In [1]:
```
import pandas as pd

```

In [2]:
```
data=pd.read_csv("/home/placement/Downloads/Titanic Dataset.csv")
```

In [3]:
```
data.describe()
```

Out[3]:

	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 [4]:
```
data.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 [5]:
```
list(data)
```

Out[5]:
```
['PassengerId',
 'Survived',
 'Pclass',
 'Name',
 'Sex',
 'Age',
 'SibSp',
 'Parch',
 'Ticket',
 'Fare',
 'Cabin',
 'Embarked']
```

In [6]:
```
data.isna().sum()#no.of null values
```

Out[6]:
```
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 [7]:
```
data1=data.drop(['Cabin','Name','PassengerId','Ticket','SibSp','Parch'],axis=1)#deleting columns
```

In [8]:
```
data1
```

Out[8]:

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

891 rows × 6 columns

In [9]:
```
data1.isna().sum()#finding null values
```

Out[9]:
```
Survived      0
Pclass        0
Sex           0
Age         177
Fare          0
Embarked      2
dtype: int64
```

In [10]:
```
data1.shape#no of rows and columns
```

Out[10]:
```
(891, 6)
```

In [11]:
```
data1['Sex']=data1['Sex'].map({'male':1,'female':0})#assinging 1 to male and 0 to female using map
```

In [12]:
```
data1
```

Out[12]:

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

891 rows × 6 columns

In [13]:
```
data1=data1.fillna(data1.mean())#fill null values using mean

```

```
/snap/jupyter/6/lib/python3.7/site-packages/ipykernel_launcher.py:1: FutureWarning: Dropping of nuisance co
lumns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  """Entry point for launching an IPython kernel.
```

In [14]:
```
data1
```

Out[14]:

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

891 rows × 6 columns

In [15]:
```
data1.isna().sum()
```

Out[15]:
```
Survived    0
Pclass      0
Sex         0
Age         0
Fare        0
Embarked    2
dtype: int64
```

In [16]:
```
data2=pd.get_dummies(data1)#where the pclass it shows"1" other pclass it shows "0"
data2
```

Out[16]:

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

891 rows × 8 columns

In [17]:
```
x=data2.drop(['Survived'],axis=1)#deleting churn
```

In [18]:
```
x
```

Out[18]:

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

891 rows × 7 columns

In [20]:
```
y=data['Survived']
```

In [21]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

In [22]:
```
x.head(5)
```

Out[22]:

	Pclass	Sex	Age	Fare	Embarked_C	Embarked_Q	Embarked_S
0	3	1	22.0	7.2500	0	0	1
1	1	0	38.0	71.2833	1	0	0
2	3	0	26.0	7.9250	0	0	1
3	1	0	35.0	53.1000	0	0	1
4	3	1	35.0	8.0500	0	0	1

In [23]:
```
from sklearn.model_selection import GridSearchCV #GridSearchCV is for parameter tuning
from sklearn.ensemble import RandomForestClassifier
cls=RandomForestClassifier()
n_estimators=[25,50,75,100,125,150,175,200] #number of decision trees in the forest, default = 100
criterion=['gini','entropy'] #criteria for choosing nodes default = 'gini'
max_depth=[3,5,10] #maximum number of nodes in a tree default = None (it will go till all possible nodes
parameters={'n_estimators': n_estimators,'criterion':criterion,'max_depth':max_depth} #this will underg
RFC_cls = GridSearchCV(cls, parameters)
RFC_cls.fit(x_train,y_train)
```

Out[23]:
```
GridSearchCV(estimator=RandomForestClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [3, 5, 10],
                         'n_estimators': [25, 50, 75, 100, 125, 150, 175, 200]})
```

In [24]:
```
RFC_cls.best_params_
```

Out[24]:
```
{'criterion': 'gini', 'max_depth': 5, 'n_estimators': 200}
```

In [25]:
```
cls=RandomForestClassifier(n_estimators=175,criterion='entropy',max_depth=10)
```

In [26]:
```
cls.fit(x_train,y_train)
```

Out[26]:
```
RandomForestClassifier(criterion='entropy', max_depth=10, n_estimators=175)
```

In [27]:
```
rfy_pred=cls.predict(x_test)
```

In [28]:
```
rfy_pred
```

Out[28]:
```
array([0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 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, 1, 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, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 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, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 1, 1, 0])
```

In [29]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,rfy_pred)
```

Out[29]:
```
array([[149,  26],
       [ 35,  85]])
```

In [30]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,rfy_pred)#EFFICENCY OF THE CONFUSION MATRIX
```

Out[30]:
```
0.7932203389830509
```

In [ ]:
```

```