

```
In [148]: import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

```
In [149]: data=pd.read_csv("/home/placement/Desktop/prasanna/Titanic Dataset.csv")
```

```
In [150]: data.describe()
```

Out[150]:

	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 [151]: data.dtypes
```

```
Out[151]: 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 [152]: data.isna().sum()
```

```
Out[152]: 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 [153]: data=data.drop(['Name','PassengerId','Ticket','Cabin','SibSp','Parch'],axis=1)
```

```
In [154]: data
```

```
Out[154]:
```

	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 [155]: data['Sex']=data['Sex'].map({'male':1,'female':0})
```

```
In [156]: data
```

```
Out[156]:
```

	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 [157]: data=data.fillna(data.median())
```

```
In [158]: data
```

```
Out[158]:
```

	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	28.0	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 [159]: data.isna().sum()
```

```
Out[159]: Survived    0
Pclass      0
Sex         0
Age         0
Fare        0
Embarked    2
dtype: int64
```

```
In [160]: data['Pclass']=data['Pclass'].map({1:'F',2:'S',3:'T'})
```

```
In [161]: data
```

```
Out[161]:
```

	Survived	Pclass	Sex	Age	Fare	Embarked
0	0	T	1	22.0	7.2500	S
1	1	F	0	38.0	71.2833	C
2	1	T	0	26.0	7.9250	S
3	1	F	0	35.0	53.1000	S
4	0	T	1	35.0	8.0500	S
...
886	0	S	1	27.0	13.0000	S
887	1	F	0	19.0	30.0000	S
888	0	T	0	28.0	23.4500	S
889	1	F	1	26.0	30.0000	C
890	0	T	1	32.0	7.7500	Q

891 rows × 6 columns

```
In [162]: data=pd.get_dummies(data)
```

In [163]: data

Out[163]:

	Survived	Sex	Age	Fare	Pclass_F	Pclass_S	Pclass_T	Embarked_C	Embarked_Q	Embarked_S
0	0	1	22.0	7.2500	0	0	1	0	0	1
1	1	0	38.0	71.2833	1	0	0	1	0	0
2	1	0	26.0	7.9250	0	0	1	0	0	1
3	1	0	35.0	53.1000	1	0	0	0	0	1
4	0	1	35.0	8.0500	0	0	1	0	0	1
...
886	0	1	27.0	13.0000	0	1	0	0	0	1
887	1	0	19.0	30.0000	1	0	0	0	0	1
888	0	0	28.0	23.4500	0	0	1	0	0	1
889	1	1	26.0	30.0000	1	0	0	1	0	0
890	0	1	32.0	7.7500	0	0	1	0	1	0

891 rows × 10 columns

```
In [164]: y=data['Survived']#predicted value removed from data frame
x=data.drop(['Survived'],axis=1)
```

In [165]:

y

Out[165]:

```

0      0
1      1
2      1
3      1
4      0
...
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64

```

In [166]:

x

Out[166]:

	Sex	Age	Fare	Pclass_F	Pclass_S	Pclass_T	Embarked_C	Embarked_Q	Embarked_S
0	1	22.0	7.2500	0	0	1	0	0	1
1	0	38.0	71.2833	1	0	0	1	0	0
2	0	26.0	7.9250	0	0	1	0	0	1
3	0	35.0	53.1000	1	0	0	0	0	1
4	1	35.0	8.0500	0	0	1	0	0	1
...
886	1	27.0	13.0000	0	1	0	0	0	1
887	0	19.0	30.0000	1	0	0	0	0	1
888	0	28.0	23.4500	0	0	1	0	0	1
889	1	26.0	30.0000	1	0	0	1	0	0
890	1	32.0	7.7500	0	0	1	0	1	0

891 rows × 9 columns


```
In [167]: 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 [168]: x_test.head(5)
```

Out[168]:

	Sex	Age	Fare	Pclass_F	Pclass_S	Pclass_T	Embarked_C	Embarked_Q	Embarked_S
709	1	28.0	15.2458	0	0	1	1	0	0
439	1	31.0	10.5000	0	1	0	0	0	1
840	1	20.0	7.9250	0	0	1	0	0	1
720	0	6.0	33.0000	0	1	0	0	0	1
39	0	14.0	11.2417	0	0	1	1	0	0

```
In [169]: y_test.head(5)
```

Out[169]:

709	1
439	0
840	0
720	1
39	1

Name: Survived, dtype: int64

```
In [170]: x_train.head(5)
```

Out[170]:

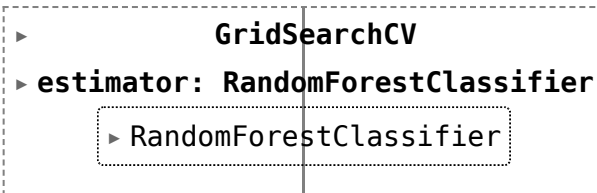
	Sex	Age	Fare	Pclass_F	Pclass_S	Pclass_T	Embarked_C	Embarked_Q	Embarked_S
6	1	54.0	51.8625	1	0	0	0	0	1
718	1	28.0	15.5000	0	0	1	0	1	0
685	1	25.0	41.5792	0	1	0	1	0	0
73	1	26.0	14.4542	0	0	1	1	0	0
882	0	22.0	10.5167	0	0	1	0	0	1

```
In [171]: y_train.head(5)
```

```
Out[171]: 6      0
          718    0
          685    0
          73     0
          882    0
          Name: Survived, dtype: int64
```

```
In [180]: 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 undergo 8*2
          RFC_cls = GridSearchCV(cls, parameters)
          RFC_cls.fit(x_train,y_train)
```

```
Out[180]:
```



```
  ►      GridSearchCV
  ► estimator: RandomForestClassifier
        ► RandomForestClassifier
```

```
In [182]: RFC_cls.best_params_
```

```
Out[182]: {'criterion': 'entropy', 'max_depth': 5, 'n_estimators': 150}
```

```
In [183]: cls=RandomForestClassifier(n_estimators=150,criterion='entropy',max_depth=5)
```

```
In [184]: cls.fit(x_train,y_train)
```

```
Out[184]: RandomForestClassifier
RandomForestClassifier(criterion='entropy', max_depth=5, n_estimators=150)
```

```
In [185]: rfy_pred=cls.predict(x_test)
```

```
In [186]: rfy_pred
```

```
Out[186]: array([0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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, 1, 0, 1, 1, 0, 0, 1,
                0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
                1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0,
                0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
                0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
                0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
                1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 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, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0,
                0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
                1, 0, 1, 0, 0, 0, 1, 1, 0])
```

```
In [187]: from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,rfy_pred)
```

```
Out[187]: array([[158, 17],
                [ 41, 79]])
```

```
In [188]: from sklearn.metrics import accuracy_score
accuracy_score(y_test,rfy_pred)
```

```
Out[188]: 0.8033898305084746
```

In []: