HEART DISEASE PREDICITION PROJECT USING DECISION TREE

#IMPORTING

In [131]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [325]:
```
mydata=pd.read_excel("/HeartDisease.csv.xlsx")
```

In [326]:
```
mydata
```

Out[326]:

	age	gender	chest_pain	rest_bps	cholestrol	fasting_blood_sugar	rest_ecg	thalach	exer_angina	old_peak	slope	ca	thalassemi
0	63	1	3	145	233	1	0	150	0	2.3	0	0	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
298	57	0	0	140	241	0	1	123	1	0.2	1	0	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	

303 rows × 14 columns

In [286]:
```
mydata.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   gender               303 non-null    int64  
 2   chest_pain           303 non-null    int64  
 3   rest_bps             303 non-null    int64  
 4   cholestrol           303 non-null    int64  
 5   fasting_blood_sugar  303 non-null    int64  
 6   rest_ecg             303 non-null    int64  
 7   thalach              303 non-null    int64  
 8   exer_angina          303 non-null    int64  
 9   old_peak             303 non-null    float64
 10  slope                303 non-null    int64  
 11  ca                   303 non-null    int64  
 12  thalassemia          303 non-null    int64  
 13  target               303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

In [287]:
```
mydata.describe()
```

Out[287]:

	age	gender	chest_pain	rest_bps	cholestrol	fasting_blood_sugar	rest_ecg	thalach	exer_angina	old_peak
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000

In [327]:
```
mydata.isnull().sum()
```

Out[327]:
```
age                   0
gender                0
chest_pain            0
rest_bps              0
cholestrol            0
fasting_blood_sugar   0
rest_ecg              0
thalach               0
exer_angina           0
old_peak              0
slope                 0
ca                    0
thalassemia           0
target                0
dtype: int64
```

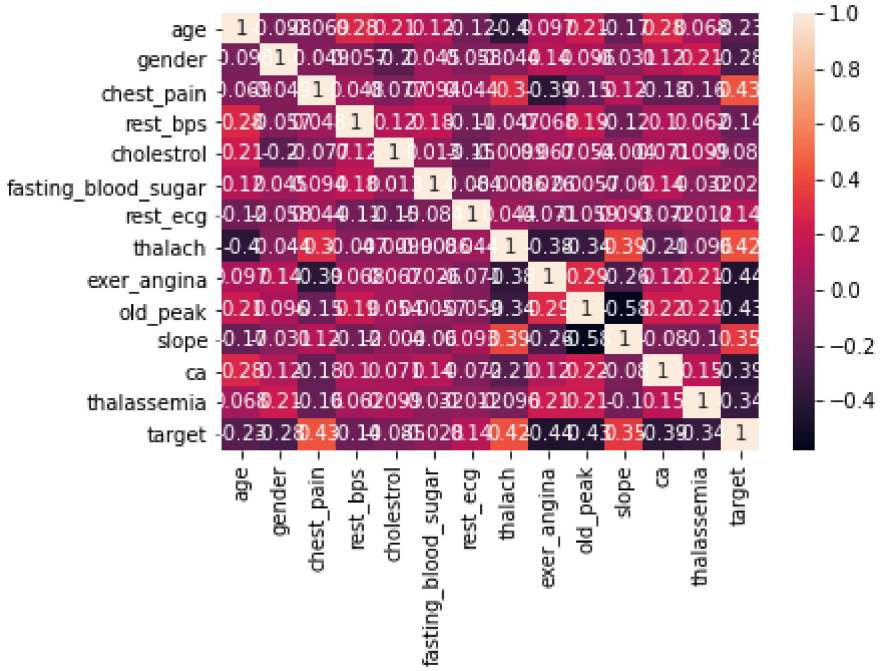
FINDING THE **CORRELATION* *

In [328]:
```
mydata_corr=mydata.corr()
mydata_corr
```

Out[328]:

	age	gender	chest_pain	rest_bps	cholestrol	fasting_blood_sugar	rest_ecg	thalach	exer_angina	old_
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.096801	0.21
gender	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.141664	0.09
chest_pain	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.394280	-0.14
rest_bps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	0.19
cholestrol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	0.05
fasting_blood_sugar	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.00
rest_ecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-0.05
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-0.34
exer_angina	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	0.28
old_peak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1.00
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-0.57
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.115739	0.22
thalassemia	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.206754	0.21
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-0.436757	-0.43

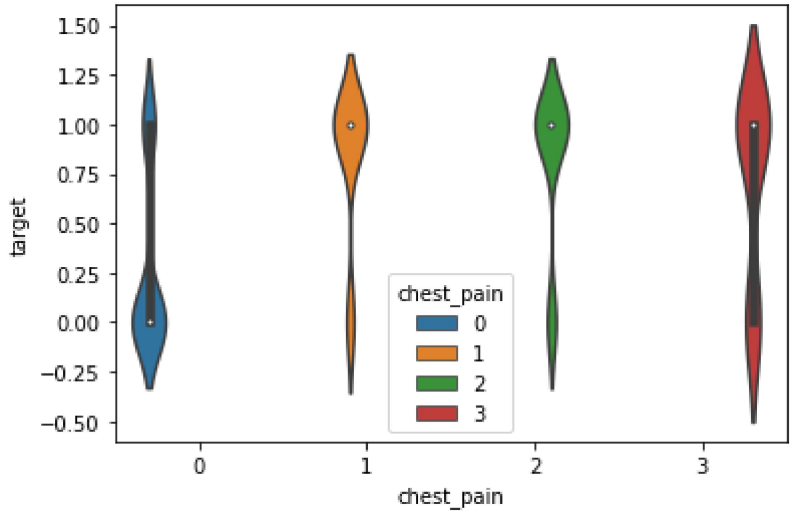
In [138]:
```
sns.heatmap(mydata_corr,annot=True);
plt.figure(figsize=(16,9));
```



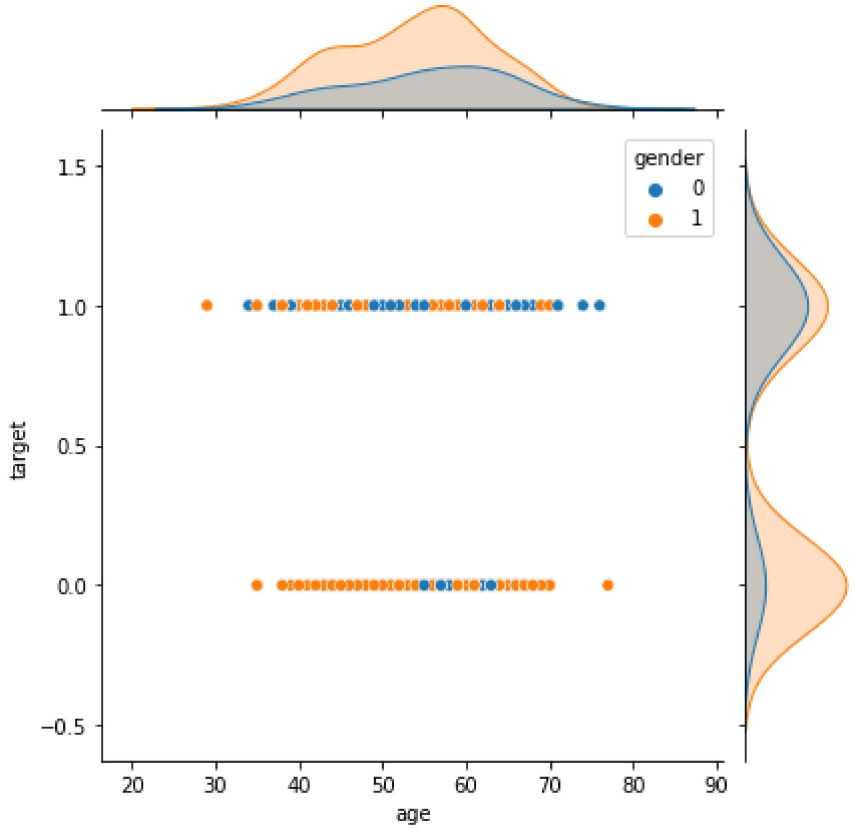
<Figure size 1152x648 with 0 Axes>

DATA VISUALIZATION

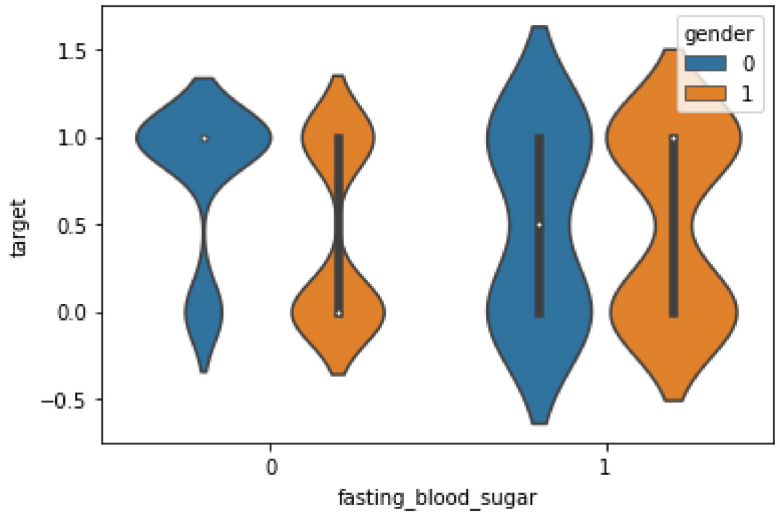
In [139]:
```
sns.violinplot(x='chest_pain',y='target',hue="chest_pain",data=mydata);
```



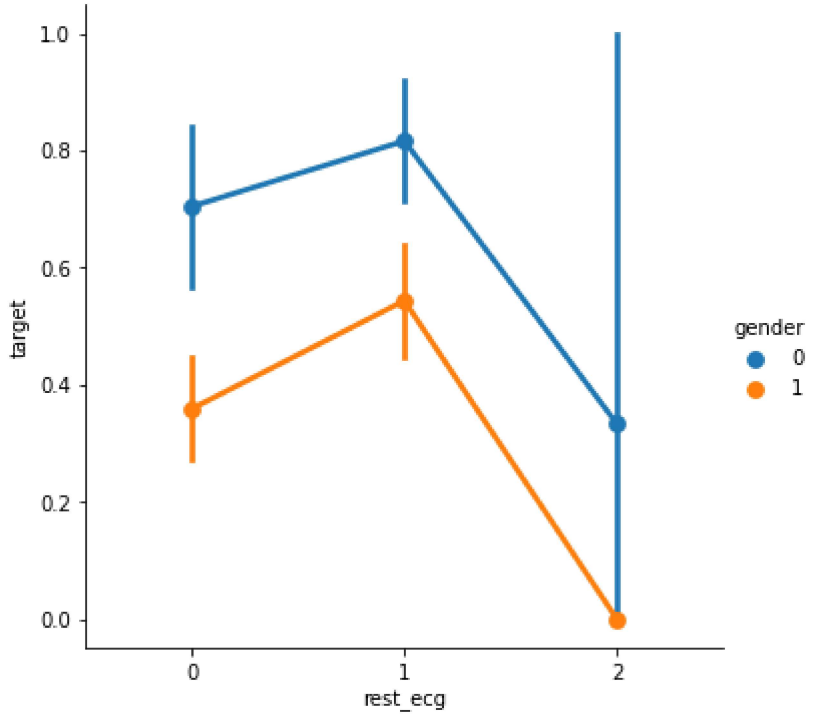
In [140]:
```
sns.jointplot(x="age",y='target',hue="gender",data=mydata);
```



In [141]:
```
sns.violinplot(x="fasting_blood_sugar",y='target',hue='gender',kind='point',data=mydata);
```

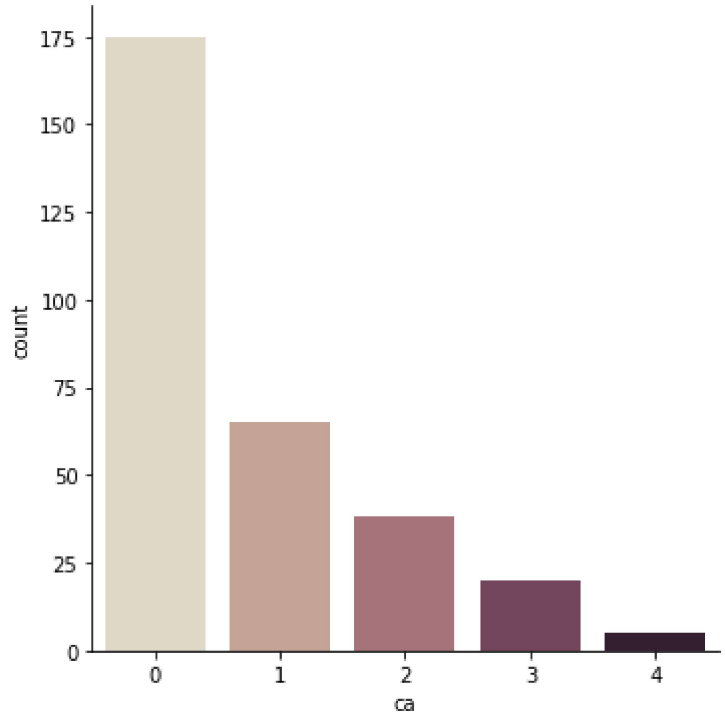


In [142]:
```
sns.catplot(x="rest_ecg",y='target',hue='gender',kind='point',data=mydata);
```

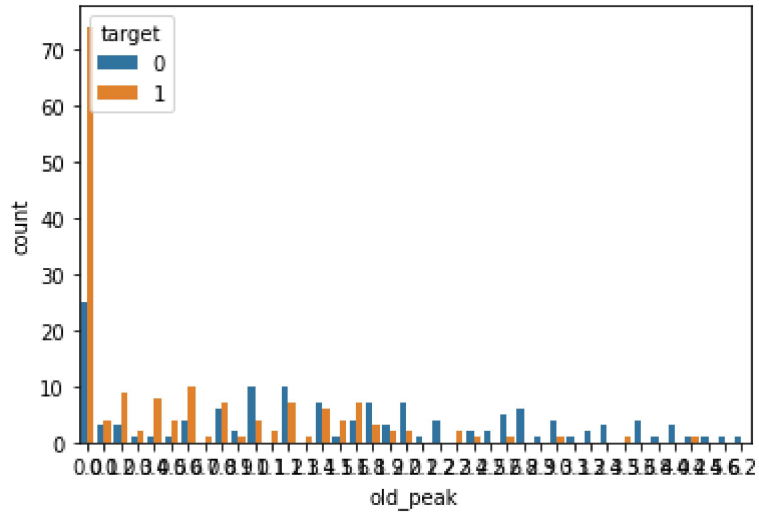


In [143]:
```
sns.catplot(x='ca',kind='count',palette='ch:.29',data=mydata)
```

Out[143]: <seaborn.axisgrid.FacetGrid at 0x7ffa142c89d0>



In [144]:
```
sns.countplot(x="old_peak",hue="target",data=mydata);
plt.figure(figsize=(18,5))
plt.show()
```




```
<Figure size 1296x360 with 0 Axes>
```

SPLITING THE VARIABLES

In [329]:
```
y_dep=mydata.target
y_dep
```

Out[329]:
```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

In [330]:
```
x_ind=mydata.drop("target",axis=1)
x_ind
```

Out[330]:

	age	gender	chest_pain	rest_bps	cholestrol	fasting_blood_sugar	rest_ecg	thalach	exer_angina	old_peak	slope	ca	thalassemi
0	63	1	3	145	233	1	0	150	0	2.3	0	0	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
298	57	0	0	140	241	0	1	123	1	0.2	1	0	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	

303 rows × 13 columns

In [331]:
```
from sklearn.model_selection import train_test_split
```

In [332]:
```
x_train,x_test,y_train,y_test=train_test_split(x_ind,y_dep,train_size=0.8,random_state=2)
```

DECISION TREE

In [333]:
```
from sklearn import tree
```

In [334]:
```
model=tree.DecisionTreeClassifier(random_state=2)
model=model.fit(x_train,y_train)
```

In [335]:
```
# PREDICTION
```

In [336]:
```
y_pred=model.predict(x_test)
```

In [337]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [338]:
```
confusion_matrix(y_test,y_pred)
```

Out[338]:
```
array([[25,  7],
       [ 1, 28]])
```

In [339]:
```
# ACCURACY
```

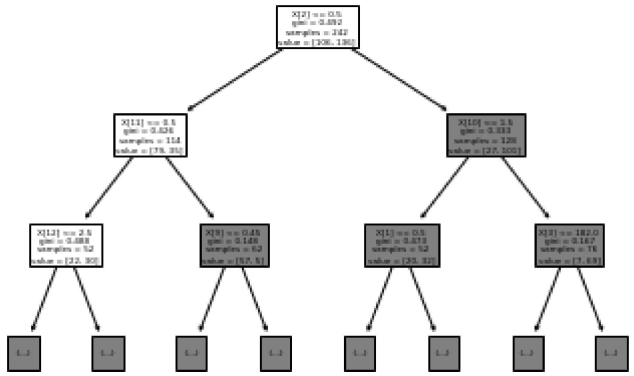
In [340]:
```
accuracy_score(y_test,y_pred)
```

Out[340]:
```
0.8688524590163934
```

PLOTING THE DECISION TREE

In [341]:
```
tree.plot_tree(model,max_depth=2)
```

Out[341]:
```
[Text(167.4, 190.26, 'X[2] <= 0.5\ngini = 0.492\nsamples = 242\nvalue = [106, 136]'),
 Text(83.7, 135.9, 'X[11] <= 0.5\ngini = 0.426\nsamples = 114\nvalue = [79, 35]'),
 Text(41.85, 81.53999999999999, 'X[12] <= 2.5\ngini = 0.488\nsamples = 52\nvalue = [22, 30]'),
 Text(20.925, 27.180000000000007, '\n  (...)  \n'),
 Text(62.775000000000006, 27.180000000000007, '\n  (...)  \n'),
 Text(125.55000000000001, 81.53999999999999, 'X[9] <= 0.45\ngini = 0.148\nsamples = 62\nvalue = [57, 5]'),
 Text(104.625, 27.180000000000007, '\n  (...)  \n'),
 Text(146.475, 27.180000000000007, '\n  (...)  \n'),
 Text(251.10000000000002, 135.9, 'X[10] <= 1.5\ngini = 0.333\nsamples = 128\nvalue = [27, 101]'),
 Text(209.25, 81.53999999999999, 'X[1] <= 0.5\ngini = 0.473\nsamples = 52\nvalue = [20, 32]'),
 Text(188.32500000000002, 27.180000000000007, '\n  (...)  \n'),
 Text(230.175, 27.180000000000007, '\n  (...)  \n'),
 Text(292.95, 81.53999999999999, 'X[3] <= 182.0\ngini = 0.167\nsamples = 76\nvalue = [7, 69]'),
 Text(272.02500000000003, 27.180000000000007, '\n  (...)  \n'),
 Text(313.875, 27.180000000000007, '\n  (...)  \n')]
```



GRAPHICAL REPRESENTATION FOR IMPURITIES MODEL

In [342]:
```
import graphviz
from sklearn.externals.six import StringIO
from sklearn.tree import export_graphviz
import IPython
from IPython.display import Image
import pydotplus
import warnings
warnings.filterwarnings('ignore')
```

In [343]:
```
my_graph=StringIO()
```

In [344]:
```
export_graphviz(model,out_file=my_graph,filled=True,feature_names=['age', 'gender', 'chest_pain', 'rest_bps', 'c
       'fasting_blood_sugar', 'rest_ecg', 'thalach', 'exer_angina', 'old_peak',
       'slope', 'ca', 'thalassemia'],class_names=np.array(['HD negative','HD positive']))
```

In [345]:
```
graph=pydotplus.graph_from_dot_data(my_graph.getvalue())
```

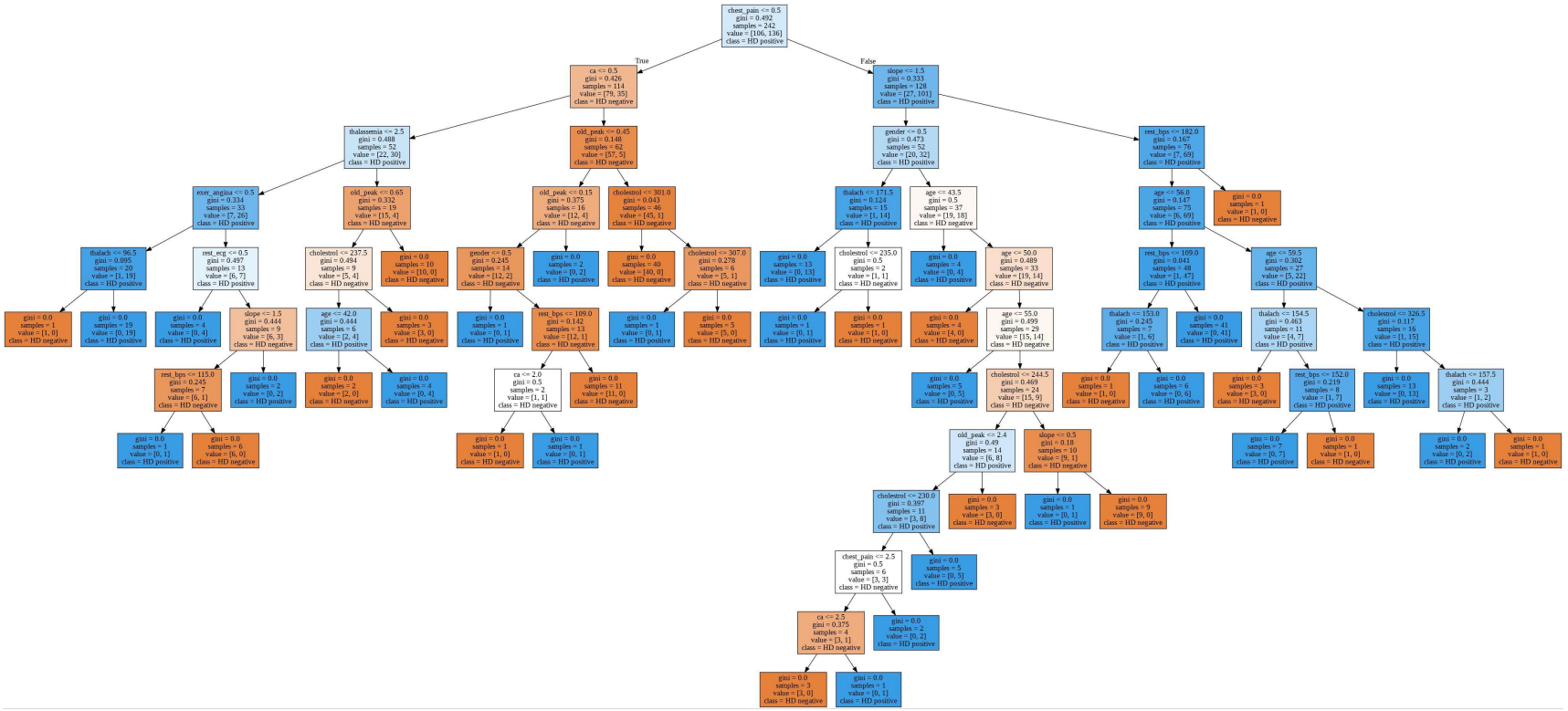
In [346]:
```
# CREATING & SAVING
```

In [347]:
```
graph.write_jpg("Heart_tree_model1.jpg")
```

Out[347]:
```
True
```

In [348]:
```
Image(graph.create_jpg())
```

Out[348]:



ENTROPY MODEL

In [254]:
```
model_E=tree.DecisionTreeClassifier(criterion="entropy",random_state=2)
```

In [255]:
```
model_E=model_E.fit(x_train,y_train)
```

In [256]:
```
# PREDICTION
```

In [257]:
```
y_pred_E=model_E.predict(x_test)
y_pred_E
```

Out[257]:
```
array([1, 1, 1, 1, 0, 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, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0,
       1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1])
```

In [258]:
```
# CONFUSION MATRIX
```

In [259]:
```
from sklearn.metrics import confusion_matrix
```

In [260]:
```
confusion_matrix(y_test,y_pred_E)
```

Out[260]:
```
array([[22, 10],
       [ 4, 25]])
```

In [261]:
```
# ACCURACY
```

In [283]:
```
accuracy_score(y_test,y_pred_E)
```

Out[283]:
```
0.7704918032786885
```

***WE GOT ENTROPY ACCURACY OF 81% ***

In [263]:
```
# GRAPHICAL REPRESENTATION
```

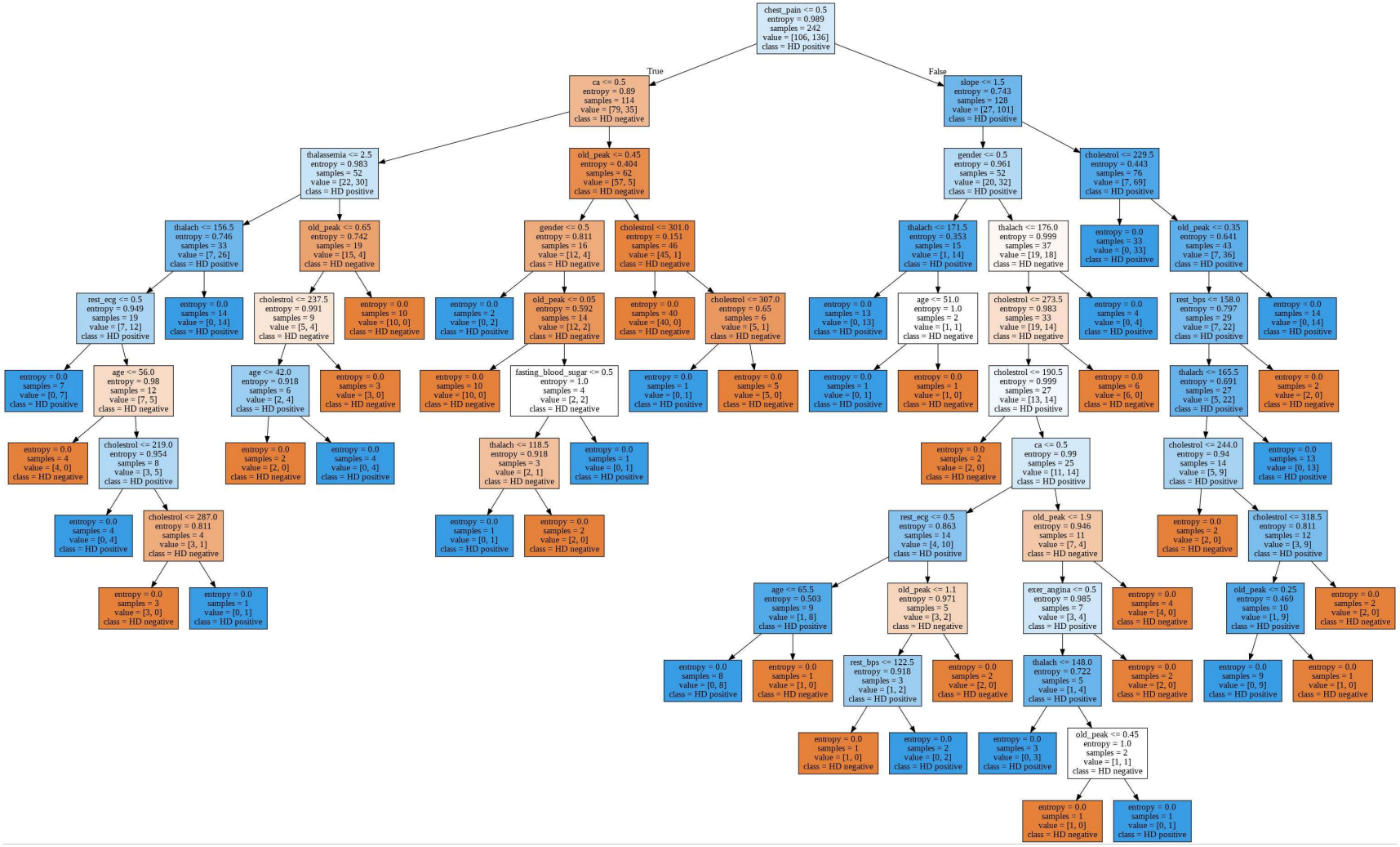
In [264]:
```
import graphviz
from sklearn.externals.six import StringIO
from sklearn.tree import export_graphviz
import IPython
from IPython.display import Image
import pydotplus
```

In [265]:
```
my_graph_E=StringIO()
```

In [266]:
```
export_graphviz(model_E,out_file=my_graph_E,filled=True,feature_names=['age', 'gender', 'chest_pain', 'rest_bps'
       'fasting_blood_sugar', 'rest_ecg', 'thalach', 'exer_angina', 'old_peak',
       'slope', 'ca', 'thalassemia'],class_names=np.array(['HD negative','HD positive']))
```

In [267]:
```
graph1=pydotplus.graph_from_dot_data(my_graph_E.getvalue())
graph1.write_jpg("Heart_disease_Tree_model2.jpg")
Image(graph1.create_jpg())
```

Out[267]:



HYPER-PARAMETER TUNING

In [349]:
```
from sklearn.model_selection import RandomizedSearchCV
```

In [350]:
```
parameters={"max_depth":(10,20,30,40,50,60,70,100),'criterion':('gini','entropy'),'max_features':('log2','auto',
            'min_samples_split':(2,4,6)}
```

In [351]:
```
parameters
```

Out[351]:
```
{'criterion': ('gini', 'entropy'),
 'max_depth': (10, 20, 30, 40, 50, 60, 70, 100),
 'max_features': ('log2', 'auto', 'sqrt'),
 'min_samples_split': (2, 4, 6)}
```

In [352]:
```
dT_hp=RandomizedSearchCV(tree.DecisionTreeClassifier(),param_distributions=parameters,cv=5)
```

In [353]:
```
dT_hp
```

Out[353]:
```
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)
```

In [354]:
```
dT_hp.fit(x_train,y_train)
```

Out[354]:
```
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)
```

In [355]:
```
dT_hp.best_estimator_
```

Out[355]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=40, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=4,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

In [372]:
```
model_after_ht=tree.DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=40, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=4,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

In [373]:
```
model_after_Ht=model_after_Ht.fit(x_train,y_train)
```

In [374]:
```
# PREDICTION
```

In [375]:
```
y_pred_after_hp=model_after_Ht.predict(x_test)
y_pred_after_hp
```

Out[375]:
```
array([1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0,
       1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0])
```

In [376]:
```
# CONFUSION MATRIX
```

In [377]:
```
confusion_matrix(y_test,y_pred_after_hp)
```

Out[377]:
```
array([[24,  8],
       [ 4, 25]])
```

In [378]:
```
# ACCURACY
```

In [379]:
```
accuracy_score(y_test,y_pred_after_hp)
```

Out[379]:
```
0.8032786885245902
```

GRAPHICAL REPRESENTATION FOR ESTIMATER

In [194]:
```
import graphviz
from sklearn.externals.six import StringIO
from sklearn.tree import export_graphviz
import IPython
from IPython.display import Image
import pydotplus
```

In [195]:
```
my_graph_B=StringIO()
```

In [196]:
```
export_graphviz(model_after_Ht,out_file=my_graph_B,filled=True,feature_names=['age', 'gender', 'chest_pain', 're
       'fasting_blood_sugar', 'rest_ecg', 'thalach', 'exer_angina', 'old_peak',
       'slope', 'ca', 'thalassemia'],class_names=np.array(['HD negative','HD positive']))
```

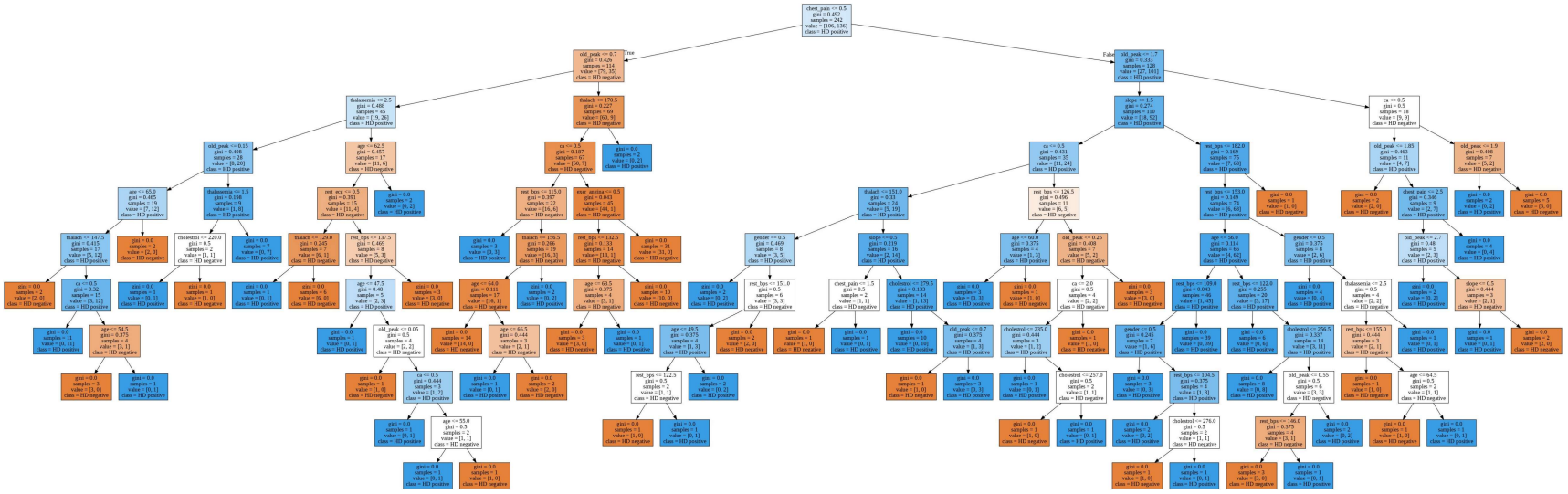
In [197]:
```
graph2=pydotplus.graph_from_dot_data(my_graph_B.getvalue())
```

In [198]:
```
graph2.write_jpg("Heart_disease_Tree_model3.jpg")
```

Out[198]: True

In [199]:
```
Image(graph2.create_jpg())
```

Out[199]:



In [199]:

WE CONCLUDE THAT....BY USING THE ESTIMATER (HYPERPARAMETER TUNING MODEL) WITH THE ACCURACY OF 80% FOR THE ABOVE DATASET(HEART_DISEASE)

THEREFORE IT IS CALLED TO BE THE BEST MODEL.

In [ ]: