In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
```

In [2]:
```
var=pd.read_csv('C://Users/Gopi/Desktop/machine learning/csv files/titanic.csv')
var.head(1)
```

Out[2]:

| | 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 | 7.25 | NaN | S |

In [3]:
```
var['Age'] = var['Age'].fillna(var['Age'].mean())
```

In [4]:
```
var['Cabin'] = var.Cabin.fillna(0)
```

In [5]:
```
print(var.shape)
varun=var
varun.drop(['Name','Ticket','Cabin'],axis=1,inplace = True)
print(varun.shape)
```

```
(891, 12)
(891, 9)
```

In [6]:
```
a=pd.get_dummies(varun['Sex'])
b=pd.get_dummies(varun['Embarked'])
```

In [7]:
```
varun=pd.concat([varun,a,b],axis='columns')
varun.shape
```

Out[7]: (891, 14)

In [8]:
```
varun.drop(['Sex','Embarked'],axis=1,inplace = True)
```

In [10]:
```
varun.head(1)
```

Out[10]:

| | PassengerId | Survived | Pclass | Age | SibSp | Parch | Fare | female | male | C | Q | S |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 3 | 22.0 | 1 | 0 | 7.25 | 0 | 1 | 0 | 0 | 1 |

In [11]:
```
varun.drop(['PassengerId','Survived'],axis=1,inplace = True)
```

In [12]:
```
varun.head(2)
```

Out[12]:

| | Pclass | Age | SibSp | Parch | Fare | female | male | C | Q | S |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 3 | 22.0 | 1 | 0 | 7.2500 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 38.0 | 1 | 0 | 71.2833 | 1 | 0 | 1 | 0 | 0 |

In [15]:
```
y=var['Survived']
X=varun
```

In [23]:
```
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)
```

In [27]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(712, 10)
(179, 10)
(712,)
(179,)
```

In [62]:
```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)

tree.fit(X_train,y_train)

y_pred=tree.predict(X_test)

from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_pred,y_test)
print(cm)
ac=accuracy_score(y_pred,y_test)
print(ac)
```

```
[[81 28]
 [18 52]]
0.7430167597765364
```

In [59]:
```
from sklearn.ensemble import RandomForestClassifier
forest = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)

forest.fit(X_train,y_train)

y_pred=forest.predict(X_test)

from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_pred,y_test)
print(cm)
ac=accuracy_score(y_pred,y_test)
print(ac)
```

```
[[84 24]
 [15 56]]
0.7821229050279329
```

In [60]:
```
from sklearn.naive_bayes import GaussianNB
nb= GaussianNB()

nb.fit(X_train,y_train)

y_pred=nb.predict(X_test)

from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_pred,y_test)
print(cm)
ac=accuracy_score(y_pred,y_test)
print(ac)
```

```
[[83 23]
 [16 57]]
0.7821229050279329
```

In [61]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train,y_train)

y_pred=knn.predict(X_test)

from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_pred,y_test)
print(cm)
ac=accuracy_score(y_pred,y_test)
print(ac)
```

```
[[79 37]
 [20 43]]
0.6815642458100558
```