In [1]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree  import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```python
titanic=pd.read_csv('titanic.csv')
```

In [3]:

```python
titanic.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
```

In [4]:

```python
titanic
```

Out[4]:

	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	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine	female	NaN	1	2	W./C. 6607	23.4500	NaN	S

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
				Helen "Carrie"								
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [5]:

```
titanic.describe()
```

Out[5]:

	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 [6]:

```
titanic.dtypes
```

Out[6]:

```
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 [7]:

```
titanic=titanic.drop(['Name','Ticket','Cabin','PassengerId'],axis=1)
```

In [8]:

```
titanic
```

Out[8]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S
...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
887	1	1	female	19.0	0	0	30.0000	S
888	0	3	female	NaN	1	2	23.4500	S
889	1	1	male	26.0	0	0	30.0000	C
890	0	3	male	32.0	0	0	7.7500	Q

891 rows × 8 columns

In [9]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
list1=['Sex','Embarked']
for val in list1:
    titanic[val]=le.fit_transform(titanic[val].astype(str))
```

In [10]:

```
titanic
```

Out[10]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	22.0	1	0	7.2500	2
1	1	1	0	38.0	1	0	71.2833	0
2	1	3	0	26.0	0	0	7.9250	2
3	1	1	0	35.0	1	0	53.1000	2
4	0	3	1	35.0	0	0	8.0500	2
...	...	...	...	...	...	...	...	...
886	0	2	1	27.0	0	0	13.0000	2
887	1	1	0	19.0	0	0	30.0000	2
888	0	3	0	NaN	1	2	23.4500	2
889	1	1	1	26.0	0	0	30.0000	0
890	0	3	1	32.0	0	0	7.7500	1

891 rows × 8 columns

In [11]:

```
#to understand it has two or more unique variables and is it regression or classification
titanic.Survived.unique()
```

Out[11]:

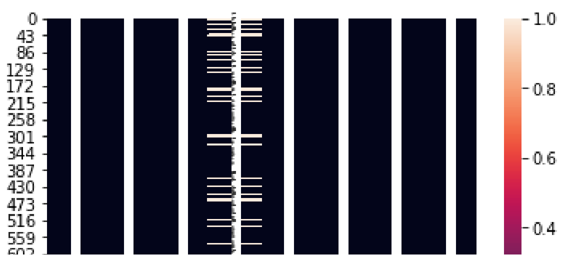
```
array([0, 1], dtype=int64)
```

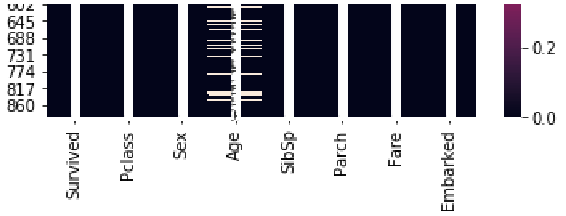
In [12]:

```
# identifing null values
sns.heatmap(titanic.isnull(),annot=True)
```

Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x21293837108>
```





In [13]:

```
titanic.isnull().sum()
```

Out[13]:

```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      0
dtype: int64
```

In [14]:

```
#filling null values with mean
from sklearn.impute import SimpleImputer
imp=SimpleImputer(strategy='mean')
titanic['Age']=imp.fit_transform(titanic['Age'].values.reshape(-1,1))
```

In [15]:

```
titanic
```

Out[15]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	22.000000	1	0	7.2500	2
1	1	1	0	38.000000	1	0	71.2833	0
2	1	3	0	26.000000	0	0	7.9250	2
3	1	1	0	35.000000	1	0	53.1000	2
4	0	3	1	35.000000	0	0	8.0500	2
...	...	...	...	...	...	...	...	...
886	0	2	1	27.000000	0	0	13.0000	2
887	1	1	0	19.000000	0	0	30.0000	2
888	0	3	0	29.699118	1	2	23.4500	2
889	1	1	1	26.000000	0	0	30.0000	0
890	0	3	1	32.000000	0	0	7.7500	1

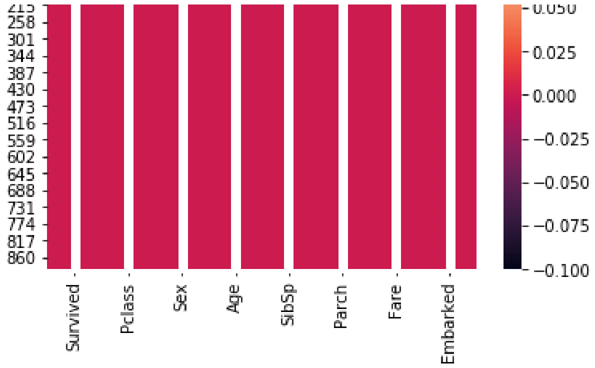
891 rows × 8 columns

In [16]:

```
sns.heatmap(titanic.isnull(),annot=True)
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x21293fbaa48>
```



In [17]:

```
titanic.skew()
```

Out[17]:

```
Survived    0.478523
Pclass     -0.630548
Sex        -0.618921
Age         0.434488
SibSp       3.695352
Parch       2.749117
Fare        4.787317
Embarked   -1.246689
dtype: float64
```

In [18]:

```
for col in titanic.columns:
    if titanic.skew().loc[col]>0.55:
        titanic[col]=np.log1p(titanic[col])
```

In [19]:

```
#reduced skewness
titanic.skew()
```

Out[19]:

```
Survived    0.478523
Pclass     -0.630548
Sex        -0.618921
Age         0.434488
SibSp       1.661245
Parch       1.675439
Fare        0.394928
Embarked   -1.246689
dtype: float64
```

In [20]:

```
titanic.corr()
```

Out[20]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
Survived	1.000000	-0.338481	-0.543351	-0.069809	0.029430	0.114999	0.329862	-0.163517
Pclass	-0.338481	1.000000	0.131900	-0.331339	0.022021	-0.002530	-0.661022	0.157112
Sex	-0.543351	0.131900	1.000000	0.084153	-0.165302	-0.256638	-0.263276	0.104057
Age	-0.069809	-0.331339	0.084153	1.000000	-0.231168	-0.231807	0.102485	-0.022239
SibSp	0.029430	0.022021	-0.165302	-0.231168	1.000000	0.473259	0.375371	0.036131
Parch	0.114999	-0.002530	-0.256638	-0.231807	0.473259	1.000000	0.363261	0.025070
Fare	0.329862	-0.661022	-0.263276	0.102485	0.375371	0.363261	1.000000	-0.197567

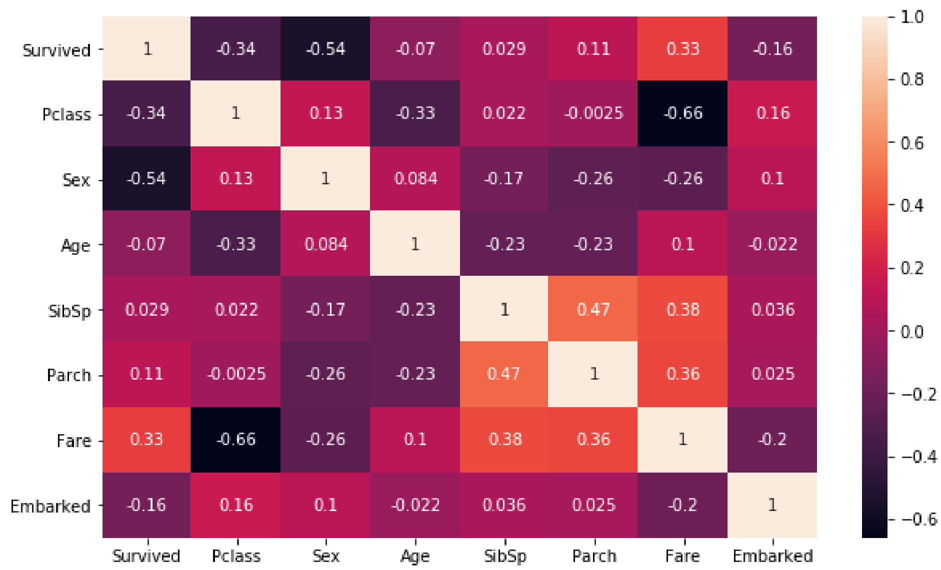
	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
Embarked	-0.163517	0.157112	0.104057	-0.022239	0.036131	0.025070	-0.197567	1.000000

In [21]:

```
plt.figure(figsize=(10,6))
sns.heatmap(titanic.corr(),annot=True)
```

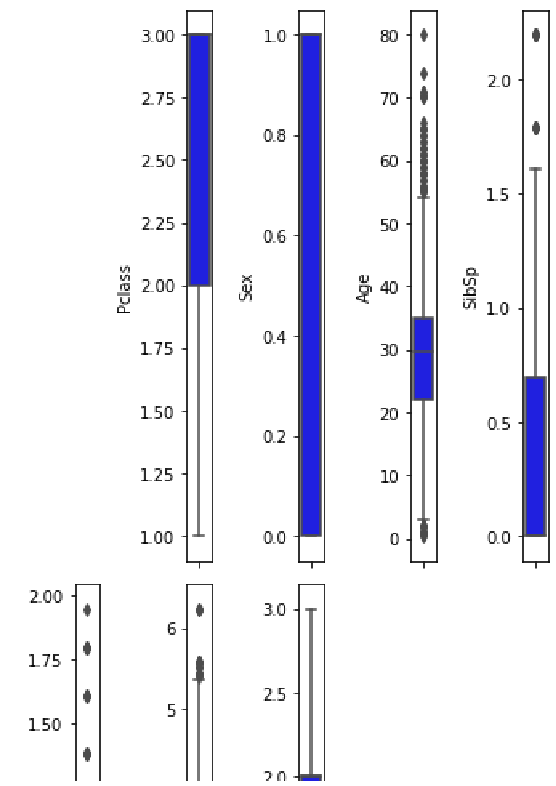
Out[21]:

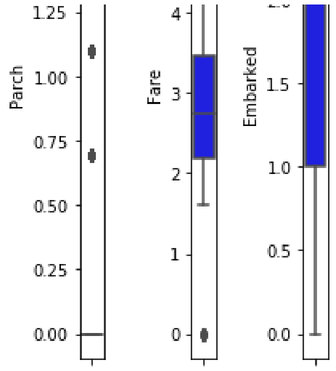
<matplotlib.axes._subplots.AxesSubplot at 0x2129bc7a148>



In [22]:

```
col=titanic.columns.values
ncol=5
nrow=5
plt.figure(figsize=(ncol,5*ncol))
for i in  range(1,len(col)):
    plt.subplot(nrow,ncol,i+1)
    sns.boxplot(titanic[col[i]],color='blue',orient='v')
    plt.tight_layout()
```





In [23]:

```
#Removing outliers
from scipy.stats import zscore
z_score=abs(zscore(titanic))
print(titanic.shape)
tit=titanic.loc[(z_score<3).all(axis=1)]
print(tit.shape)
```

```
(891, 8)
(844, 8)
```

In [24]:

```
tit
```

Out[24]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	22.000000	0.693147	0.000000	2.110213	2
1	1	1	0	38.000000	0.693147	0.000000	4.280593	0
2	1	3	0	26.000000	0.000000	0.000000	2.188856	2
3	1	1	0	35.000000	0.693147	0.000000	3.990834	2
4	0	3	1	35.000000	0.000000	0.000000	2.202765	2
...	...	...	...	...	...	...	...	...
886	0	2	1	27.000000	0.000000	0.000000	2.639057	2
887	1	1	0	19.000000	0.000000	0.000000	3.433987	2
888	0	3	0	29.699118	0.693147	1.098612	3.196630	2
889	1	1	1	26.000000	0.000000	0.000000	3.433987	0
890	0	3	1	32.000000	0.000000	0.000000	2.169054	1

844 rows × 8 columns

In [25]:

```
tit=pd.DataFrame(data=tit)
```

In [26]:

```
#x and y values allocation for training and testing
x=tit.iloc[:,1:-1]
```

In [27]:

```
x
```

Out[27]:

	Pclass	Sex	Age	SibSp	Parch	Fare
0	3	1	22.000000	0.693147	0.000000	2.110213
1	1	0	38.000000	0.693147	0.000000	4.280593
2	3	0	26.000000	0.000000	0.000000	2.188856
3	1	0	35.000000	0.693147	0.000000	3.990834
4	3	1	35.000000	0.000000	0.000000	2.202765
...	...	...	...	...	...	...
886	2	1	27.000000	0.000000	0.000000	2.639057
887	1	0	19.000000	0.000000	0.000000	3.433987
888	3	0	29.699118	0.693147	1.098612	3.196630
889	1	1	26.000000	0.000000	0.000000	3.433987
890	3	1	32.000000	0.000000	0.000000	2.169054

844 rows × 6 columns

In [28]:

```
x.shape
```

Out[28]:

```
(844, 6)
```

In [29]:

```
y=tit.iloc[:,0]
```

In [30]:

```
y
```

Out[30]:

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 844, dtype: int64
```

In [31]:

```
y.shape
```

Out[31]:

```
(844,)
```

In [32]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=40)
```

In [33]:

```
#we are using follwing models for prediction
#LogisticRegression
#KNeighborsClassifier
#GaussianNB
#SVC
```

```
#SVC
#DecisionTreeClassifier
#RandomForestClassifier
#AdaBoostClassifier
```

In [34]:

```
lr=LogisticRegression()
lr.fit(x_train,y_train)
lr.score(x_train,y_train)
pred=lr.predict(x_test)
print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
0.7992125984251969
[[132  17]
 [ 34  71]]
              precision    recall  f1-score   support

           0       0.80      0.89      0.84       149
           1       0.81      0.68      0.74       105

    accuracy                           0.80       254
   macro avg       0.80      0.78      0.79       254
weighted avg       0.80      0.80      0.80       254
```

In [35]:

```
lrscores=cross_val_score(lr,x,y,cv=5)
print(lrscores)
print(lrscores.mean(),lrscores.std())
```

```
[0.79289941 0.76923077 0.75739645 0.77514793 0.80952381]
0.7808396731473654 0.018357364834250496
```

In [36]:

```
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
knn.score(x_train,y_train)
predknn=knn.predict(x_test)
print(accuracy_score(y_test,predknn))
print(confusion_matrix(y_test,predknn))
print(classification_report(y_test,predknn))
```

```
0.8031496062992126
[[137  12]
 [ 38  67]]
              precision    recall  f1-score   support

           0       0.78      0.92      0.85       149
           1       0.85      0.64      0.73       105

    accuracy                           0.80       254
   macro avg       0.82      0.78      0.79       254
weighted avg       0.81      0.80      0.80       254
```

In [37]:

```
knnscores=cross_val_score(knn,x,y,cv=5)
print(knnscores)
print(knnscores.mean(),knnscores.std())
```

```
[0.75147929 0.74556213 0.76923077 0.76923077 0.80357143]
0.767814877430262 0.020221654866068396
```

In [38]:

In [38]:

```
mnb=MultinomialNB()
mnb.fit(x_train,y_train)
mnb.score(x_train,y_train)
predmnb=mnb.predict(x_test)
print(accuracy_score(y_test,predmnb))
print(confusion_matrix(y_test,predmnb))
print(classification_report(y_test,predmnb))
```

```
0.7244094488188977
[[139  10]
 [ 60  45]]
              precision    recall  f1-score   support

           0       0.70      0.93      0.80       149
           1       0.82      0.43      0.56       105

    accuracy                           0.72       254
   macro avg       0.76      0.68      0.68       254
weighted avg       0.75      0.72      0.70       254
```

In [39]:

```
mnbscores=cross_val_score(mnb,x,y,cv=5)
print(mnbscores)
print(mnbscores.mean(),mnbscores.std())
```

```
[0.67455621 0.76331361 0.74556213 0.72781065 0.76785714]
0.7358199492814876 0.03374785164419144
```

In [40]:

```
svc=SVC(kernel='rbf')
svc.fit(x_train,y_train)
svc.score(x_train,y_train)
predsvc=svc.predict(x_test)
print(accuracy_score(y_test,predsvc))
print(confusion_matrix(y_test,predsvc))
print(classification_report(y_test,predsvc))
```

```
0.6141732283464567
[[145   4]
 [ 94  11]]
              precision    recall  f1-score   support

           0       0.61      0.97      0.75       149
           1       0.73      0.10      0.18       105

    accuracy                           0.61       254
   macro avg       0.67      0.54      0.47       254
weighted avg       0.66      0.61      0.51       254
```

In [41]:

```
svcscores=cross_val_score(svc,x,y,cv=5)
print(svcscores)
print(svcscores.mean(),svcscores.std())
```

```
[0.63905325 0.70414201 0.66272189 0.63313609 0.66666667]
0.6611439842209073 0.02511677693054297
```

In [42]:

```
dtc=DecisionTreeClassifier()
dtc.fit(x_train,y_train)
dtc.score(x_train,y_train)
preddtc=dtc.predict(x_test)
print(accuracy_score(y_test,preddtc))
```

```
print(confusion_matrix(y_test,preddtc))
print(classification_report(y_test,preddtc))
```

```
0.7795275590551181
[[125  24]
 [ 32  73]]
              precision    recall  f1-score   support

           0       0.80      0.84      0.82       149
           1       0.75      0.70      0.72       105

    accuracy                           0.78       254
   macro avg       0.77      0.77      0.77       254
weighted avg       0.78      0.78      0.78       254
```

In [43]:

```
dtcscores=cross_val_score(dtc,x,y,cv=5)
print(dtcscores)
print(dtcscores.mean(),dtcscores.std())
```

```
[0.71597633 0.77514793 0.82248521 0.75739645 0.76785714]
0.7677726120033812 0.0341713035774167
```

In [44]:

```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
rf.score(x_train,y_train)
predrf=rf.predict(x_test)
print(accuracy_score(y_test,predrf))
print(confusion_matrix(y_test,predrf))
print(classification_report(y_test,predrf))
```

```
0.8188976377952756
[[133  16]
 [ 30  75]]
              precision    recall  f1-score   support

           0       0.82      0.89      0.85       149
           1       0.82      0.71      0.77       105

    accuracy                           0.82       254
   macro avg       0.82      0.80      0.81       254
weighted avg       0.82      0.82      0.82       254
```

In [45]:

```
rfscores=cross_val_score(rf,x,y,cv=5)
print(rfscores)
print(rfscores.mean(),rfscores.std())
```

```
[0.75739645 0.79881657 0.84615385 0.76923077 0.85119048]
0.8045576218653141 0.03849673064870086
```

In [46]:

```
from sklearn.ensemble import AdaBoostClassifier
ad=AdaBoostClassifier()
ad.fit(x_train,y_train)
ad.score(x_train,y_train)
predad=ad.predict(x_test)
print(accuracy_score(y_test,predad))
print(confusion_matrix(y_test,predad))
print(classification_report(y_test,predad))
```

```
0.7874015748031497
[[133  16]
```

```
 [ 38  67]]
              precision    recall  f1-score   support

           0       0.78      0.89      0.83       149
           1       0.81      0.64      0.71       105

    accuracy                           0.79       254
   macro avg       0.79      0.77      0.77       254
weighted avg       0.79      0.79      0.78       254
```

In [47]:

```
adscores=cross_val_score(ad,x,y,cv=5)
print(adscores)
print(adscores.mean(),adscores.std())
```

```
[0.73964497 0.79881657 0.81656805 0.81656805 0.83928571]
0.8021766694843618 0.03380179992055992
```

In [48]:

```
#RandomForestClassifier is the best model among all models
import joblib
joblib.dump(rf,'titanic.pkl')
```

Out[48]:

```
['titanic.pkl']
```

In [ ]:

```

```