task2-titanic-survival-prediction

June 25, 2024

[114]:
```
#Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('/content/Titanic-Dataset.csv')
df
```

[114]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th…	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0					
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1					
4	5	0	3	Allen, Mr. William Henry	male	35.0	0					
..	…	…	…	…	…	…	…					
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0					
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0					
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1					
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0					
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0					

```
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]
```

[115]:
```
#Printing first and last rows
df.head()
```

[115]:
```
   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   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
```

[116]:
```
df.tail()
```

[116]:
```
     PassengerId  Survived  Pclass                                      Name  \
886          887         0       2                     Montvila, Rev. Juozas
887          888         1       1              Graham, Miss. Margaret Edith
888          889         0       3  Johnston, Miss. Catherine Helen "Carrie"
889          890         1       1                     Behr, Mr. Karl Howell
890          891         0       3                       Dooley, Mr. Patrick

        Sex   Age  SibSp  Parch      Ticket   Fare Cabin Embarked
886    male  27.0      0      0      211536  13.00   NaN        S
```

```
887  female  19.0      0      0      112053  30.00   B42        S
888  female   NaN      1      2  W./C. 6607  23.45   NaN        S
889    male  26.0      0      0      111369  30.00  C148        C
890    male  32.0      0      0      370376   7.75   NaN        Q
```

[117]:
```
df.shape
```

[117]:
```
(891, 12)
```

[118]:
```
#Check for missing values
df.isna().sum()
```

[118]:
```
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
```

[119]:
```
#Check the datatypes of each columns
df.dtypes
```

[119]:
```
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
```

[120]:
```
df.columns
```

[120]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
```

```
      dtype='object')
```

[121]:
```
#Check for the null values
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
```

[122]:
```
#Describe all columns
df.describe(include='all')
```

[122]:

	PassengerId	Survived	Pclass	Name	Sex	\
count	891.000000	891.000000	891.000000	891	891	
unique	NaN	NaN	NaN	891	2	
top	NaN	NaN	NaN	Braund, Mr. Owen Harris	male	
freq	NaN	NaN	NaN	1	577	
mean	446.000000	0.383838	2.308642	NaN	NaN	
std	257.353842	0.486592	0.836071	NaN	NaN	
min	1.000000	0.000000	1.000000	NaN	NaN	
25%	223.500000	0.000000	2.000000	NaN	NaN	
50%	446.000000	0.000000	3.000000	NaN	NaN	
75%	668.500000	1.000000	3.000000	NaN	NaN	
max	891.000000	1.000000	3.000000	NaN	NaN	

	Age	SibSp	Parch	Ticket	Fare	Cabin	\
count	714.000000	891.000000	891.000000	891	891.000000	204	
unique	NaN	NaN	NaN	681	NaN	147	
top	NaN	NaN	NaN	347082	NaN	B96 B98	
freq	NaN	NaN	NaN	7	NaN	4	
mean	29.699118	0.523008	0.381594	NaN	32.204208	NaN	

```
std      14.526497    1.102743    0.806057     NaN   49.693429      NaN
min       0.420000    0.000000    0.000000     NaN    0.000000      NaN
25%      20.125000    0.000000    0.000000     NaN    7.910400      NaN
50%      28.000000    0.000000    0.000000     NaN   14.454200      NaN
75%      38.000000    1.000000    0.000000     NaN   31.000000      NaN
max      80.000000    8.000000    6.000000     NaN  512.329200      NaN
```

```
       Embarked
count       889
unique        3
top           S
freq        644
mean        NaN
std         NaN
min         NaN
25%         NaN
50%         NaN
75%         NaN
max         NaN
```

[123]:
```
df.isna().sum()
```

[123]:
```
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
```

[124]:
```
#Dropping the unwanted columns and cabin contians large number of missing␣
 ↪values.
df.drop(["PassengerId","Name","Cabin","Ticket"],axis=1,inplace=True)
```

[125]:
```
#Filling the missing values
df['Age']=df['Age'].fillna(df['Age'].mean())
df['Embarked']=df['Embarked'].fillna(df['Embarked'].mode()[0])
```

[126]:
```
df.isna().sum()
```

[126]:
```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

[127]:
```
df['Survived'].unique()
```

[127]:
```
array([0, 1])
```

[128]:
```
#Fid the count for each columns
df['Survived'].value_counts()
#Hence this is a balanced dataset.
```

[128]:
```
Survived
0    549
1    342
Name: count, dtype: int64
```

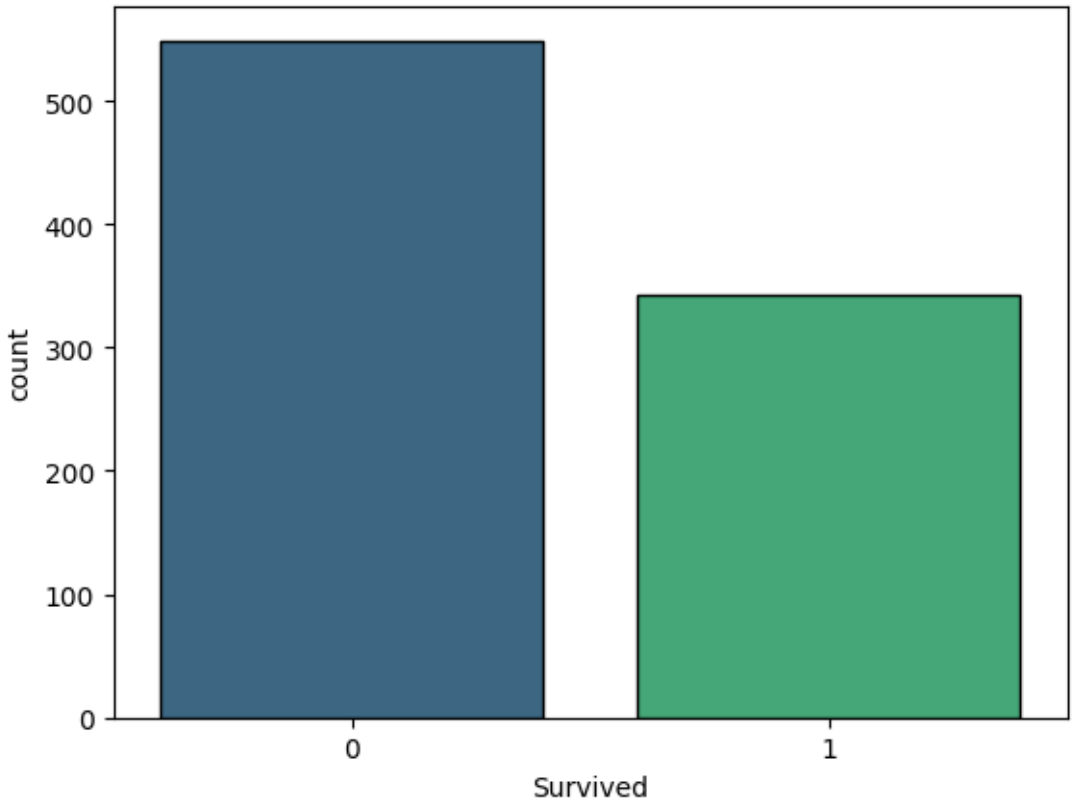
[129]:
```
#Visualization of each columns
sns.countplot(x='Survived',data=df,palette='viridis',edgecolor='k')
```

```
<ipython-input-129-ded6ec762ffe>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x='Survived',data=df,palette='viridis',edgecolor='k')
```

[129]:
```
<Axes: xlabel='Survived', ylabel='count'>
```



[130]:
```
df['Pclass'].unique()
```

[130]:
```
array([3, 1, 2])
```

[131]:
```
df['Pclass'].value_counts()
```

[131]:
```
Pclass
3    491
1    216
2    184
Name: count, dtype: int64
```

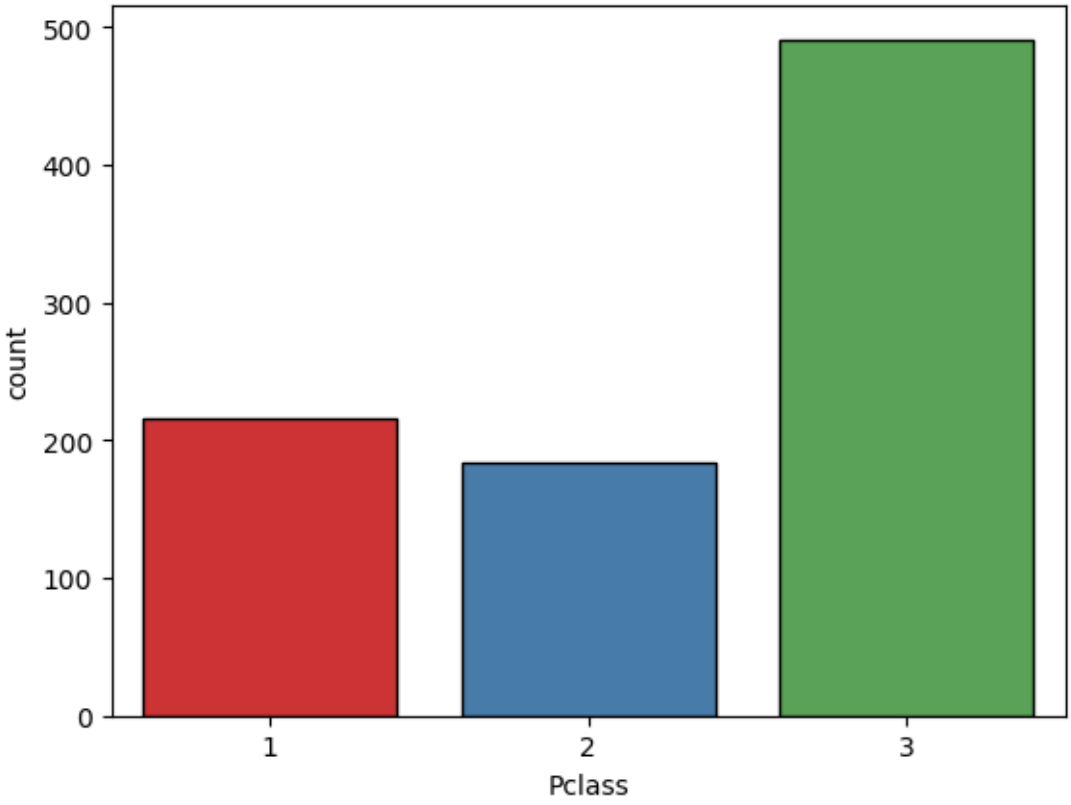
[132]:
```
sns.countplot(x='Pclass',data=df,palette='Set1',edgecolor='k')
```

```
<ipython-input-132-fa537bff240c>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x='Pclass',data=df,palette='Set1',edgecolor='k')
```

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



[133]:
```
df['Sex'].unique()
```

[133]:
```
array(['male', 'female'], dtype=object)
```

[134]:
```
df['Sex'].value_counts()
```

[134]:
```
Sex
male      577
female    314
Name: count, dtype: int64
```

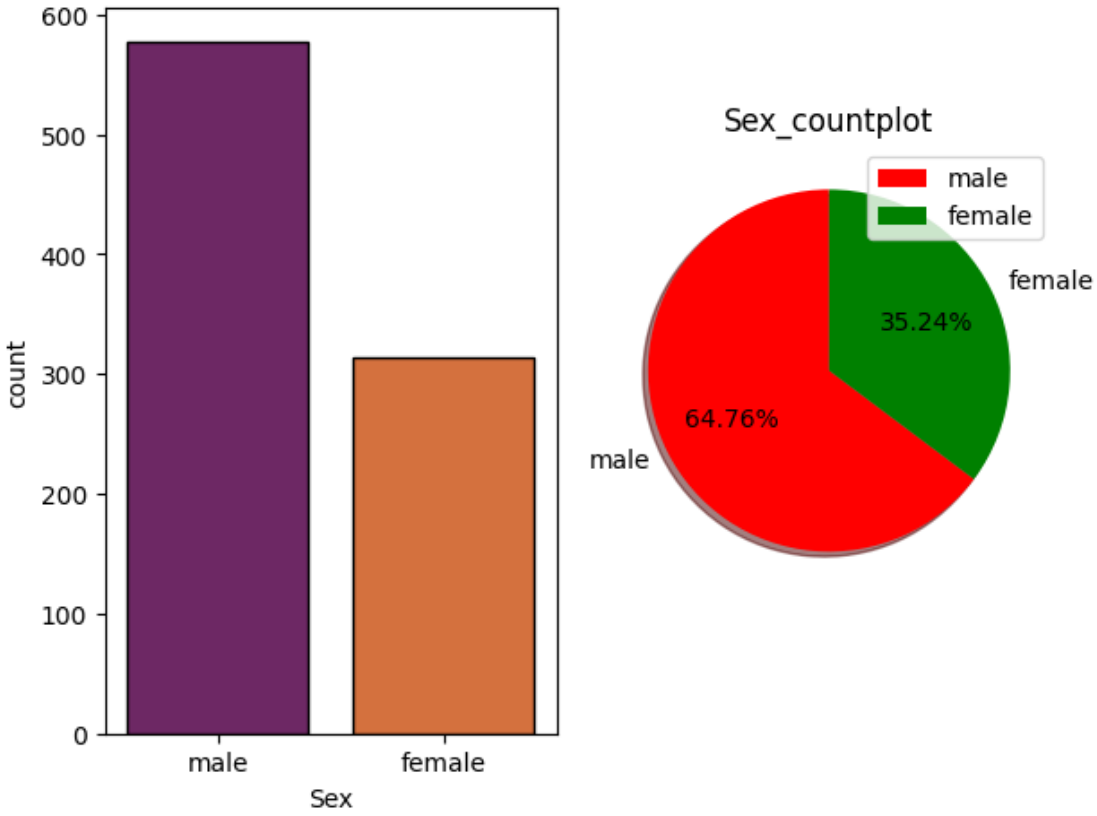
[135]:
```
plt.subplot(1,2,1)
sns.countplot(x='Sex',data=df,palette='inferno',edgecolor='k')
plt.subplot(1,2,2)
labels=['male','female']
colors=['red','green']
plt.pie(x=df['Sex'].
  ↪value_counts(),labels=labels,colors=colors,data=df,autopct='%1.
  ↪2f%%',startangle=90,shadow=True)
```

```
plt.legend(loc='upper right')
plt.title('Sex_countplot')
plt.tight_layout()
plt.show()
```

```
<ipython-input-135-b2fe888bb2e4>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x='Sex',data=df,palette='inferno',edgecolor='k')
```



[136]:
```
df['SibSp'].unique()
```

[136]:
```
array([1, 0, 3, 4, 2, 5, 8])
```

[137]:
```
df['Parch'].unique()
```

[137]:
```
array([0, 1, 2, 5, 3, 4, 6])
```

[138]:
```
df.columns
```

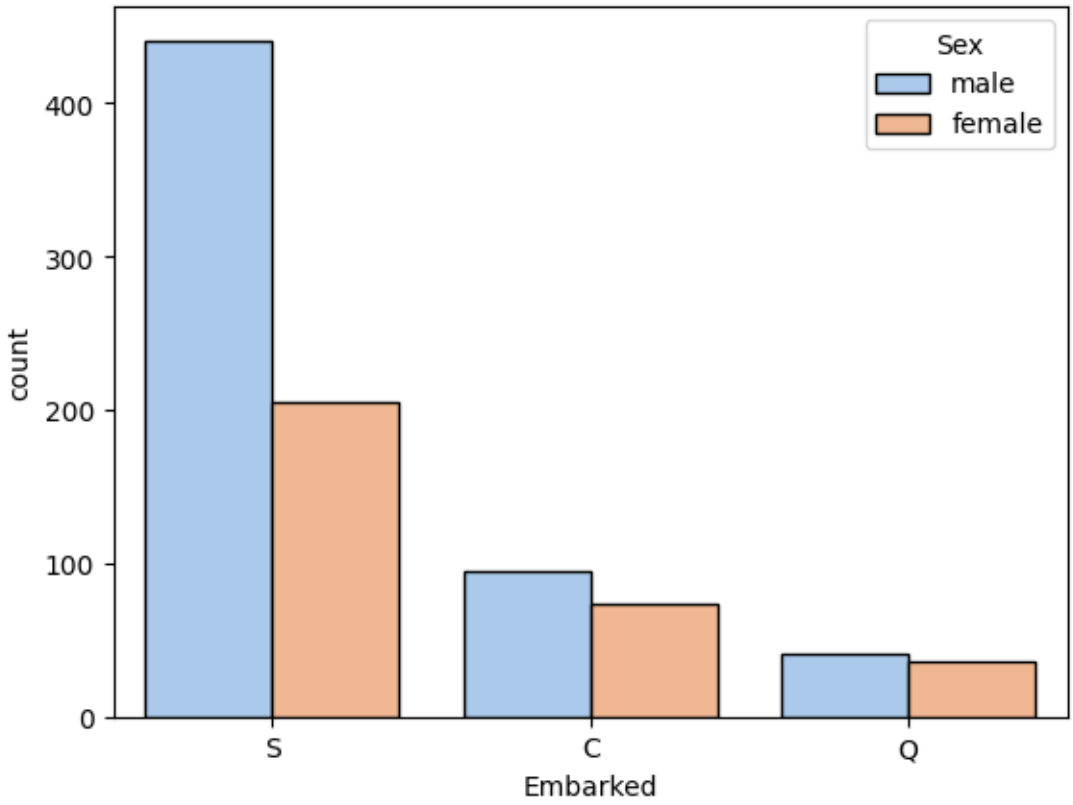
[138]:
```
Index(['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare',
       'Embarked'],
      dtype='object')
```

[139]:
```
df['Embarked'].unique()
```

[139]:
```
array(['S', 'C', 'Q'], dtype=object)
```

[140]:
```
sns.countplot(x='Embarked',data=df,hue='Sex',palette='pastel',edgecolor='k')
```

[140]:
```
<Axes: xlabel='Embarked', ylabel='count'>
```



[141]:
```
df.dtypes
```

[141]:
```
Survived      int64
Pclass        int64
Sex          object
Age         float64
SibSp         int64
```

```
Parch         int64
Fare        float64
Embarked     object
dtype: object
```

[142]:
```
df.columns
```

[142]:
```
Index(['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare',
       'Embarked'],
      dtype='object')
```

[143]:
```
#Encoding the categorized columns
from sklearn.preprocessing import LabelEncoder
encode=LabelEncoder()
columns=['Sex','Embarked']
for i in columns:
  df[i]=encode.fit_transform(df[i])
```

[144]:
```
df.dtypes
```

[144]:
```
Survived      int64
Pclass        int64
Sex           int64
Age         float64
SibSp         int64
Parch         int64
Fare        float64
Embarked      int64
dtype: object
```

[145]:
```
#correlation
corre=df.corr()
corre
```

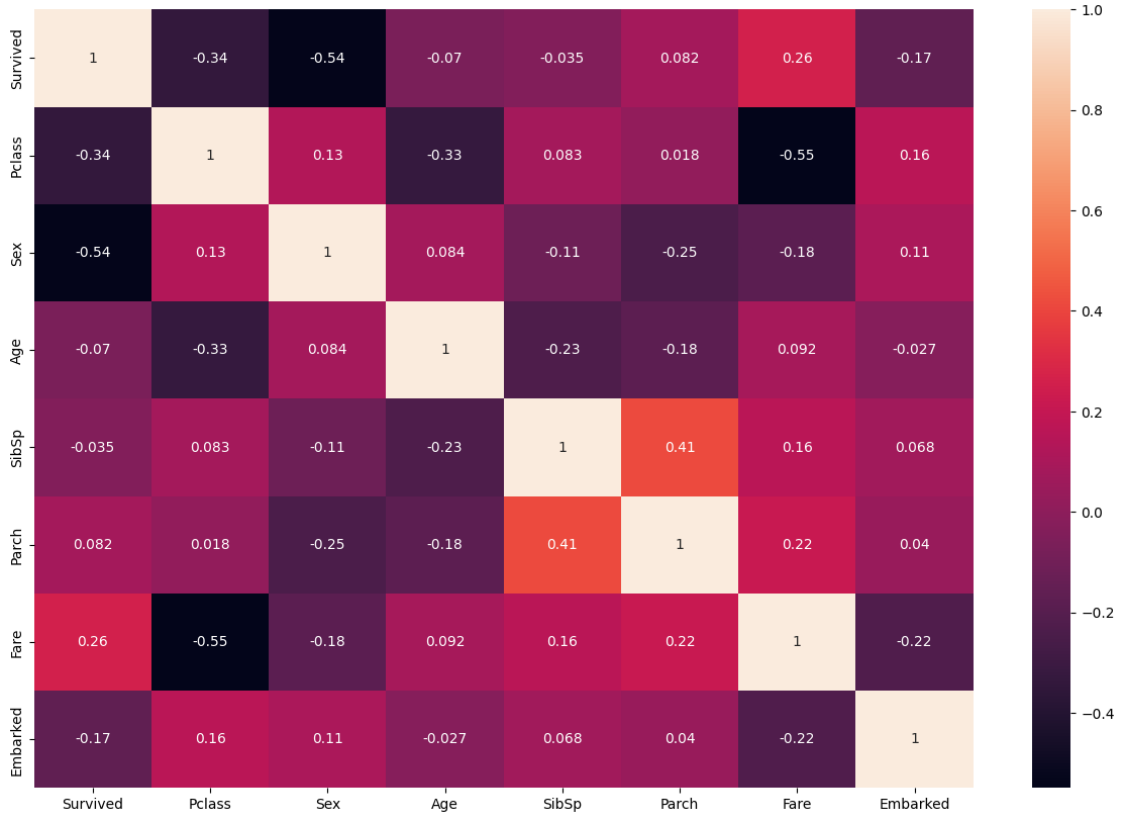
[145]:
```
           Survived    Pclass       Sex       Age     SibSp     Parch  \
Survived   1.000000 -0.338481 -0.543351 -0.069809 -0.035322  0.081629
Pclass    -0.338481  1.000000  0.131900 -0.331339  0.083081  0.018443
Sex       -0.543351  0.131900  1.000000  0.084153 -0.114631 -0.245489
Age       -0.069809 -0.331339  0.084153  1.000000 -0.232625 -0.179191
SibSp     -0.035322  0.083081 -0.114631 -0.232625  1.000000  0.414838
Parch      0.081629  0.018443 -0.245489 -0.179191  0.414838  1.000000
Fare       0.257307 -0.549500 -0.182333  0.091566  0.159651  0.216225
Embarked  -0.167675  0.162098  0.108262 -0.026749  0.068230  0.039798

               Fare  Embarked
Survived   0.257307 -0.167675
Pclass    -0.549500  0.162098
```

```
Sex       -0.182333  0.108262
Age        0.091566 -0.026749
SibSp      0.159651  0.068230
Parch      0.216225  0.039798
Fare       1.000000 -0.224719
Embarked  -0.224719  1.000000
```

[146]:
```
#Heatmat Visualization
plt.figure(figsize=(15,10))
sns.heatmap(corre,annot=True)
```

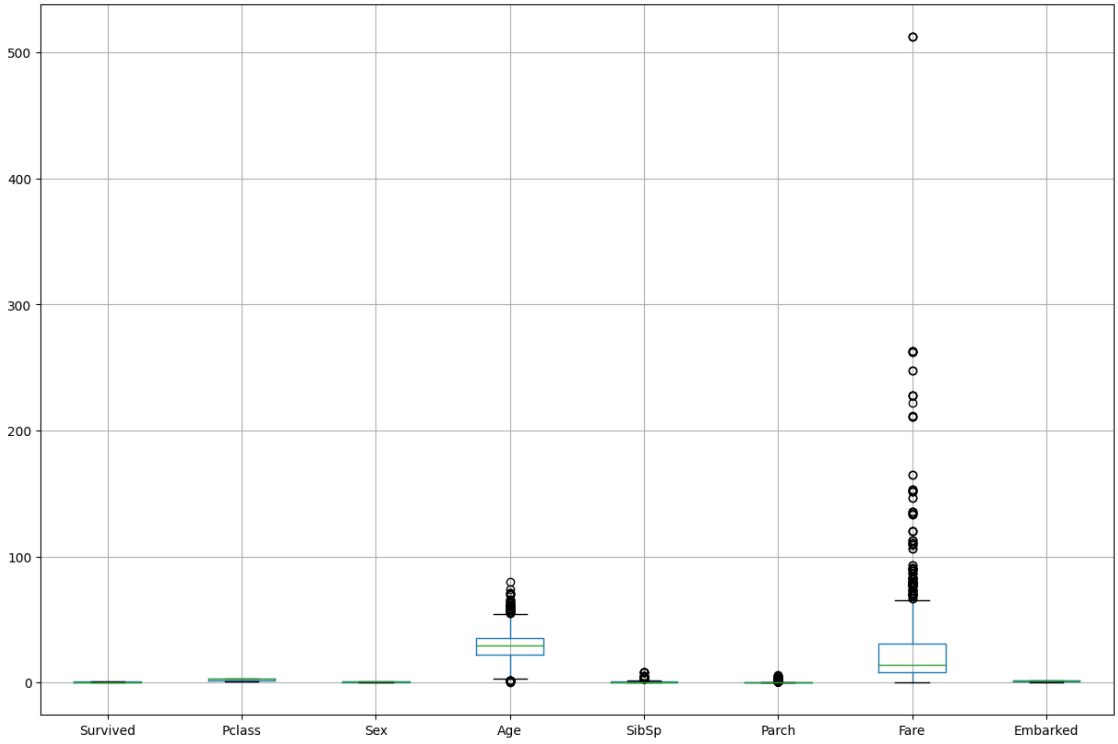
[146]: <Axes: >



Observation: corrulation between columns is not>0.95..No need to remove any feature after pearson corrulation.

[147]:
```
#Boxplot to find outliers
plt.figure(figsize=(15,10))
df.boxplot()
```

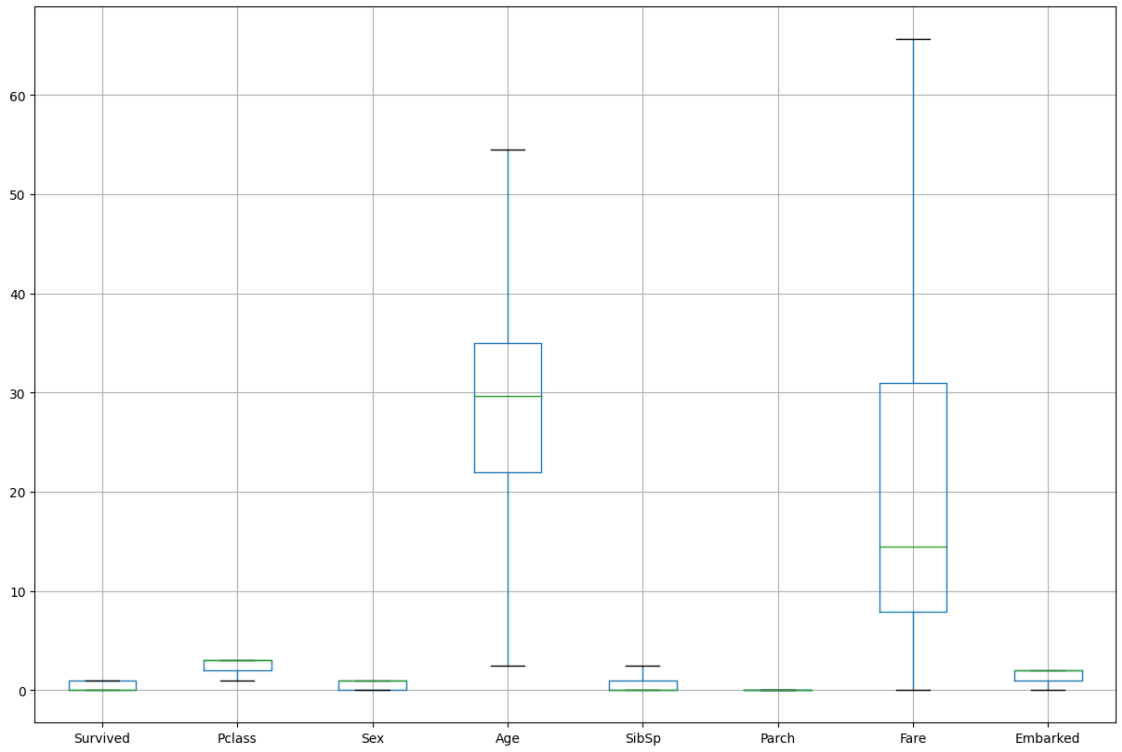
[147]: <Axes: >



[148]:
```
#Code to remove outliers
colmns=df.columns[df.columns!='Survived']
corr_pairs=[]
def out_rem(dfe,cols):
  for i in cols:
    q1=dfe[i].quantile(0.25)
    q3=dfe[i].quantile(0.75)
    IQR=q3-q1
    upper=q3+(1.5*IQR)
    lower=q1-(1.5*IQR)
    dfe[i]=dfe[i].clip(lower,upper)
out_rem(df,colmns)
```

[149]:
```
plt.figure(figsize=(15,10))
df.boxplot()
```

[149]: <Axes: >



Obseravation:Outliers are removed

[150]:
```
#Separatng x and y
x=df.drop(['Survived'],axis=1)
x.values
```

[150]:
```
array([[ 3.        ,  1.        , 22.        , …,  0.        ,
         7.25      ,  2.        ],
       [ 1.        ,  0.        , 38.        , …,  0.        ,
        65.6344    ,  0.        ],
       [ 3.        ,  0.        , 26.        , …,  0.        ,
         7.925     ,  2.        ],
       …,
       [ 3.        ,  0.        , 29.69911765, …,  0.        ,
        23.45      ,  2.        ],
       [ 1.        ,  1.        , 26.        , …,  0.        ,
        30.        ,  0.        ],
       [ 3.        ,  1.        , 32.        , …,  0.        ,
         7.75      ,  1.        ]])
```

[151]:
```
y=df['Survived'].values
y
```

[151]:
```
array([0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1,
       1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1,
       0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1,
       1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1,
       1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0])
```

[152]:
```
#Split the data set for training and testing
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
 ↪30,random_state=42)
```

[153]:
```
from sklearn.preprocessing import StandardScaler
norm=StandardScaler()
norm.fit(x_train)
x_train=norm.transform(x_train)
x_test=norm.transform(x_test)
```

[154]:
```
#Model creation using Randomforest classifier
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(n_estimators=10,criterion='entropy',random_state=42)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
y_pred
```

[154]:
```
array([0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0,
       0, 0, 0, 0])
```

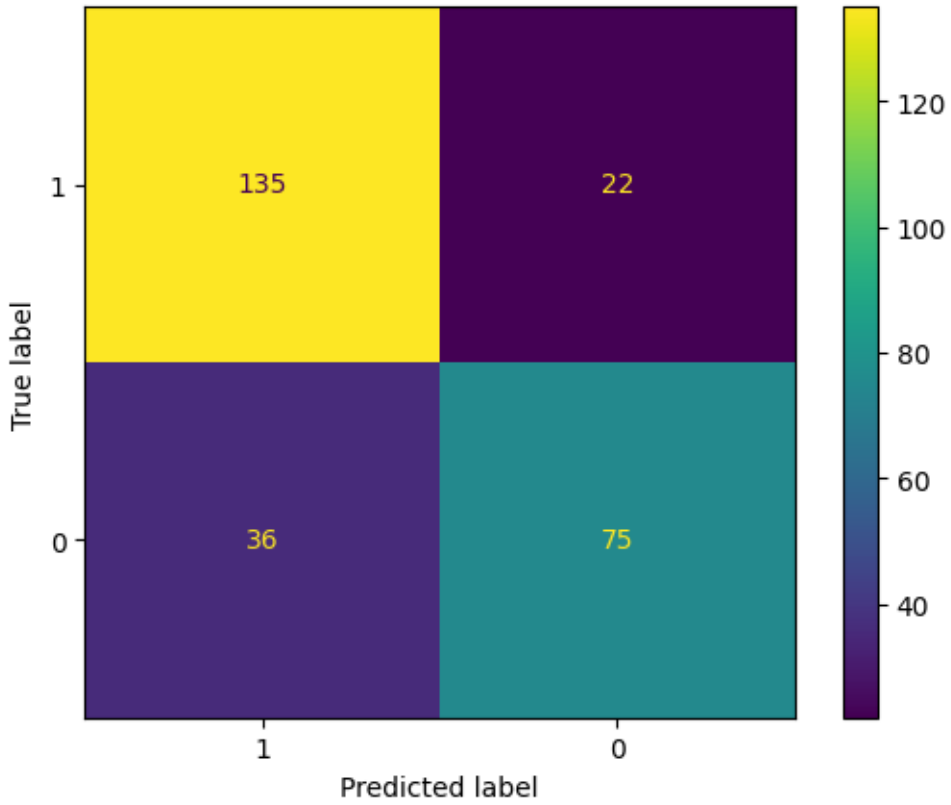
[155]:
```
#Performance evaluation of the model
from sklearn.metrics import␣
 ↪accuracy_score,confusion_matrix,classification_report,ConfusionMatrixDisplay
#create confusion matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[135  22]
 [ 36  75]]
```

[156]:
```
#Print accuracy score
print("Accuracy score is",accuracy_score(y_test,y_pred))
```

```
Accuracy score is 0.7835820895522388
```

[157]:
```
#Plot confusion matrix
labels=[1,0]
cmd=ConfusionMatrixDisplay(cm,display_labels=labels)
cmd.plot()
plt.show()
```



[158]:
```
#Display classification report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.86      0.82       157
           1       0.77      0.68      0.72       111

    accuracy                           0.78       268
   macro avg       0.78      0.77      0.77       268
weighted avg       0.78      0.78      0.78       268
```

[159]:
```
model_tune=RandomForestClassifier(random_state=42)
```

[160]:
```
from sklearn.model_selection import GridSearchCV
params={'n_estimators': [50, 100],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'max_features': ['sqrt', 'log2']}
```

[161]:
```
tuning=GridSearchCV(model_tune,params,cv=10,scoring='accuracy')
tuning.fit(x_train,y_train)
```

[161]:
```
GridSearchCV(cv=10, estimator=RandomForestClassifier(random_state=42),
             param_grid={'max_depth': [None, 10, 20],
                         'max_features': ['sqrt', 'log2'],
                         'min_samples_split': [2, 5],
                         'n_estimators': [50, 100]},
             scoring='accuracy')
```

[162]:
```
# Get best parameters and best scores for each algorithm
best_params_rf = tuning.best_params_
best_params_rf
```

[162]:
```
{'max_depth': 10,
 'max_features': 'sqrt',
 'min_samples_split': 5,
 'n_estimators': 100}
```

[163]:
```
model_final=RandomForestClassifier(max_depth=10,max_features='sqrt',min_samples_split=5,n_esti
model_final.fit(x_train,y_train)
y_pred2=model_final.predict(x_test)
y_pred2
```

[163]:
```
array([0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0,
       0, 0, 0, 0])
```

[164]:
```
print("Accuracy score is: ",accuracy_score(y_test,y_pred))
```

```
Accuracy score is:  0.7835820895522388
```