In [638]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [639]:
```python
data=pd.read_csv("D:\TITANIC.csv")
data
```

Out[639]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emb
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	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	

891 rows × 12 columns

In [640]:
```
data.info()
data.shape
```

```
<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
```

Out[640]:
```
(891, 12)
```

In [641]:
```
data.describe()
```

Out[641]:

	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
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [642]:
```
data.isnull().sum()
```

Out[642]:
```
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
```

In [643]:
```
data['Age']=data['Age'].fillna(value=data['Age'].median())
data['Embarked'].fillna('S')
data
```

Out[643]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emb
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	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	28.0	1	2	W./C. 6607	23.4500	NaN	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	

891 rows × 12 columns

In [644]:
```
data['Age'].describe()
```

Out[644]:
```
count    891.000000
mean      29.361582
std       13.019697
min        0.420000
25%       22.000000
50%       28.000000
75%       35.000000
max       80.000000
Name: Age, dtype: float64
```

In [645]:
```
q3=data["Age"].quantile(0.75)
q3
q1=data["Age"].quantile(0.25)
q1
IQR=data["Age"].quantile(0.75)-data['Age'].quantile(0.25)
print(IQR)
```

```
13.0
```

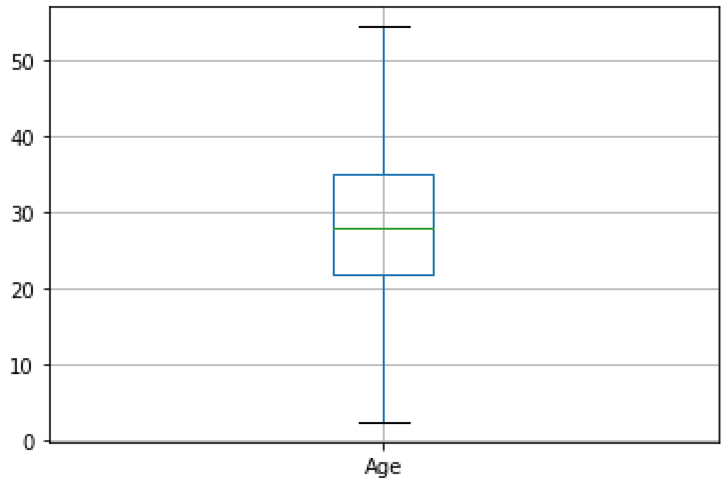
In [646]:
```
upper_outlierlimit=data['Age'].quantile(0.75)+1.5*IQR
lower_outlierlimit=data['Age'].quantile(0.25)-1.5*IQR
print(upper_outlierlimit)
print(lower_outlierlimit)
```

```
54.5
2.5
```

In [647]:
```
outliervalues=data[(data['Age']>=upper_outlierlimit)|(data['Age']<=lower_outlie
rlimit)]
outliervalues
data['Age']=np.where(data['Age']>=54.5,54.5,data['Age'])
data['Age']=np.where(data['Age']<=2.5,2.5,data['Age'])
print(data['Age'],data['Age'])
```

```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888    28.0
889    26.0
890    32.0
Name: Age, Length: 891, dtype: float64 0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888    28.0
889    26.0
890    32.0
Name: Age, Length: 891, dtype: float64
```

In [648]:
```
data.boxplot(column=['Age'])          # NEW BOXPLOT FOR 'AGE' AFTER REMOVING OUTLIERS
plt.show()
```



In [649]:
```
data['Fare'].describe()
```

Out[649]:
```
count    891.000000
mean      32.204208
std       49.693429
min        0.000000
25%        7.910400
50%       14.454200
75%       31.000000
max      512.329200
Name: Fare, dtype: float64
```

In [650]:
```
q3=data["Fare"].quantile(0.75)
q3
q1=data["Fare"].quantile(0.25)
q1
IQR=data["Fare"].quantile(0.75)-data['Fare'].quantile(0.25)
print(IQR)
```

```
23.0896
```

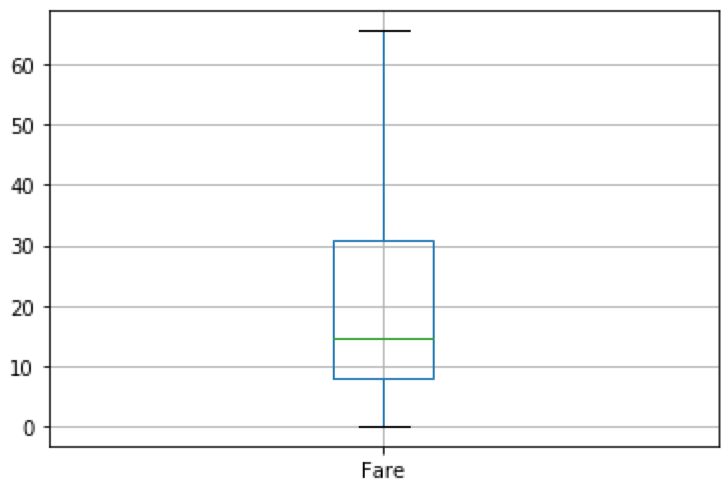
In [651]:
```
upper_outlierlimit=data['Fare'].quantile(0.75)+1.5*IQR
lower_outlierlimit=data['Fare'].quantile(0.25)-1.5*IQR
print(upper_outlierlimit)
print(lower_outlierlimit)
```

```
65.6344
-26.724
```

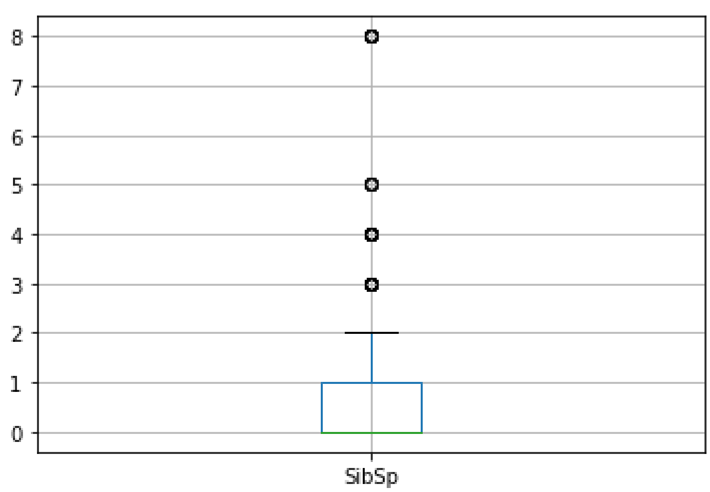
In [652]:
```
data['Fare']=np.where(data['Fare']>=65.6344,65.6344,data['Fare'])

print(data['Fare'])
```

```
0       7.2500
1      65.6344
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
```

In [653]:
```
data.boxplot(column=['Fare'])    # NEW BOXPLOT FOR 'FARE' AFTER REMOVING OUTLIE
RS
plt.show()
```



In [654]:
```
data.boxplot(column=['SibSp'])
plt.show()
```



In [655]:
```
data['SibSp'].describe()
```

Out[655]:
```
count    891.000000
mean       0.523008
std        1.102743
min        0.000000
25%        0.000000
50%        0.000000
75%        1.000000
max        8.000000
Name: SibSp, dtype: float64
```

In [656]:
```
q3=data["SibSp"].quantile(0.75)
q1=data["SibSp"].quantile(0.25)
IQR=data["SibSp"].quantile(0.75)-data['SibSp'].quantile(0.25)
print(IQR)
```

```
1.0
```

In [657]:
```
upper_outlierlimit=data['SibSp'].quantile(0.75)+1.5*IQR
lower_outlierlimit=data['SibSp'].quantile(0.25)-1.5*IQR
print(upper_outlierlimit)
print(lower_outlierlimit)
```

```
2.5
-1.5
```

In [658]:

```
outliervalues=data[(data['SibSp']>=upper_outlierlimit)|(data['SibSp']<=lower_outlierlimit)]
outliervalues
```

Out[658]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emba
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.5	3	1	349909	21.0750	NaN	
16	17	0	3	Rice, Master. Eugene	male	2.5	4	1	382652	29.1250	NaN	
24	25	0	3	Palsson, Miss. Torborg Danira	female	8.0	3	1	349909	21.0750	NaN	
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	65.6344	C23 C25 C27	
50	51	0	3	Panula, Master. Juha Niilo	male	7.0	4	1	3101295	39.6875	NaN	
59	60	0	3	Goodwin, Master. William Frederick	male	11.0	5	2	CA 2144	46.9000	NaN	
63	64	0	3	Skoog, Master. Harald	male	4.0	3	2	347088	27.9000	NaN	
68	69	1	3	Andersson, Miss. Erna Alexandra	female	17.0	4	2	3101281	7.9250	NaN	
71	72	0	3	Goodwin, Miss. Lillian Amy	female	16.0	5	2	CA 2144	46.9000	NaN	
85	86	1	3	Backstrom, Mrs. Karl Alfred (Maria Mathilda Gu...	female	33.0	3	0	3101278	15.8500	NaN	
88	89	1	1	Fortune, Miss. Mabel Helen	female	23.0	3	2	19950	65.6344	C23 C25 C27	
119	120	0	3	Andersson, Miss. Ellis Anna Maria	female	2.5	4	2	347082	31.2750	NaN	
159	160	0	3	Sage, Master. Thomas Henry	male	28.0	8	2	CA. 2343	65.6344	NaN	
164	165	0	3	Panula, Master. Eino Viljami	male	2.5	4	1	3101295	39.6875	NaN	
171	172	0	3	Rice, Master. Arthur	male	4.0	4	1	382652	29.1250	NaN	

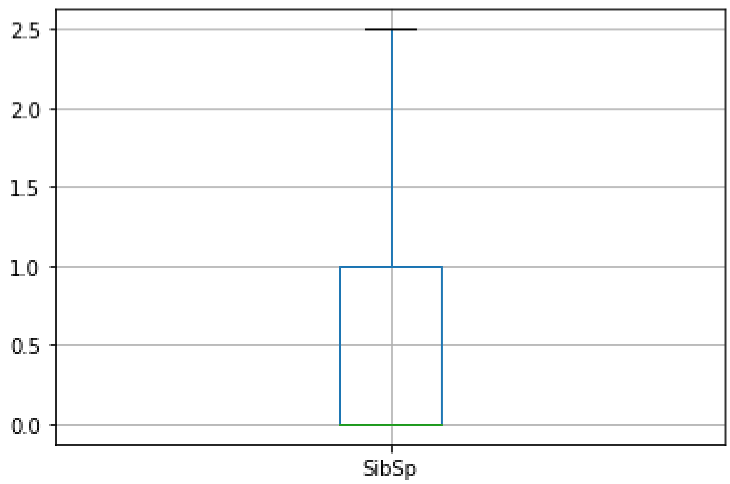
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emba
176	177	0	3	Lefebre, Master. Henry Forbes	male	28.0	3	1	4133	25.4667	NaN	
180	181	0	3	Sage, Miss. Constance Gladys	female	28.0	8	2	CA. 2343	65.6344	NaN	
182	183	0	3	Asplund, Master. Clarence Gustaf Hugo	male	9.0	4	2	347077	31.3875	NaN	
201	202	0	3	Sage, Mr. Frederick	male	28.0	8	2	CA. 2343	65.6344	NaN	
229	230	0	3	Lefebre, Miss. Mathilde	female	28.0	3	1	4133	25.4667	NaN	
233	234	1	3	Asplund, Miss. Lillian Gertrud	female	5.0	4	2	347077	31.3875	NaN	
261	262	1	3	Asplund, Master. Edvin Rojj Felix	male	3.0	4	2	347077	31.3875	NaN	
266	267	0	3	Panula, Mr. Ernesti Arvid	male	16.0	4	1	3101295	39.6875	NaN	
278	279	0	3	Rice, Master. Eric	male	7.0	4	1	382652	29.1250	NaN	
324	325	0	3	Sage, Mr. George John Jr	male	28.0	8	2	CA. 2343	65.6344	NaN	
341	342	1	1	Fortune, Miss. Alice Elizabeth	female	24.0	3	2	19950	65.6344	C23 C25 C27	
374	375	0	3	Palsson, Miss. Stina Viola	female	3.0	3	1	349909	21.0750	NaN	
386	387	0	3	Goodwin, Master. Sidney Leonard	male	2.5	5	2	CA 2144	46.9000	NaN	
409	410	0	3	Lefebre, Miss. Ida	female	28.0	3	1	4133	25.4667	NaN	
480	481	0	3	Goodwin, Master. Harold Victor	male	9.0	5	2	CA 2144	46.9000	NaN	
485	486	0	3	Lefebre, Miss. Jeannie	female	28.0	3	1	4133	25.4667	NaN	
541	542	0	3	Andersson, Miss. Ingeborg Constanzia	female	9.0	4	2	347082	31.2750	NaN	

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emba
542	543	0	3	Andersson, Miss. Sigrid Elisabeth	female	11.0	4	2	347082	31.2750	NaN	
634	635	0	3	Skoog, Miss. Mabel	female	9.0	3	2	347088	27.9000	NaN	
642	643	0	3	Skoog, Miss. Margit Elizabeth	female	2.5	3	2	347088	27.9000	NaN	
683	684	0	3	Goodwin, Mr. Charles Edward	male	14.0	5	2	CA 2144	46.9000	NaN	
686	687	0	3	Panula, Mr. Jaako Arnold	male	14.0	4	1	3101295	39.6875	NaN	
726	727	1	2	Renouf, Mrs. Peter Henry (Lillian Jefferys)	female	30.0	3	0	31027	21.0000	NaN	
787	788	0	3	Rice, Master. George Hugh	male	8.0	4	1	382652	29.1250	NaN	
792	793	0	3	Sage, Miss. Stella Anna	female	28.0	8	2	CA. 2343	65.6344	NaN	
813	814	0	3	Andersson, Miss. Ebba Iris Alfrida	female	6.0	4	2	347082	31.2750	NaN	
819	820	0	3	Skoog, Master. Karl Thorsten	male	10.0	3	2	347088	27.9000	NaN	
824	825	0	3	Panula, Master. Urho Abraham	male	2.5	4	1	3101295	39.6875	NaN	
846	847	0	3	Sage, Mr. Douglas Bullen	male	28.0	8	2	CA. 2343	65.6344	NaN	
850	851	0	3	Andersson, Master. Sigvard Harald Elias	male	4.0	4	2	347082	31.2750	NaN	
863	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	female	28.0	8	2	CA. 2343	65.6344	NaN	

In [659]:
```
data['SibSp']=np.where(data['SibSp']>=2.5,2.5,data['SibSp'])
print(data['SibSp'])
```

```
0      1.0
1      1.0
2      0.0
3      1.0
4      0.0
      ...
886    0.0
887    0.0
888    1.0
889    0.0
890    0.0
Name: SibSp, Length: 891, dtype: float64
```

In [660]:
```
data.boxplot(column=['SibSp'])                      # NEW BOXPLOT FOR 'Sibsp' A
FTER REMOVING OUTLIERS
plt.show()
```



In [661]:
```
data
```

Out[661]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emb
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1.0	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1.0	0	PC 17599	65.6344	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	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	0	113803	53.1000	C123	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0.0	0	373450	8.0500	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0.0	0	211536	13.0000	NaN	
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0.0	0	112053	30.0000	B42	
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	28.0	1.0	2	W./C. 6607	23.4500	NaN	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0.0	0	111369	30.0000	C148	
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0.0	0	370376	7.7500	NaN	

891 rows × 12 columns

In [662]:
```
data.drop('Cabin',axis=1,inplace=True)
```

In [663]:
```
data
```

Out[663]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1.0	0	A/5 21171	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1.0	0	PC 17599	65.6344	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0.0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1.0	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0.0	0	373450	8.0500	S
...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0.0	0	211536	13.0000	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0.0	0	112053	30.0000	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	28.0	1.0	2	W./C. 6607	23.4500	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0.0	0	111369	30.0000	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0.0	0	370376	7.7500	Q

891 rows × 11 columns

In [664]:
```
sex=pd.get_dummies(data['Sex'],drop_first=True)
embark=pd.get_dummies(data['Embarked'],drop_first=True)
```

In [665]:
```
data.drop(['Sex','Name','Ticket','Embarked'],axis=1,inplace=True)
```

In [666]:
```
data.head()
```
Out[666]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
0	1	0	3	22.0	1.0	0	7.2500
1	2	1	1	38.0	1.0	0	65.6344
2	3	1	3	26.0	0.0	0	7.9250
3	4	1	1	35.0	1.0	0	53.1000
4	5	0	3	35.0	0.0	0	8.0500

In [667]:
```
data=pd.concat([data,sex,embark],axis=1)
data.head()
```
Out[667]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	male	Q	S
0	1	0	3	22.0	1.0	0	7.2500	1	0	1
1	2	1	1	38.0	1.0	0	65.6344	0	0	0
2	3	1	3	26.0	0.0	0	7.9250	0	0	1
3	4	1	1	35.0	1.0	0	53.1000	0	0	1
4	5	0	3	35.0	0.0	0	8.0500	1	0	1

In [668]:
```
cols=data.columns
cols=['PassengerId','Pclass','Age','SibSp','Parch','Fare','male','Q','S','Survi
ved']
```
In [669]:
```
data=data[cols]
data
```
Out[669]:

	PassengerId	Pclass	Age	SibSp	Parch	Fare	male	Q	S	Survived
0	1	3	22.0	1.0	0	7.2500	1	0	1	0
1	2	1	38.0	1.0	0	65.6344	0	0	0	1
2	3	3	26.0	0.0	0	7.9250	0	0	1	1
3	4	1	35.0	1.0	0	53.1000	0	0	1	1
4	5	3	35.0	0.0	0	8.0500	1	0	1	0
...	...	...	...	...	...	...	...	...	...	...
886	887	2	27.0	0.0	0	13.0000	1	0	1	0
887	888	1	19.0	0.0	0	30.0000	0	0	1	1
888	889	3	28.0	1.0	2	23.4500	0	0	1	0
889	890	1	26.0	0.0	0	30.0000	1	0	0	1
890	891	3	32.0	0.0	0	7.7500	1	1	0	0

891 rows × 10 columns

In [670]:
```python
X=data.iloc[:,:-1].values
X
```

Out[670]:
```
array([[  1.,   3.,  22., ...,   1.,   0.,   1.],
       [  2.,   1.,  38., ...,   0.,   0.,   0.],
       [  3.,   3.,  26., ...,   0.,   0.,   1.],
       ...,
       [889.,   3.,  28., ...,   0.,   0.,   1.],
       [890.,   1.,  26., ...,   1.,   0.,   0.],
       [891.,   3.,  32., ...,   1.,   1.,   0.]])
```

In [671]:
```python
X.shape
```

Out[671]:
```
(891, 9)
```

In [672]:
```python
y=data.iloc[:,-1].values
y
```

Out[672]:
```
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, 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], dtype=int64)
```

In [673]:
```
y.shape
```

Out[673]:
```
(891,)
```

In [674]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_state=
0)
```

In [675]:
```
X_test
```

Out[675]:
```
array([[496.,   3.,  28., ...,   1.,   0.,   0.],
       [649.,   3.,  28., ...,   1.,   0.,   1.],
       [279.,   3.,   7., ...,   1.,   1.,   0.],
       ...,
       [216.,   1.,  31., ...,   0.,   0.,   0.],
       [834.,   3.,  23., ...,   1.,   0.,   1.],
       [373.,   3.,  19., ...,   1.,   0.,   1.]])
```

In [676]:
```
y_test
```

Out[676]:
```
array([0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1,
       0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0], dtype=int64)
```

In [677]:
```
from sklearn.preprocessing import StandardScaler
sc_x=StandardScaler()
X_train=sc_x.fit_transform(X_train)
X_test=sc_x.transform(X_test)
```

In [678]:
```
np.set_printoptions(suppress=True)
X_train
```

Out[678]:
```
array([[-1.16343003,  0.81925059, -0.08737676, ..., -1.37207547,
        -0.31426968, -1.62827579],
       [-0.01263834, -0.38096838,  0.16081042, ...,  0.72882288,
        -0.31426968,  0.61414657],
       [ 1.44220868, -0.38096838,  0.16081042, ...,  0.72882288,
        -0.31426968, -1.62827579],
       ...,
       [ 0.71863397,  0.81925059, -0.08737676, ...,  0.72882288,
         3.18198052, -1.62827579],
       [ 0.44921786,  0.81925059,  0.57445571, ..., -1.37207547,
        -0.31426968,  0.61414657],
       [ 0.93031806, -0.38096838,  2.1049433 , ...,  0.72882288,
        -0.31426968,  0.61414657]])
```

LogisticRegression

In [679]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression(random_state=0)
model.fit(X_train,y_train)
```

Out[679]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [680]:
```
y_pred=model.predict(X_test)
y_pred
```

Out[680]:
```
array([0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0,
       1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0], dtype=int64)
```

In [681]:
```
from sklearn.metrics import confusion_matrix
confusion=confusion_matrix(y_test,y_pred)
print(confusion)
```

```
[[93 17]
 [16 53]]
```

In [682]:
```
TN=confusion[0,0]
FP=confusion[0,1]
FN=confusion[1,0]
TP=confusion[1,1]
```

In [683]:
```
print(confusion)
print('TN:',TN)
print('FP:',FP)
print('FN:',FN)
print("TP:",TP)
```

```
[[93 17]
 [16 53]]
TN: 93
FP: 17
FN: 16
TP: 53
```

In [684]:
```
from sklearn import metrics
```

In [685]:
```
accuracy=metrics.accuracy_score(y_test,y_pred)
accuracy1=(TN+TP)/(TN+TP+FP+FN)
print("Accuracy from metrics:",accuracy)
print("Accuracy calculated:",accuracy1)
```

```
Accuracy from metrics: 0.8156424581005587
Accuracy calculated: 0.8156424581005587
```

In [686]:
```
print("Recall:",metrics.recall_score(y_test,y_pred),2)
print("Calculated Recall:",TP/(TP+FN))
print("Specificity:",TN/(TN+FP))
print('Precision:',round(metrics.precision_score(y_test,y_pred),2))
print("Precision calculated:",round(TP/float(TP+FP),2))
```

```
Recall: 0.7681159420289855 2
Calculated Recall: 0.7681159420289855
Specificity: 0.8454545454545455
Precision: 0.76
Precision calculated: 0.76
```

In [687]:
```
from sklearn.metrics import accuracy_score,f1_score,precision_score,roc_auc_sco
re,recall_score
accuuracy=accuracy_score(y_test,y_pred)
precision=precision_score(y_test,y_pred)
f1=f1_score(y_test,y_pred)
roc_auc=roc_auc_score(y_test,y_pred)
recall=recall_score(y_test,y_pred)
```

In [688]:
```
print("Accuracy is:",round(accuracy,4)*100)
print("F1 score is:",round(f1,2)*100)
print("Precision is:",round(precision,2)*100)
print("Recall is:",round(recall,2)*100)
print("Roc Auc is:",round(roc_auc,2)*100)
```

```
Accuracy is: 81.56
F1 score is: 76.0
Precision is: 76.0
Recall is: 77.0
Roc Auc is: 81.0
```

In [689]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.85      0.85       110
           1       0.76      0.77      0.76        69

    accuracy                           0.82       179
   macro avg       0.81      0.81      0.81       179
weighted avg       0.82      0.82      0.82       179
```

In [690]:
```
summary_lr=pd.DataFrame([accuracy,f1,precision,recall,roc_auc],index=["Accurac
y","F1","Precision","Recall","Roc_Auc"],columns=['Values'])
summary_lr
```

Out[690]:

	Values
Accuracy	0.815642
F1	0.762590
Precision	0.757143
Recall	0.768116
Roc_Auc	0.806785

DecisionTree

In [691]:
```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier(random_state=0)
dtc.fit(X_train,y_train)
```

Out[691]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=0, splitter='best')
```

In [692]:
```
y_pred_dt=dtc.predict(X_test)
y_pred_dt
```

Out[692]:
```
array([0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0], dtype=int64)
```

In [693]:
```
from sklearn.metrics import confusion_matrix
confusion=confusion_matrix(y_test,y_pred_dt)
print(confusion)
```

```
[[97 13]
 [22 47]]
```

In [694]:
```
TN=confusion[0,0]
FP=confusion[0,1]
FN=confusion[1,0]
TP=confusion[1,1]
print(confusion)
print('TN:',TN)
print('FP:',FP)
print('FN:',FN)
print("TP:",TP)
```

```
[[97 13]
 [22 47]]
TN: 97
FP: 13
FN: 22
TP: 47
```

In [695]:
```
accuracy=metrics.accuracy_score(y_test,y_pred_dt)
accuracy1=(TN+TP)/(TN+TP+FP+FN)
print("Accuracy from metrics:",accuracy)
print("Accuracy calculated:",accuracy1)
```

```
Accuracy from metrics: 0.8044692737430168
Accuracy calculated: 0.8044692737430168
```

In [696]:
```
from sklearn import metrics
from sklearn.metrics import accuracy_score,f1_score,precision_score,roc_auc_sco
re,recall_score
accuuracy=accuracy_score(y_test,y_pred_dt)
precision=precision_score(y_test,y_pred_dt)
f1=f1_score(y_test,y_pred_dt)
roc_auc=roc_auc_score(y_test,y_pred_dt)
recall=recall_score(y_test,y_pred_dt)
```

In [697]:
```
print("Accuracy is:",round(accuracy,2)*100)
print("F1 score is:",round(f1,2)*100)
print("Precision is:",round(precision,2)*100)
print("Recall is:",round(recall,2)*100)
print("Roc Auc is:",round(roc_auc,2)*100)
```

```
Accuracy is: 80.0
F1 score is: 73.0
Precision is: 78.0
Recall is: 68.0
Roc Auc is: 78.0
```

In [698]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred_dt))
```

```
              precision    recall  f1-score   support

           0       0.82      0.88      0.85       110
           1       0.78      0.68      0.73        69

    accuracy                           0.80       179
   macro avg       0.80      0.78      0.79       179
weighted avg       0.80      0.80      0.80       179
```

In [699]:
```
summary_dtc=pd.DataFrame([accuracy,f1,precision,recall,roc_auc],index=["Accurac
y","F1","Precision","Recall","Roc_Auc"],columns=['Values'])
summary_dtc
```

Out[699]:

	Values
Accuracy	0.804469
F1	0.728682
Precision	0.783333
Recall	0.681159
Roc_Auc	0.781489

RandomForest

In [700]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(X_train,y_train)
```

Out[700]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [701]:
```
y_pred_rt=rfc.predict(X_test)
y_pred_rt
```

Out[701]:
```
array([0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0,
       1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 1], dtype=int64)
```

In [702]:
```
from sklearn.metrics import confusion_matrix
confusion=confusion_matrix(y_test,y_pred_rt)
print(confusion)
```

```
[[101   9]
 [ 18  51]]
```

In [703]:
```
TN=confusion[0,0]
FP=confusion[0,1]
FN=confusion[1,0]
TP=confusion[1,1]
print(confusion)
print('TN:',TN)
print('FP:',FP)
print('FN:',FN)
print("TP:",TP)
```

```
[[101   9]
 [ 18  51]]
TN: 101
FP: 9
FN: 18
TP: 51
```

In [704]:
```
accuracy=metrics.accuracy_score(y_test,y_pred_rt)
accuracy1=(TN+TP)/(TN+TP+FP+FN)
print("Accuracy from metrics:",accuracy)
print("Accuracy calculated:",accuracy1)
```

```
Accuracy from metrics: 0.8491620111731844
Accuracy calculated: 0.8491620111731844
```

In [705]:
```
from sklearn import metrics
from sklearn.metrics import accuracy_score,f1_score,precision_score,roc_auc_sco
re,recall_score
accuuracy=accuracy_score(y_test,y_pred_rt)
precision=precision_score(y_test,y_pred_rt)
f1=f1_score(y_test,y_pred_rt)
roc_auc=roc_auc_score(y_test,y_pred_rt)
recall=recall_score(y_test,y_pred_rt)
```

In [706]:
```
print("Accuracy is:",round(accuracy,2)*100)
print("F1 score is:",round(f1,2)*100)
print("Precision is:",round(precision,2)*100)
print("Recall is:",round(recall,2)*100)
print("Roc Auc is:",round(roc_auc,2)*100)
```

```
Accuracy is: 85.0
F1 score is: 79.0
Precision is: 85.0
Recall is: 74.0
Roc Auc is: 83.0
```

In [707]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred_rt))
```

```
              precision    recall  f1-score   support

           0       0.85      0.92      0.88       110
           1       0.85      0.74      0.79        69

    accuracy                           0.85       179
   macro avg       0.85      0.83      0.84       179
weighted avg       0.85      0.85      0.85       179
```

In [708]:
```
summary_rf=pd.DataFrame([accuracy,f1,precision,recall,roc_auc],index=["Accurac
y","F1","Precision","Recall","Roc_Auc"],columns=['Values'])
summary_rf
```

Out[708]:

	Values
Accuracy	0.849162
F1	0.790698
Precision	0.850000
Recall	0.739130
Roc_Auc	0.828656

KNN(KNeighborsClassifier)

In [709]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(X_train,y_train)
```

Out[709]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

In [710]:
```
y_pred_k=knn.predict(X_test)
y_pred_k
```

Out[710]:
```
array([0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0], dtype=int64)
```

In [711]:
```
from sklearn.metrics import confusion_matrix
confusion=confusion_matrix(y_test,y_pred_k)
print(confusion)
```

```
[[99 11]
 [22 47]]
```

In [712]:

```
TN=confusion[0,0]
FP=confusion[0,1]
FN=confusion[1,0]
TP=confusion[1,1]
print(confusion)
print('TN:',TN)
print('FP:',FP)
print('FN:',FN)
print("TP:",TP)
```

```
[[99 11]
 [22 47]]
TN: 99
FP: 11
FN: 22
TP: 47
```

In [713]:

```
accuracy=metrics.accuracy_score(y_test,y_pred_k)
accuracy1=(TN+TP)/(TN+TP+FP+FN)
print("Accuracy from metrics:",accuracy)
print("Accuracy calculated:",accuracy1)
```

```
Accuracy from metrics: 0.8156424581005587
Accuracy calculated: 0.8156424581005587
```

In [714]:

```
from sklearn import metrics
from sklearn.metrics import accuracy_score,f1_score,precision_score,roc_auc_sco
re,recall_score
accuuracy=accuracy_score(y_test,y_pred_k)
precision=precision_score(y_test,y_pred_k)
f1=f1_score(y_test,y_pred_k)
roc_auc=roc_auc_score(y_test,y_pred_k)
recall=recall_score(y_test,y_pred_k)
```

In [715]:

```
print("Accuracy is:",round(accuracy,4)*100)
print("F1 score is:",round(f1,2)*100)
print("Precision is:",round(precision,2)*100)
print("Recall is:",round(recall,2)*100)
print("Roc Auc is:",round(roc_auc,2)*100)
```

```
Accuracy is: 81.56
F1 score is: 74.0
Precision is: 81.0
Recall is: 68.0
Roc Auc is: 79.0
```

In [716]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred_k))
```

```
              precision    recall  f1-score   support

           0       0.82      0.90      0.86       110
           1       0.81      0.68      0.74        69

    accuracy                           0.82       179
   macro avg       0.81      0.79      0.80       179
weighted avg       0.82      0.82      0.81       179
```

In [717]:
```
summary_k=pd.DataFrame([accuracy,f1,precision,recall,roc_auc],index=["Accuracy"
,"F1","Precision","Recall","Roc_Auc"],columns=['Values'])
summary_k
```

Out[717]:

	Values
Accuracy	0.815642
F1	0.740157
Precision	0.810345
Recall	0.681159
Roc_Auc	0.790580

In [729]:
```
Final_result={'LogesticRegression':[0.81],'DecisionTree':[0.80],'RandomForest':
[0.85],'knn KNeighborsClassifier ':[0.81]}
res=pd.DataFrame(Final_result)
res
```

Out[729]:

	LogesticRegression	DecisionTree	RandomForest	knn KNeighborsClassifier
0	0.81	0.8	0.85	0.81

MODEL SELECTION USING CROSS VALIDATION

In [719]:
```
from sklearn.model_selection import cross_val_score
```

In [720]:
```
# FOR LOGESTIC REGRESSION
```

In [721]:
```
scores_model=cross_val_score(model,X_train,y_train,cv=10,scoring='accuracy')
print('Score:',scores_model)
print("Mean Score:",scores_model.mean())
```

```
Score: [0.80555556 0.69444444 0.84507042 0.77464789 0.74647887 0.74647887
 0.78873239 0.78873239 0.84507042 0.81690141]
Mean Score: 0.7852112676056338
```

In [722]:
```
# FOR DECISIONTREE
```

In [723]:
```
scores_dtc=cross_val_score(dtc,X_train,y_train,cv=10,scoring='accuracy')
print('Score:',scores_dtc)
print("Mean Score:",scores_dtc.mean())
```

```
Score: [0.69444444 0.70833333 0.8028169  0.69014085 0.73239437 0.73239437
 0.73239437 0.66197183 0.78873239 0.8028169 ]
Mean Score: 0.7346439749608764
```

In [724]:
```
# FOR RANDOMFOREST
```

In [725]:
```
scores_rfc=cross_val_score(rfc,X_train,y_train,cv=10,scoring='accuracy')
print('Score:',scores_rfc)
print("Mean Score:",scores_rfc.mean())
```

```
Score: [0.73611111 0.81944444 0.81690141 0.85915493 0.8028169  0.8028169
 0.83098592 0.74647887 0.84507042 0.85915493]
Mean Score: 0.8118935837245695
```

In [726]:
```
# FOR KNN
```

In [727]:
```
scores_knn=cross_val_score(knn,X_train,y_train,cv=10,scoring='accuracy')
print('Score:',scores_knn)
print("Mean Score:",scores_knn.mean())
```

```
Score: [0.77777778 0.79166667 0.83098592 0.83098592 0.73239437 0.76056338
 0.70422535 0.76056338 0.88732394 0.83098592]
Mean Score: 0.7907472613458528
```

CONCLUSION

In [730]:
```
FROM ABOVE CROSS VALIDATION "RANDOMFOREST" HAS A AVERAGE ACCURACY OF 81.18% .
 SO "RANDOMFORESTCLASSIFIER" SUITS BEST MODEL FOR "TITANIC DATA SET".
```

In [ ]:
```

```