```
import pandas as pd
heart_dis=pd.read_excel("heart_disease.xlsx")
```

```
heart_dis
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

```
heart_dis.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

```
heart_dis.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1

```
y_dep=heart_dis.target
```

```
x_ind=heart_dis.drop("target",axis=1)
```

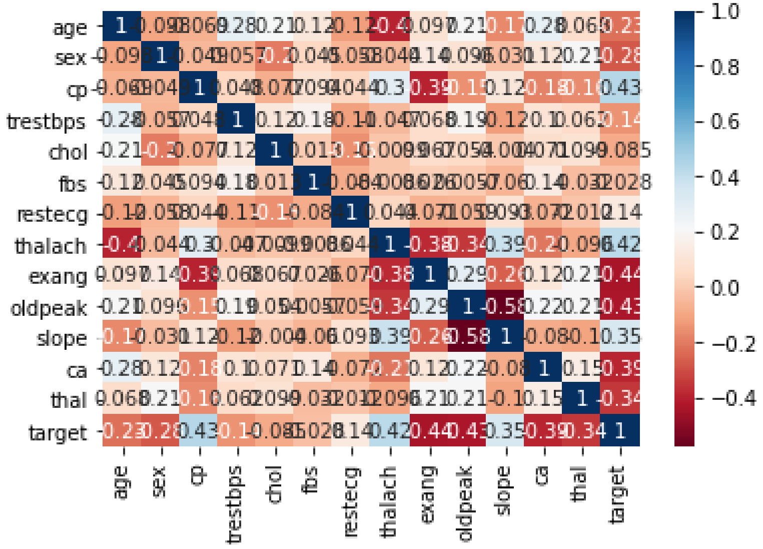
```
h_corr=heart_dis.corr()
h_corr
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.096801	(
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.141664	(
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.394280	-(
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	(
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	(
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	(
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-(
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-(
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	(
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-(
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.115739	(
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.206754	(
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-0.436757	-(

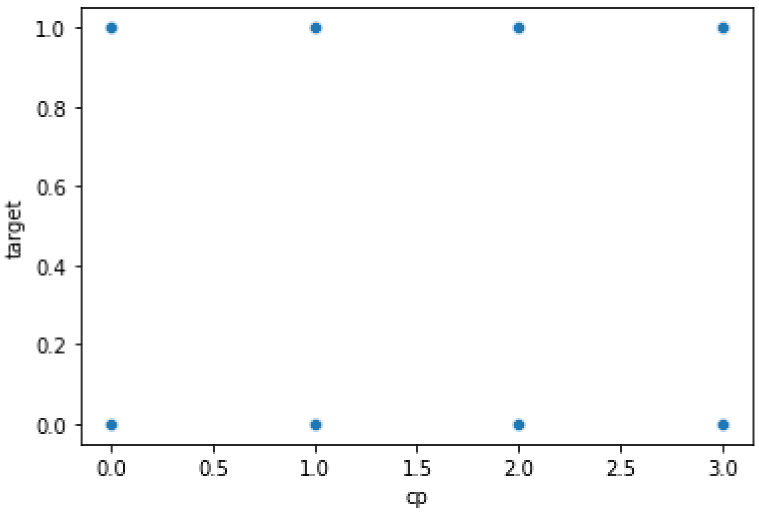
```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
sns.heatmap(h_corr,annot=True,cmap="RdBu")
```

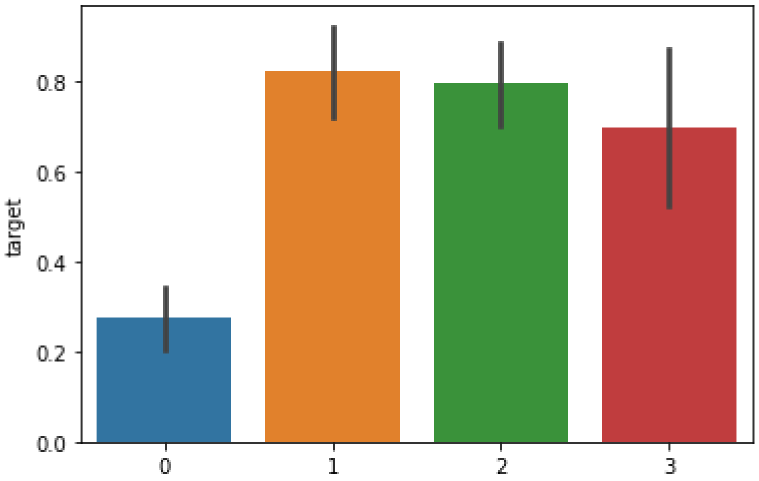
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdb2f373250>
```



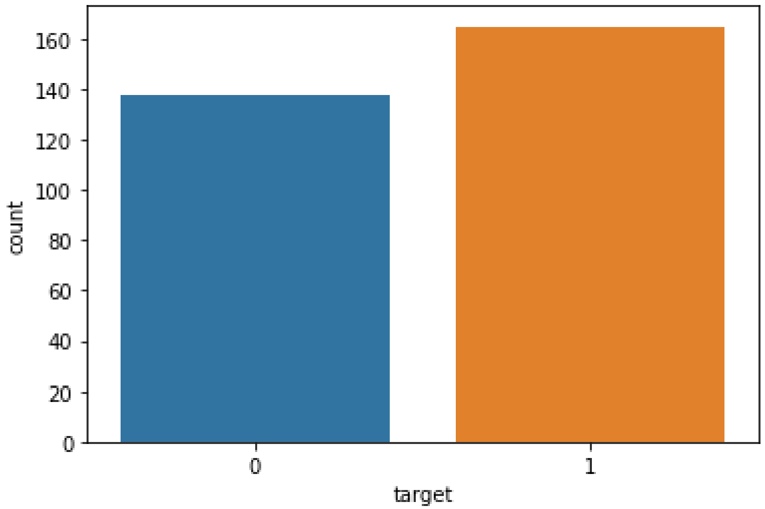
```
Psns.scatterplot(x="cp",y="target",data=heart_dis);
```



```
sns.barplot(x="cp",y="target",data=heart_dis);
```



```
sns.countplot(x="target",data=heart_dis);
```



```
import sklearn
from sklearn import model_selection
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x_ind, y_dep, test_size=0.3, random_state=2)
```

```
from sklearn import tree
model1=tree.DecisionTreeClassifier()
```

```
model1.fit(x_train,y_train)
y_pred=model1.predict(x_test)
y_pred
```

```
     array([1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1,
            0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0,
            1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1,
            1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1,
            1, 0, 0])
```

```
from sklearn.metrics import accuracy_score,confusion_matrix
```

```
confusion_matrix(y_test,y_pred)
```

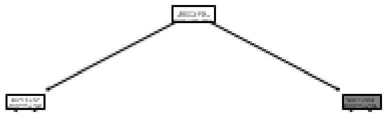
```
     array([[32,  9],
            [ 3, 47]])
```

```
accuracy_score(y_test,y_pred)
```

```
     0.8681318681318682
```

```
tree.plot_tree(model1,max_depth=3)
```

```
[Text(167.4, 195.696, 'X[2] <= 0.5\ngini = 0.496\nsamples = 212\nvalue = [97, 115]'),
 Text(83.7, 152.208, 'X[11] <= 0.5\ngini = 0.411\nsamples = 104\nvalue = [74, 30]'),
 Text(41.85, 108.72, 'X[12] <= 2.5\ngini = 0.498\nsamples = 47\nvalue = [22, 25]'),
 Text(20.925, 65.232, 'X[6] <= 0.5\ngini = 0.366\nsamples = 29\nvalue = [7, 22]'),
 Text(10.4625, 21.744, '\n  (...)  \n'),
 Text(31.387500000000003, 21.744, '\n  (...)  \n'),
 Text(62.775000000000006, 65.232, 'X[9] <= 0.45\ngini = 0.278\nsamples = 18\nvalue = [15, 3]'),
 Text(52.3125, 21.744, '\n  (...)  \n'),
 Text(73.2375, 21.744, '\n  (...)  \n'),
 Text(125.55000000000001, 108.72, 'X[9] <= 0.45\ngini = 0.16\nsamples = 57\nvalue = [52, 5]'),
 Text(104.625, 65.232, 'X[1] <= 0.5\ngini = 0.391\nsamples = 15\nvalue = [11, 4]'),
 Text(94.16250000000001, 21.744, '\n  (...)  \n'),
 Text(115.0875, 21.744, '\n  (...)  \n'),
 Text(146.475, 65.232, 'X[4] <= 301.0\ngini = 0.046\nsamples = 42\nvalue = [41, 1]'),
 Text(136.01250000000002, 21.744, '\n  (...)  \n'),
 Text(156.9375, 21.744, '\n  (...)  \n'),
 Text(251.10000000000002, 152.208, 'X[0] <= 56.5\ngini = 0.335\nsamples = 108\nvalue = [23, 85]'),
 Text(209.25, 108.72, 'X[12] <= 2.5\ngini = 0.172\nsamples = 63\nvalue = [6, 57]'),
 Text(188.32500000000002, 65.232, 'X[9] <= 3.55\ngini = 0.075\nsamples = 51\nvalue = [2, 49]'),
 Text(177.8625, 21.744, '\n  (...)  \n'),
 Text(198.7875, 21.744, '\n  (...)  \n'),
 Text(230.175, 65.232, 'X[11] <= 0.5\ngini = 0.444\nsamples = 12\nvalue = [4, 8]'),
 Text(219.7125, 21.744, '\n  (...)  \n'),
 Text(240.63750000000002, 21.744, '\n  (...)  \n'),
 Text(292.95, 108.72, 'X[1] <= 0.5\ngini = 0.47\nsamples = 45\nvalue = [17, 28]'),
 Text(272.02500000000003, 65.232, 'X[0] <= 57.5\ngini = 0.208\nsamples = 17\nvalue = [2, 15]'),
 Text(261.5625, 21.744, '\n  (...)  \n'),
 Text(282.4875, 21.744, '\n  (...)  \n'),
 Text(313.875, 65.232, 'X[4] <= 245.5\ngini = 0.497\nsamples = 28\nvalue = [15, 13]'),
 Text(303.4125, 21.744, '\n  (...)  \n'),
 Text(324.33750000000003, 21.744, '\n  (...)  \n')]
```



```
import graphviz
from sklearn.tree import export_graphviz
from six import StringIO
import IPython
from IPython.display import Image
```

```
import pydotplus
```

```
my_graph=StringIO()
```

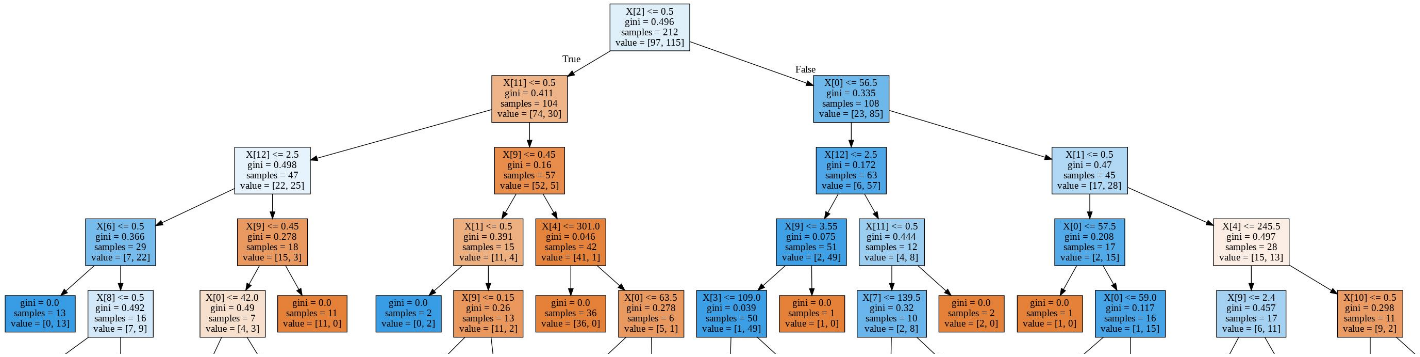
```
export_graphviz(model1,out_file=my_graph,filled=True)
```

```
my_graph=pydotplus.graph_from_dot_data(my_graph.getvalue())
```

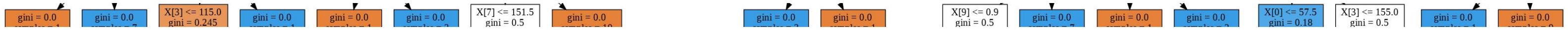
```
my_graph.write_jpg("DT.jpg")
```

```
     True
```

```
Image(my_graph.create_jpg())
```



```
model_e=tree.DecisionTreeClassifier(criterion="entropy",random_state=2)
```



```
model_e.fit(x_train,y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       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=2, splitter='best')
```

```
y_pred_e=model_e.predict(x_test)
```

```
confusion_matrix(y_test,y_pred_e)
```

```
array([[28, 13],
       [ 5, 45]])
```

```
accuracy_score(y_test,y_pred_e)
```

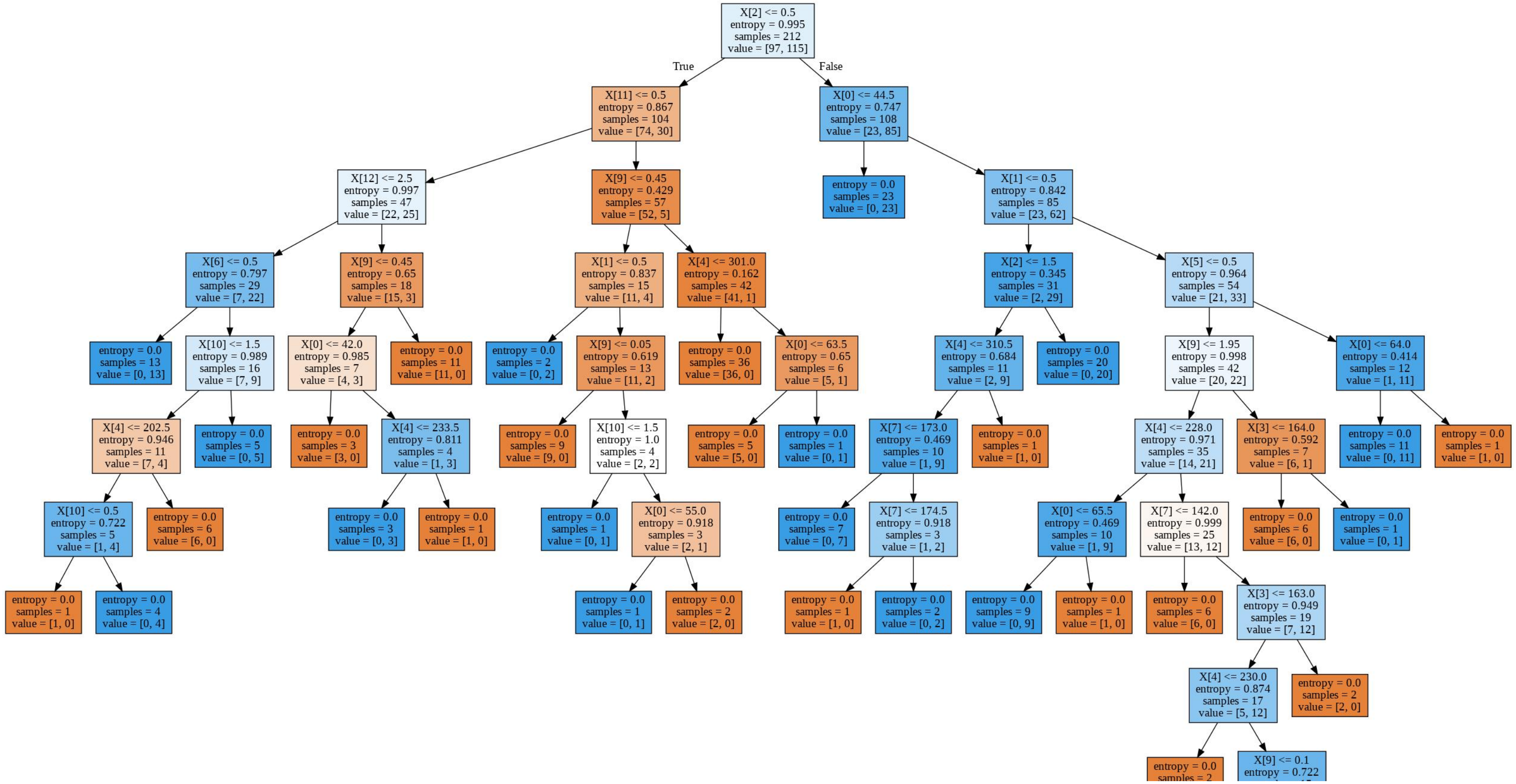
```
0.8021978021978022
```

```
my_graph=StringIO()
```

```
export_graphviz(model_e,out_file=my_graph,filled=True )
export_graphviz(model_e,out_file=my_graph,filled=True,)


my_graph=pydotplus.graph_from_dot_data(my_graph.getvalue())


Image(my_graph.create_jpg())
```



```
from sklearn.model_selection import RandomizedSearchCV
```

```
parameters={"max_depth":(10,20,30,40,50,60,70,100),'criterion':('gini','entropy'),
            'max_features':('log2','auto','sqrt'),'min_samples_split':(2,4,6)}
```

```
DT_hp=RandomizedSearchCV(tree.DecisionTreeClassifier(),param_distributions=parameters,cv=5)
```

```
DT_hp.fit(x_train,y_train)
```

```
RandomizedSearchCV(cv=5, error_score=nan,
                   estimator=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=None,
                                                    splitter='best'),
                   iid='deprecated', n_iter=10, n_jobs=None,
                   param_distributions={'criterion': ('gini', 'entropy'),
                                        'max_depth': (10, 20, 30, 40, 50, 60,
                                                      70, 100),
                                        'max_features': ('log2', 'auto',
                                                         'sqrt'),
                                        'min_samples_split': (2, 4, 6)},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=False, scoring=None, verbose=0)
```

```
DT_hp.best_estimator_
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=40, max_features='sqrt', 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=None, splitter='best')
```

```
model_after_Ht=tree.DecisionTreeClassifier(criterion='entropy', max_depth=40, max_features='sqrt')
```

```
model_after_Ht=model_after_Ht.fit(x_train,y_train)
```

```
pred_after_hp=model_after_Ht.predict(x_test)
```

```
confusion_matrix(y_test,pred_after_hp)
```

```
array([[28, 13],
       [ 8, 42]])
```

```
accuracy_score(y_test,pred_after_hp)
```

```
0.7692307692307693
```

finally got the good fit accuracy of 76%