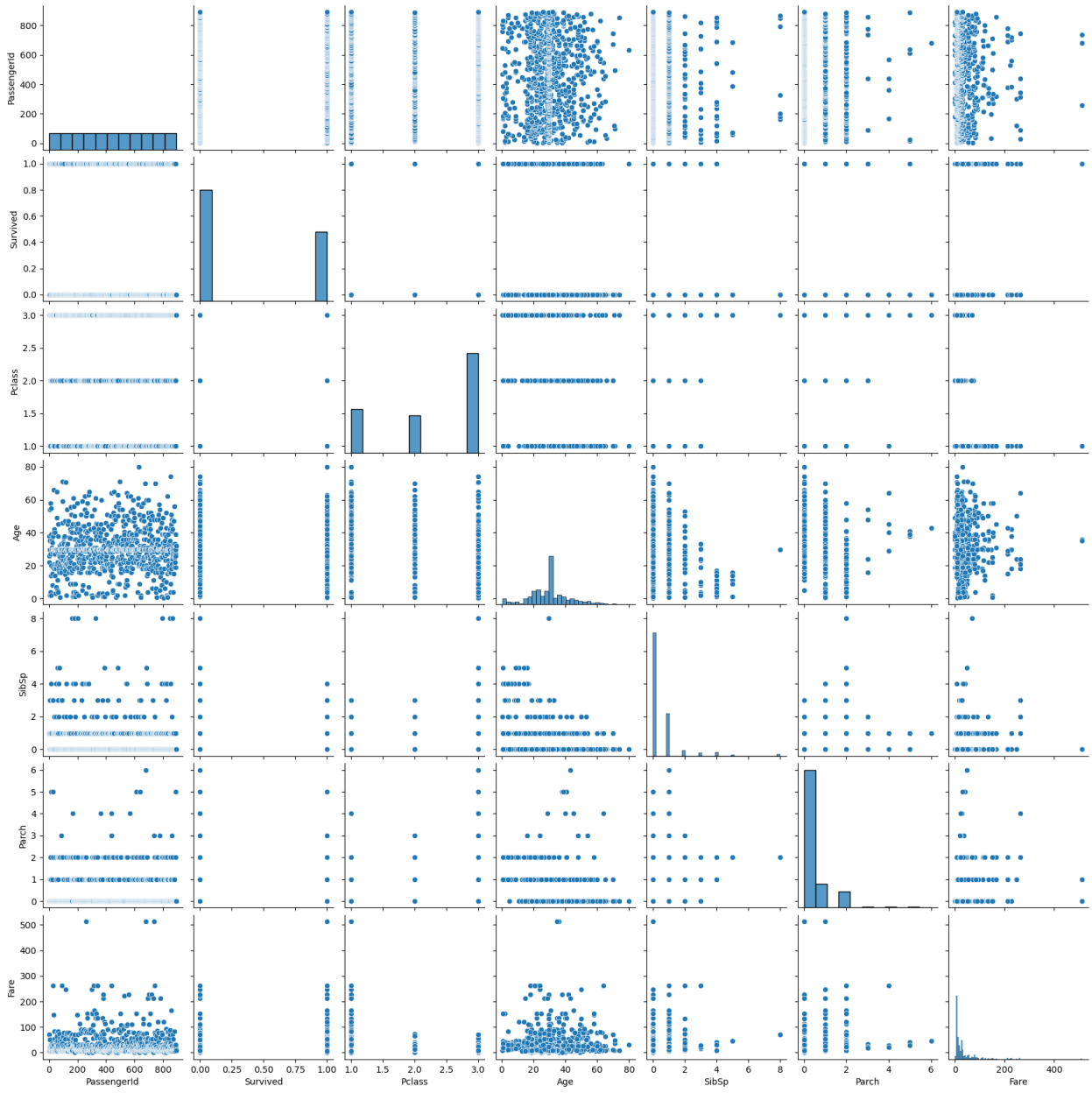
```
plt.subplots(figsize = [11,6])
sns.heatmap(corr, annot = True, cmap = "YlGnBu")
```

```
<Axes: >
```




```
sns.pairplot(dataset)
```

```
<seaborn.axisgrid.PairGrid at 0x2483ba8f5d0>
```



5. Detecting and managing Outliers

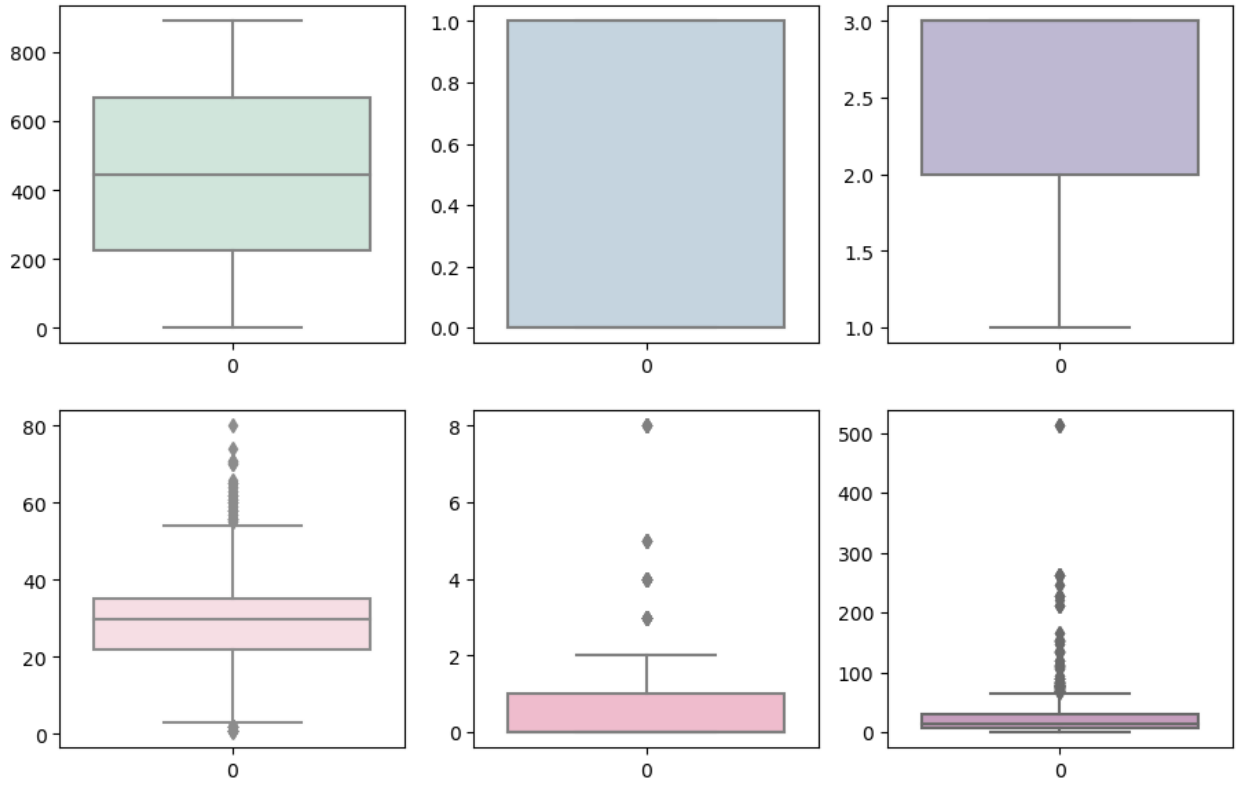
```
dataset.info()

<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          891 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        891 non-null    object
 11  Embarked     891 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
# Plotting for all numerical datatypes (Ignoring Parch as although its
a numerical datatype its a categorical data)
plt.subplots(figsize = [11,7])
plt.subplot(2,3,1)
sns.boxplot(dataset["PassengerId"], color = "#CCE8DB")
plt.subplot(2,3,2)
sns.boxplot(dataset["Survived"], color = "#C1D4E3")
plt.subplot(2,3,3)
sns.boxplot(dataset["Pclass"], color = "#BBB4D6")
plt.subplot(2,3,4)
sns.boxplot(dataset["Age"], color = "#FADAE2")
plt.subplot(2,3,5)
sns.boxplot(dataset["SibSp"], color = "#F8B3CA")
plt.subplot(2,3,6)
sns.boxplot(dataset["Fare"], color = "#CC97C1")
```

```
C:\Users\raman\AppData\Local\Temp\ipykernel_17896\1552390636.py:3:
MatplotlibDeprecationWarning: Auto-removal of overlapping axes is
deprecated since 3.6 and will be removed two minor releases later;
explicitly call ax.remove() as needed.
  plt.subplot(2,3,1)
```

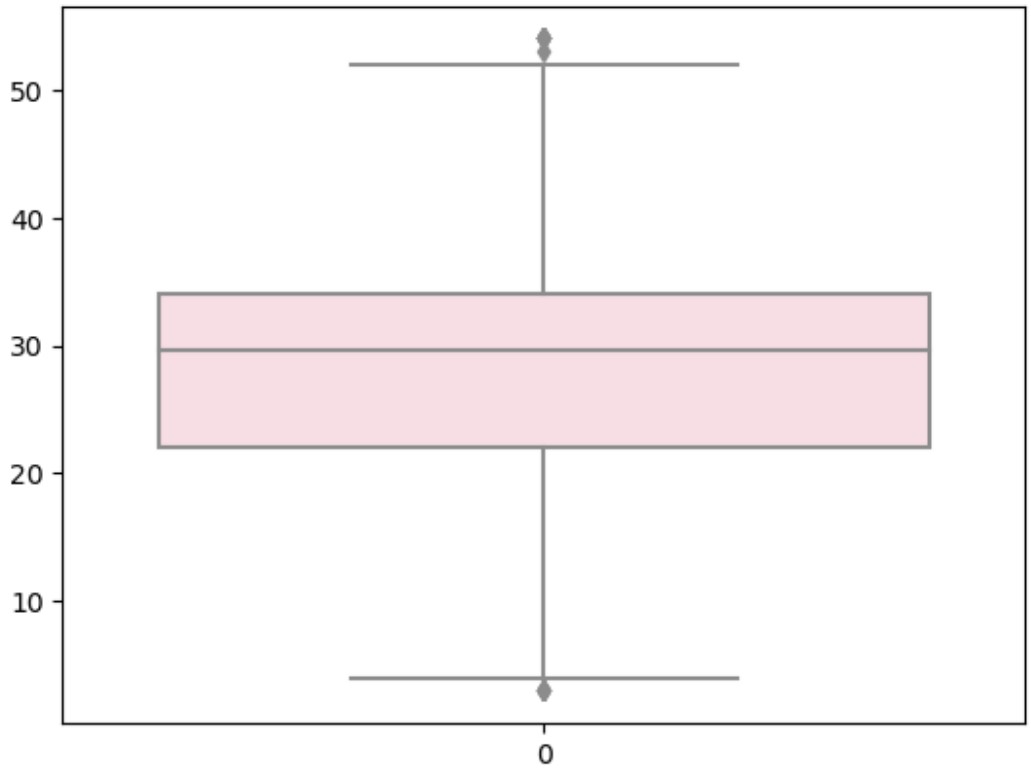
```
<Axes: >
```



Here there are many outliers in Age, SibSp and Fare

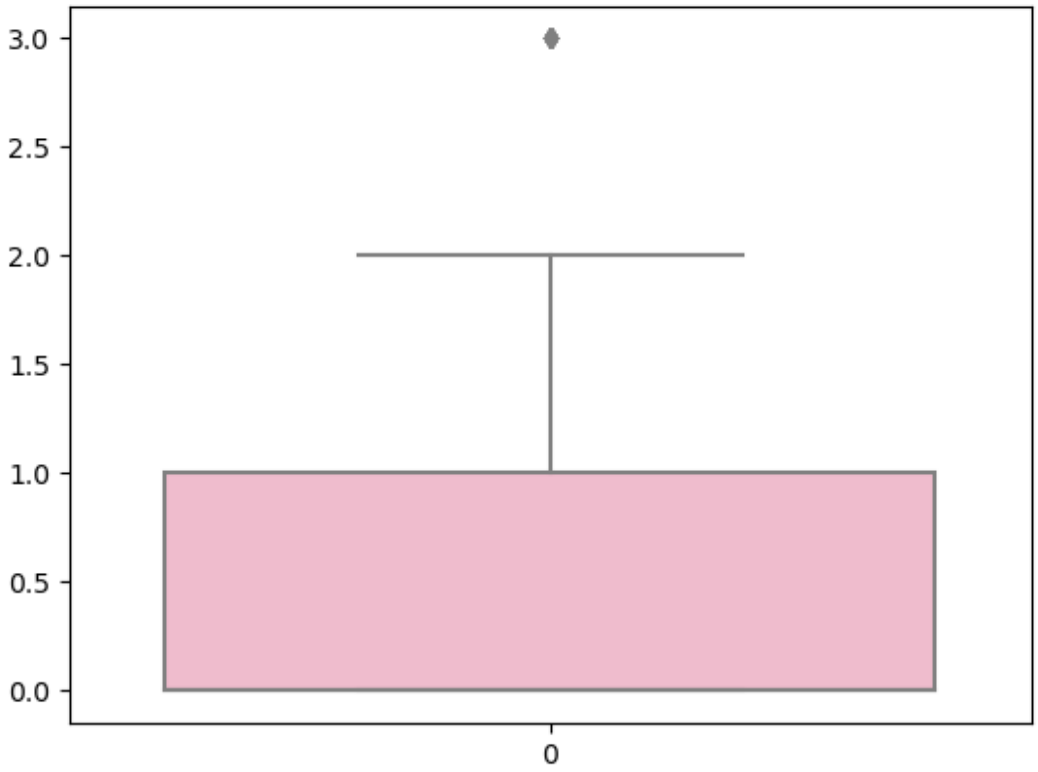
```
# Removing Outliers in Age by IQR method
q1 = dataset.Age.quantile(0.25)
q3 = dataset.Age.quantile(0.75)
IQR = q3 - q1
upper_limit = q3 + (1.5 * IQR)
lower_limit = q1 - (1.5 * IQR)
median = dataset.Age.median(numeric_only = True)

dataset = dataset[dataset.Age < upper_limit]
dataset = dataset[dataset.Age > lower_limit]

sns.boxplot(dataset.Age, color = "#FADAE2")
```

```
<Axes: >
```



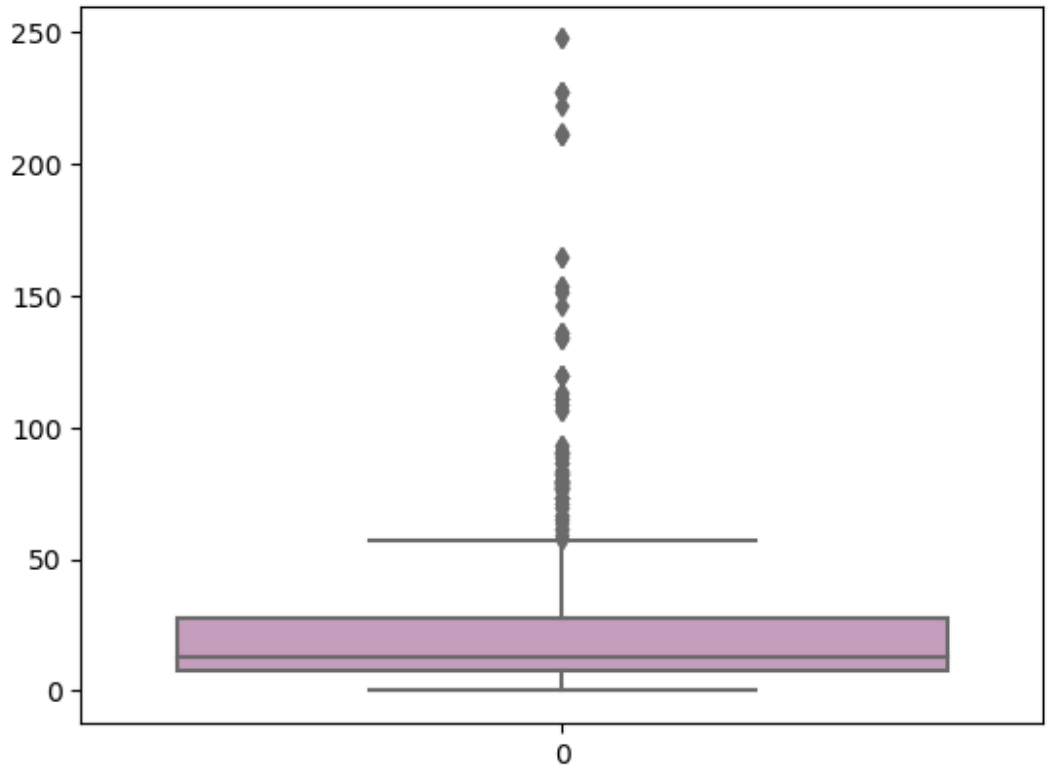

```
# Removing Outliers in SibSp by Zscore method
from scipy import stats

sibsp_zscore = stats.zscore(dataset.SibSp)

dataset = dataset[np.abs(sibsp_zscore) <= 3]

sns.boxplot(dataset["SibSp"], color = "#F8B3CA")
```

```
<Axes: >
```




```
# Removing Outliers in Fare by Percentile method
p99 = p99 = dataset.Fare.quantile(0.99)
dataset = dataset[dataset.Fare <= p99]

sns.boxplot(dataset["Fare"], color = "#CC97C1")
```

```
<Axes: >
```




```
dataset.shape
```

```
(792, 12)
```

```
dataset.describe()
```

```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   792.000000  792.000000  792.000000  792.000000  792.000000
mean    448.835859    0.383838    2.334596   29.339457    0.345960
std     255.804758    0.486627    0.821897    9.909121    0.606754
min       1.000000    0.000000    1.000000    3.000000    0.000000
25%     228.750000    0.000000    2.000000   23.000000    0.000000
50%     447.500000    0.000000    3.000000   29.699118    0.000000
75%     668.250000    1.000000    3.000000   34.000000    1.000000
max     891.000000    1.000000    3.000000   54.000000    3.000000

             Parch        Fare
count  792.000000  792.000000
mean     0.303030   27.796932
std      0.758661   36.826040
min      0.000000    0.000000
25%      0.000000    7.895800
50%      0.000000   13.000000
75%      0.000000   27.728100
max      6.000000  247.520800
```

6. Splitting dependant and independant variables

```
# Fare is the dependant variable as it depends on no.of people abroad
and the Pclass
x = dataset.drop(columns = ["Fare"])
y = dataset["Fare"]

x.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    Cabin Embarked
0      0         A/5 21171  B96 B98        S
1      0          PC 17599      C85        C
2      0  STON/O2. 3101282  B96 B98        S
3      0            113803     C123        S
4      0            373450  B96 B98        S
```

```
x.shape
```

```
(792, 11)
```

```
y.head()
```

```
0     7.2500
1    71.2833
2     7.9250
3    53.1000
4     8.0500
Name: Fare, dtype: float64
```

```
y.shape
```

```
(792,)
```

7. Encoding

```
x.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    Cabin Embarked
0      0         A/5 21171  B96 B98        S
1      0          PC 17599      C85        C
2      0  STON/O2. 3101282  B96 B98        S
3      0            113803     C123        S
4      0            373450  B96 B98        S
```

We have a total of 4 columns to encode ---> Name, Sex, Ticket, Cabin and Embarked

```
# Encoding Name, Ticket an Cabin using Label encoding as they have too
many distinct values
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
x["Name"] = le.fit_transform(x["Name"])
x["Ticket"] = le.fit_transform(x["Ticket"])
x["Cabin"] = le.fit_transform(x["Cabin"])

x["Name"].nunique()

792

x["Ticket"].nunique()

642

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

```
125
```

```
# Encoding Sex and Embarked using One Hot encoding as they don't have
may values
x.shape
```

```
(792, 11)
```

```
sex = pd.get_dummies(x["Sex"],drop_first = True)
sex.head()
```

```
   male
0     1
1     0
2     0
3     0
4     1
```

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

```
   Q  S
0  0  1
1  0  0
2  0  1
3  0  1
4  0  1
```

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

```
x.head()
```

```
   PassengerId  Survived  Pclass  Name     Sex   Age  SibSp  Parch
Ticket  \
0            1         0       3    90    male  22.0      1      0
492
1            2         1       1   166  female  38.0      1      0
562
2            3         1       3   313  female  26.0      0      0
630
3            4         1       1   241  female  35.0      1      0
41
4            5         0       3    15    male  35.0      0      0
444

   Cabin Embarked  male  Q  S
0     36        S     1  0  1
1     66        C     0  0  0
2     36        S     0  0  1
3     43        S     0  0  1
4     36        S     1  0  1
```

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

x.head()
```

```
   PassengerId  Survived  Pclass  Name   Age  SibSp  Parch  Ticket
Cabin  \
0            1         0       3    90  22.0      1      0     492
36
1            2         1       1   166  38.0      1      0     562
66
2            3         1       3   313  26.0      0      0     630
36
3            4         1       1   241  35.0      1      0      41
43
4            5         0       3    15  35.0      0      0     444
36

   male  Q  S
0     1  0  1
1     0  0  0
2     0  0  1
3     0  0  1
4     1  0  1
```

```
x.shape
```

```
(792, 12)
```

7. Splitting into training and testing set

```
from sklearn.model_selection import train_test_split as ttp
x_train, x_test, y_train, y_test = ttp(x, y, test_size = 0.2,
random_state = 0)

x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

```
((633, 12), (159, 12), (633,), (159,))
```

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([[-0.1073643 , -0.78671839,  0.79752115, ..., -1.37492216,
        -0.32060483,  0.61569876],
       [ 0.98337258, -0.78671839, -0.41020825, ...,  0.7273139 ,
```

```
        -0.32060483,  0.61569876],
       [ 1.47736573,  1.27110286, -1.61793764, ..., -1.37492216,
        -0.32060483,  0.61569876],
       ...,
       [ 1.03869981,  1.27110286,  0.79752115, ...,  0.7273139 ,
        -0.32060483, -1.62417088],
       [ 0.71859225, -0.78671839,  0.79752115, ...,  0.7273139 ,
        -0.32060483,  0.61569876],
       [ 1.27186458, -0.78671839,  0.79752115, ...,  0.7273139 ,
         3.11910461, -1.62417088]])
```

x_test

```
array([[ 0.94305092, -0.79948437, -0.39597828, ...,  0.78895436,
        -0.27262488,  0.60884288],
       [-1.10475197,  1.25080619, -0.39597828, ..., -1.26750044,
        -0.27262488,  0.60884288],
       [ 1.2441984 , -0.79948437,  0.86323264, ...,  0.78895436,
        -0.27262488,  0.60884288],
       ...,
       [-1.42848551, -0.79948437,  0.86323264, ...,  0.78895436,
        -0.27262488,  0.60884288],
       [ 1.63192579, -0.79948437,  0.86323264, ..., -1.26750044,
        -0.27262488,  0.60884288],
       [-1.48118632,  1.25080619, -1.6551892 , ..., -1.26750044,
        -0.27262488,  0.60884288]])
```

y_train

```
419    24.1500
695    13.5000
820    93.5000
819    27.9000
281     7.8542
        ...
860    14.1083
222     8.0500
709    15.2458
628     7.8958
768    24.1500
Name: Fare, Length: 633, dtype: float64
```

y_test

```
705    26.0000
161    15.7500
785     7.2500
64     27.7208
564     8.0500
        ...
593     7.7500
```

```
89      8.0500
75      7.6500
888    23.4500
61     80.0000
Name: Fare, Length: 159, dtype: float64
```