Importing the library

In [1]:

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

import warnings
warnings.filterwarnings('ignore')
```

Importing the dataset

In [2]:

```
df=pd.read_csv(r"C:\Users\91956\Desktop\tested.csv")
df
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
413	1305	0	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	S
414	1306	1	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	C
415	1307	0	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	S
416	1308	0	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	S
417	1309	0	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	C

418 rows × 12 columns

EDA

Shape

In [3]:

```
df.shape
```

Out[3]:

```
(418, 12)
```

Columns and their types

In [4]:

```
df.columns
```

Out[4]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [5]:

```
df.dtypes
```

Out[5]:

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Survived     418 non-null    int64  
 2   Pclass       418 non-null    int64  
 3   Name         418 non-null    object 
 4   Sex          418 non-null    object 
 5   Age          332 non-null    float64
 6   SibSp        418 non-null    int64  
 7   Parch        418 non-null    int64  
 8   Ticket       418 non-null    object 
 9   Fare         417 non-null    float64
 10  Cabin        91 non-null     object 
 11  Embarked     418 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 39.3+ KB
```

Dropping the non-required columns

In [7]:

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

In [8]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  418 non-null    int64  
 1   Pclass    418 non-null    int64  
 2   Sex       418 non-null    object 
 3   Age       332 non-null    float64
 4   SibSp     418 non-null    int64  
 5   Parch     418 non-null    int64  
 6   Fare      417 non-null    float64
 7   Embarked  418 non-null    object 
dtypes: float64(2), int64(4), object(2)
memory usage: 26.2+ KB
```

Null value analysis

In [9]:

```
df.isnull().sum()/len(df)
```

Out[9]:

```
Survived    0.000000
Pclass      0.000000
Sex         0.000000
Age         0.205742
SibSp       0.000000
Parch       0.000000
Fare        0.002392
Embarked    0.000000
dtype: float64
```

In [10]:

```
df.Age.fillna(df.Age.mean(),inplace=True)
```

In [11]:

```
df.Fare.fillna(df.Fare.mean(),inplace=True)
```

In [12]:

```
df.isnull().sum()/len(df)
```

Out[12]:

```
Survived    0.0
Pclass      0.0
Sex         0.0
Age         0.0
SibSp       0.0
Parch       0.0
Fare        0.0
Embarked    0.0
dtype: float64
```

Duplicate values

In [13]:

```
df.duplicated().sum()
```

Out[13]:

38

In [14]:

```
df[df.duplicated()]
```

Out[14]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
79	1	3	female	24.00000	0	0	7.7500	Q
83	0	3	male	30.27259	0	0	7.8958	S
93	0	3	male	30.27259	0	0	8.0500	S
102	0	3	male	30.27259	0	0	7.7500	Q
107	0	3	male	30.27259	0	0	7.7500	Q
124	0	3	male	30.27259	0	0	7.7500	Q
148	0	1	male	30.27259	0	0	26.5500	S
158	0	1	male	42.00000	0	0	26.5500	S
180	0	2	male	30.00000	0	0	13.0000	S
183	0	3	male	30.27259	0	0	7.7500	Q
219	0	3	male	30.27259	0	0	8.0500	S
227	1	3	female	30.27259	0	0	7.7500	Q
248	1	2	female	29.00000	1	0	26.0000	S
255	0	3	male	30.27259	0	0	7.5500	S
256	0	3	male	30.27259	0	0	7.7500	Q
265	0	3	male	30.27259	0	0	7.8958	S
267	0	3	male	30.27259	0	0	7.5500	S
268	1	3	female	30.27259	0	0	8.0500	S
271	0	3	male	30.27259	0	0	7.7500	Q
282	1	3	female	30.27259	0	0	7.7500	Q
288	0	3	male	30.27259	0	0	7.2292	C
289	0	3	male	30.27259	0	0	8.0500	S
292	0	3	male	30.27259	0	0	7.2292	C
297	0	3	male	30.27259	2	0	21.6792	C
304	1	3	female	30.27259	0	0	7.7500	Q

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
320	0	3	male	26.00000	0	0	7.7750	S
322	0	2	male	26.00000	0	0	13.0000	S
332	0	3	male	30.27259	0	0	7.2250	C
339	0	3	male	30.27259	0	0	7.2292	C
346	0	2	male	26.00000	0	0	13.0000	S
351	0	2	male	25.00000	0	0	10.5000	S
358	0	3	male	30.27259	0	0	7.7500	Q
362	1	2	female	31.00000	0	0	21.0000	S
363	0	3	male	27.00000	0	0	8.6625	S
380	0	3	male	30.27259	0	0	7.7500	Q
410	1	3	female	30.27259	0	0	7.7500	Q
413	0	3	male	30.27259	0	0	8.0500	S
416	0	3	male	30.27259	0	0	8.0500	S

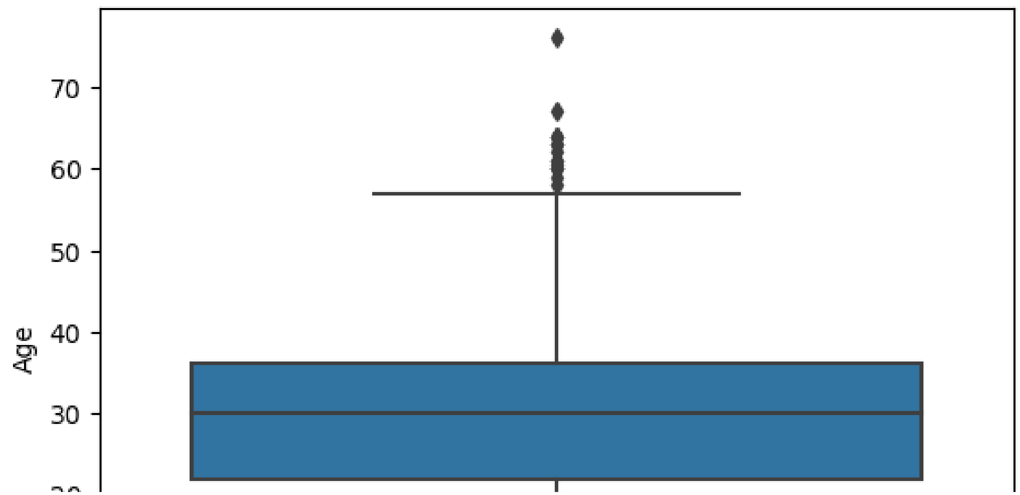
In [15]:

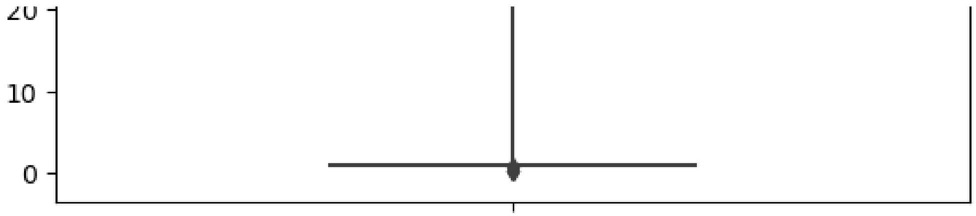
```
df.drop_duplicates(inplace=True)
```

Outlier's detection

In [16]:

```
sns.boxplot(y='Age',data=df)
plt.show()
```





In [17]:

```
Q1 = df['Age'].quantile(0.25)
Q3 = df['Age'].quantile(0.75)

IQR = Q3 - Q1

LL = Q1 - 1.5 * IQR
UL = Q3 + 1.5 * IQR

print("Q1: {} | Q3: {} | IQR: {} | LL: {} | UL: {}".format(Q1,Q3,IQR,LL,UL))
```

```
Q1: 22.0 | Q3: 36.125 | IQR: 14.125 | LL: 0.8125 | UL: 57.3125
```

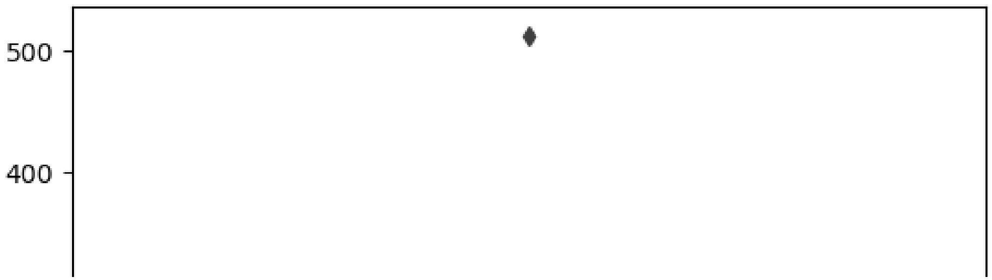
In [18]:

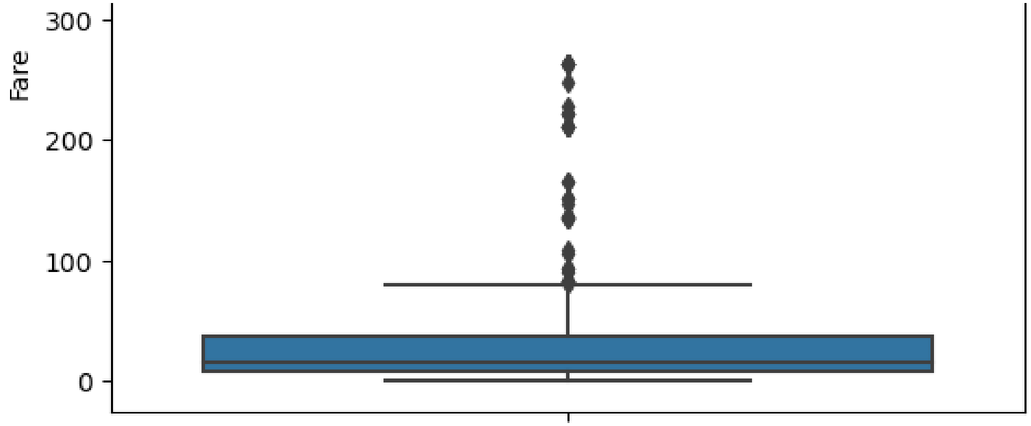
```
ul_outlier_count = df[df['Age'] > UL].shape[0]
ll_outlier_count = df[df['Age'] < LL].shape[0]

total_outlier_count = ll_outlier_count + ul_outlier_count

total_outlier_perc = total_outlier_count * 100 / df.shape[0]

print("UL_OC: {} | LL_OC: {} | T_OC: {} | T_OP: {}".format(ul_outlier_count,ll_outlier_count,total_outlier_count,total_outlier_pe
rc))
```

```
UL_OC: 16 | LL_OC: 3 | T_OC: 19 | T_OP: 5.0
```

In [19]:

```
sns.boxplot(y='Fare',data=df)
plt.show()
```





In [20]:

```
Q1 = df['Fare'].quantile(0.25)
Q3 = df['Fare'].quantile(0.75)

IQR = Q3 - Q1

LL = Q1 - 1.5 * IQR
UL = Q3 + 1.5 * IQR

print("Q1: {} | Q3: {} | IQR: {} | LL: {} | UL: {}".format(Q1,Q3,IQR,LL,UL))
```

```
Q1: 7.925 | Q3: 36.81355 | IQR: 28.88855 | LL: -35.407825 | UL: 80.146375
```

In [21]:

```
ul_outlier_count = df[df['Fare'] > UL].shape[0]
ll_outlier_count = df[df['Fare'] < LL].shape[0]

total_outlier_count = ll_outlier_count + ul_outlier_count

total_outlier_perc = total_outlier_count * 100 / df.shape[0]

print("UL_OC: {} | LL_OC: {} | T_OC: {} | T_OP: {}".format(ul_outlier_count,ll_outlier_count,total_outlier_count,total_outlier_pe
rc))
```

```
UL_OC: 41 | LL_OC: 0 | T_OC: 41 | T_OP: 10.789473684210526
```

INTERPRETATION

In [22]:

```
df
```

Out[22]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	34.50000	0	0	7.8292	Q
1	1	3	female	47.00000	1	0	7.0000	S
2	0	2	male	62.00000	0	0	9.6875	Q
3	0	3	male	27.00000	0	0	8.6625	S
4	1	3	female	22.00000	1	1	12.2875	S
...	...	...	...	...	...	...	...	...
411	1	1	female	37.00000	1	0	90.0000	Q
412	1	3	female	28.00000	0	0	7.7750	S
414	1	1	female	39.00000	0	0	108.9000	C
415	0	3	male	38.50000	0	0	7.2500	S
417	0	3	male	30.27259	1	1	22.3583	C

380 rows × 8 columns

In [23]:

```
#Number of Males and Females survived
sns.barplot(x=df['Sex'],y=df['Survived'])
plt.xticks(rotation=90)
plt.show()
```

Converting the values of object column into numerical for machine learning purpose

In [24]:

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

In [25]:

```
for col in ['Sex','Embarked']:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
df.head(10)
```

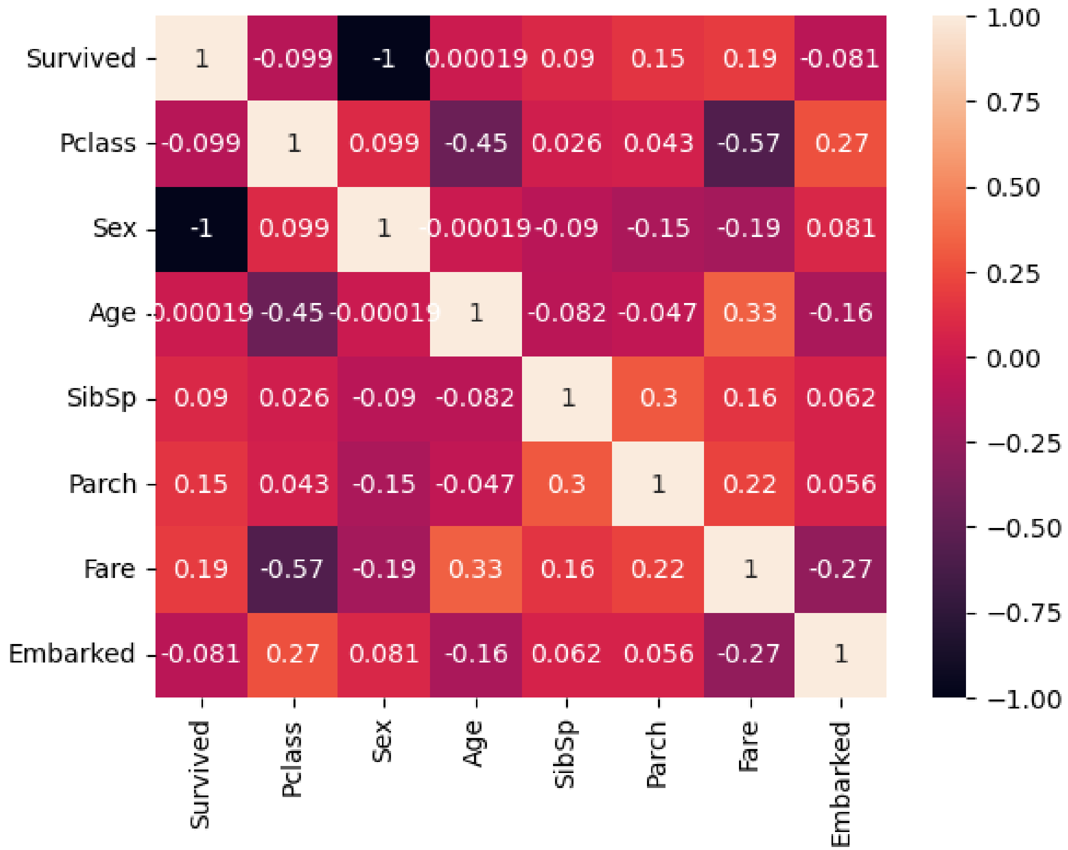
Out[25]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	34.5	0	0	7.8292	1
1	1	3	0	47.0	1	0	7.0000	2
2	0	2	1	62.0	0	0	9.6875	1
3	0	3	1	27.0	0	0	8.6625	2
4	1	3	0	22.0	1	1	12.2875	2
5	0	3	1	14.0	0	0	9.2250	2
6	1	3	0	30.0	0	0	7.6292	1
7	0	2	1	26.0	1	1	29.0000	2
8	1	3	0	18.0	0	0	7.2292	0
9	0	3	1	21.0	2	0	24.1500	2

Correlation

In [26]:

```
sns.heatmap(df.corr(),annot=True)
plt.show()
```



Machine learning

In [34]:

```
x=df.drop('Survived',axis=1)
y=df['Survived']
```

Splitting the dataset into training and testing

In [43]:

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=7)
```

Importing the algorithm and the performance measure¶

In [45]:

```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
```

In [60]:

```
from sklearn.tree import DecisionTreeClassifier
classifier2=DecisionTreeClassifier(random_state=7, max_depth=5,
                                   criterion='gini',max_leaf_nodes=7)

dt=classifier2.fit(x_train,y_train)
y_pred=classifier2.predict(x_test)

print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        46
           1       1.00      1.00      1.00        30

    accuracy                           1.00        76
   macro avg       1.00      1.00      1.00        76
weighted avg       1.00      1.00      1.00        76
```

In [59]:

```
from sklearn.model_selection import GridSearchCV
param_grid={'max_depth':[2,5,7,10,13,15,17,20],'criterion':['gini','entropy'],
           'max_leaf_nodes':[5,10,15,20,25,30],'min_samples_split':[10,20,30,40,50]}

dt1=DecisionTreeClassifier(random_state=7)
grid=GridSearchCV(dt1,param_grid,cv=10)
grid.fit(x_train,y_train)
grid.best_params_
```

Out[59]:

```
{'criterion': 'gini',
 'max_depth': 2,
 'max_leaf_nodes': 5,
 'min_samples_split': 10}
```

In [58]:

```
from sklearn.tree import DecisionTreeClassifier
classifier2=DecisionTreeClassifier(random_state=7, max_depth=2,
                                   criterion='gini',max_leaf_nodes=5,
                                  min_samples_split=10)

dt=classifier2.fit(x_train,y_train)
y_pred=classifier2.predict(x_test)


print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        46
           1       1.00      1.00      1.00        30

    accuracy                           1.00        76
   macro avg       1.00      1.00      1.00        76
weighted avg       1.00      1.00      1.00        76
```

In [57]:

```
from sklearn.svm import SVC
svc=SVC(random_state=7)
svm=svc.fit(x_train,y_train)
y_pred=svc.predict(x_test)


print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        46
           1       0.97      1.00      0.98        30

    accuracy                           0.99        76
   macro avg       0.98      0.99      0.99        76
weighted avg       0.99      0.99      0.99        76
```

In [56]:

```
from sklearn.ensemble import AdaBoostClassifier
adbc=AdaBoostClassifier(random_state=7)
adbc1=adbc.fit(x_train,y_train)
y_pred=adbc.predict(x_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support
```

```
           0       1.00      1.00      1.00        46
           1       1.00      1.00      1.00        30

    accuracy                           1.00        76
   macro avg       1.00      1.00      1.00        76
weighted avg       1.00      1.00      1.00        76
```

In [48]:

```python
from sklearn.ensemble import GradientBoostingClassifier
gbc=GradientBoostingClassifier(random_state=7,n_estimators=50,
                              max_depth=15)
gbc1=gbc.fit(x_train,y_train)
y_pred=gbc.predict(x_test)
y_pred_train=gbc.predict(x_train)

print(classification_report(y_test,y_pred))
print(classification_report(y_train,y_pred_train))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        46
           1       1.00      1.00      1.00        30

    accuracy                           1.00        76
   macro avg       1.00      1.00      1.00        76
weighted avg       1.00      1.00      1.00        76

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       190
           1       1.00      1.00      1.00       114

    accuracy                           1.00       304
   macro avg       1.00      1.00      1.00       304
weighted avg       1.00      1.00      1.00       304
```

In [ ]: