۱) وارد کردن و پیش پردازش داده ها

ابتدا داده ها را وارد کرده و ستون ها را نام گذاری می کنیم:

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

df =  pd.read_csv("F:\\IUST\\داده کاوی\\2 ترم\\HW\\HW4\\processed.cleveland.csv", sep=',', header=None,
                  names=['age', 'sex', 'cp', 'restbp', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak','slope', 'ca', 'th

df
```

Out[1]:

	age	sex	cp	restbp	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	hd
0	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	0
1	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	2
2	67.0	1.0	4.0	120.0	229.0	0.0	2.0	129.0	1.0	2.6	2.0	2.0	7.0	1
3	37.0	1.0	3.0	130.0	250.0	0.0	0.0	187.0	0.0	3.5	3.0	0.0	3.0	0
4	41.0	0.0	2.0	130.0	204.0	0.0	2.0	172.0	0.0	1.4	1.0	0.0	3.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	45.0	1.0	1.0	110.0	264.0	0.0	0.0	132.0	0.0	1.2	2.0	0.0	7.0	1
299	68.0	1.0	4.0	144.0	193.0	1.0	0.0	141.0	0.0	3.4	2.0	2.0	7.0	2
300	57.0	1.0	4.0	130.0	131.0	0.0	0.0	115.0	1.0	1.2	2.0	1.0	7.0	3
301	57.0	0.0	2.0	130.0	236.0	0.0	2.0	174.0	0.0	0.0	2.0	1.0	3.0	1
302	38.0	1.0	3.0	138.0	175.0	0.0	0.0	173.0	0.0	0.0	1.0	?	3.0	0

303 rows × 14 columns

سپس برای پیش پردازش داده ها مقادیر null را با مد همان ستون جایگزین می کنیم.که با دستور isnull در می یابیم که مقادیر nullدر ستون های ca و thalهستند که مد آنها به ترتیب ۰و ۳ است.

وهمچنین ۳ ستون 'restecg', 'cp' , 'restecg’ را به string تبدیل می کنیم زیرا برای onehot encoding نیاز است که این ستون ها onehot شوند. (بدلیل آنکه دارای چندین مقدار خاص هستند)

In [3]:
```
print(df.ca.mode())
print(df.thal.mode())

df['ca'].replace('?','0',inplace=True)
df['thal'].replace('?','3',inplace=True)

df['cp']=df['cp'].apply(str)
df['restecg']=df['restecg'].apply(str)
df['slope']=df['slope'].apply(str)

df
```

```
0    0.0
dtype: object
0    3.0
dtype: object
```

Out[3]:

	age	sex	cp	restbp	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	hd
0	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	0
1	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	2
2	67.0	1.0	4.0	120.0	229.0	0.0	2.0	129.0	1.0	2.6	2.0	2.0	7.0	1
3	37.0	1.0	3.0	130.0	250.0	0.0	0.0	187.0	0.0	3.5	3.0	0.0	3.0	0
4	41.0	0.0	2.0	130.0	204.0	0.0	2.0	172.0	0.0	1.4	1.0	0.0	3.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	45.0	1.0	1.0	110.0	264.0	0.0	0.0	132.0	0.0	1.2	2.0	0.0	7.0	1
299	68.0	1.0	4.0	144.0	193.0	1.0	0.0	141.0	0.0	3.4	2.0	2.0	7.0	2
300	57.0	1.0	4.0	130.0	131.0	0.0	0.0	115.0	1.0	1.2	2.0	1.0	7.0	3
301	57.0	0.0	2.0	130.0	236.0	0.0	2.0	174.0	0.0	0.0	2.0	1.0	3.0	1
302	38.0	1.0	3.0	138.0	175.0	0.0	0.0	173.0	0.0	0.0	1.0	0	3.0	0

303 rows × 14 columns

ستون مشخص کننده دارای بیماری قلبی(hd) را بعنوان ستونی انتخاب می کنیم که می خواهیم پیش بینی کنیم (y) و سایر ستون ها را بعنوان ویژگی هایی برای انجام پیش بینی در نظر می گیریم(X).

In [5]:
```
#split dataset in features and target variable
feature_cols = ['age', 'sex', 'cp' , 'restbp', 'chol', 'fbs' ,'restecg' ,'thalach', 'exang' ,'oldpeak',
'slope', 'ca', 'thal']
X = df[feature_cols] # Features
y = df.hd # Target variable
X
```

Out[5]:

	age	sex	cp	restbp	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0
1	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0
2	67.0	1.0	4.0	120.0	229.0	0.0	2.0	129.0	1.0	2.6	2.0	2.0	7.0
3	37.0	1.0	3.0	130.0	250.0	0.0	0.0	187.0	0.0	3.5	3.0	0.0	3.0
4	41.0	0.0	2.0	130.0	204.0	0.0	2.0	172.0	0.0	1.4	1.0	0.0	3.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	45.0	1.0	1.0	110.0	264.0	0.0	0.0	132.0	0.0	1.2	2.0	0.0	7.0
299	68.0	1.0	4.0	144.0	193.0	1.0	0.0	141.0	0.0	3.4	2.0	2.0	7.0
300	57.0	1.0	4.0	130.0	131.0	0.0	0.0	115.0	1.0	1.2	2.0	1.0	7.0
301	57.0	0.0	2.0	130.0	236.0	0.0	2.0	174.0	0.0	0.0	2.0	1.0	3.0
302	38.0	1.0	3.0	138.0	175.0	0.0	0.0	173.0	0.0	0.0	1.0	0	3.0

303 rows × 13 columns

با استفاده از تابع get dummies() ستون های غیر باینری دارای اعداد خاص را one hot encoding می کنیم:

In [6]:
```
X=pd.get_dummies(X)
X
```

Out[6]:

	age	sex	restbp	chol	fbs	thalach	exang	oldpeak	cp_1.0	cp_2.0	...	slope_3.0	ca_0	ca_0.0	ca_1.0	ca_2.0	ca_3.0	thal_3	thal_3.0	thal_6.0	tha
0	63.0	1.0	145.0	233.0	1.0	150.0	0.0	2.3	1	0	...	1	0	1	0	0	0	0	0	1	
1	67.0	1.0	160.0	286.0	0.0	108.0	1.0	1.5	0	0	...	0	0	0	0	0	1	0	1	0	
2	67.0	1.0	120.0	229.0	0.0	129.0	1.0	2.6	0	0	...	0	0	0	0	1	0	0	0	0	
3	37.0	1.0	130.0	250.0	0.0	187.0	0.0	3.5	0	0	...	1	0	1	0	0	0	0	1	0	
4	41.0	0.0	130.0	204.0	0.0	172.0	0.0	1.4	0	1	...	0	0	1	0	0	0	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
298	45.0	1.0	110.0	264.0	0.0	132.0	0.0	1.2	1	0	...	0	0	1	0	0	0	0	0	0	
299	68.0	1.0	144.0	193.0	1.0	141.0	0.0	3.4	0	0	...	0	0	0	0	1	0	0	0	0	
300	57.0	1.0	130.0	131.0	0.0	115.0	1.0	1.2	0	0	...	0	0	0	1	0	0	0	0	0	
301	57.0	0.0	130.0	236.0	0.0	174.0	0.0	0.0	0	1	...	0	0	0	1	0	0	0	1	0	
302	38.0	1.0	138.0	175.0	0.0	173.0	0.0	0.0	0	0	...	0	1	0	0	0	0	0	1	0	

303 rows × 27 columns

سپس داده های test و train را برای هردو دسته ستون ویژگی ها(X) و ستون قابل پیش بینی(y) از هم جدا می کنیم و ۸۰ درصد داده ها را برای آموزش در نظر می گیریم که در ادامه ممکن است این عدد را تغییر دهیم:

In [7]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) # 80% training and 20% test
X_train
```

Out[7]:

	age	sex	restbp	chol	fbs	thalach	exang	oldpeak	cp_1.0	cp_2.0	...	slope_3.0	ca_0	ca_0.0	ca_1.0	ca_2.0	ca_3.0	thal_3	thal_3.0	thal_6.0	tha
132	29.0	1.0	130.0	204.0	0.0	202.0	0.0	0.0	0	1	...	0	0	1	0	0	0	0	1	0	
202	57.0	1.0	150.0	126.0	1.0	173.0	0.0	0.2	0	0	...	0	0	0	1	0	0	0	0	0	
196	69.0	1.0	160.0	234.0	1.0	131.0	0.0	0.1	1	0	...	0	0	0	1	0	0	0	1	0	
75	65.0	0.0	160.0	360.0	0.0	151.0	0.0	0.8	0	0	...	0	0	1	0	0	0	0	1	0	
176	52.0	1.0	108.0	233.0	1.0	147.0	0.0	0.1	0	0	...	0	0	0	0	0	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
188	54.0	1.0	192.0	283.0	0.0	195.0	0.0	0.0	0	1	...	0	0	0	1	0	0	0	0	0	
71	67.0	1.0	125.0	254.0	1.0	163.0	0.0	0.2	0	0	...	0	0	0	0	1	0	0	0	0	
106	59.0	1.0	140.0	177.0	0.0	162.0	1.0	0.0	0	0	...	0	0	0	1	0	0	0	0	0	
270	61.0	1.0	140.0	207.0	0.0	138.0	1.0	1.9	0	0	...	0	0	0	1	0	0	0	0	0	
102	57.0	0.0	128.0	303.0	0.0	159.0	0.0	0.0	0	0	...	0	0	0	1	0	0	0	1	0	

242 rows × 27 columns

سپس مقیاس بندی داده ها با استفاده از min و max انجام می دهیم:

In [8]:
```
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
# create a scaler object
scaler = preprocessing.MinMaxScaler()
# fit and transform the data
X_train_norm = pd.DataFrame(scaler.fit_transform(X_train.values), columns=X_train.columns,index=X_train.index)
X_test_norm = pd.DataFrame(scaler.fit_transform(X_test.values), columns=X_test.columns,index=X_test.index)

X_train_norm
```

Out[8]:

	age	sex	restbp	chol	fbs	thalach	exang	oldpeak	cp_1.0	cp_2.0	...	slope_3.0	ca_0	ca_0.0	ca_1.0	ca_2.0	ca_3.0	thal_3	thal_3.0
132	0.000000	1.0	0.339623	0.268041	0.0	1.000000	0.0	0.000000	0.0	1.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0
202	0.583333	1.0	0.528302	0.000000	1.0	0.778626	0.0	0.032258	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
196	0.833333	1.0	0.622642	0.371134	1.0	0.458015	0.0	0.016129	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
75	0.750000	0.0	0.622642	0.804124	0.0	0.610687	0.0	0.129032	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0
176	0.479167	1.0	0.132075	0.367698	1.0	0.580153	0.0	0.016129	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
188	0.520833	1.0	0.924528	0.539519	0.0	0.946565	0.0	0.000000	0.0	1.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
71	0.791667	1.0	0.292453	0.439863	1.0	0.702290	0.0	0.032258	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
106	0.625000	1.0	0.433962	0.175258	0.0	0.694656	1.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
270	0.666667	1.0	0.433962	0.278351	0.0	0.511450	1.0	0.306452	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
102	0.583333	0.0	0.320755	0.608247	0.0	0.671756	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0

242 rows × 27 columns

۲) ساخت درخت اولیه: درخت اولیه را با داده های آموزش fit می کنیم و ویژگی هدف را با سایر ویژگی ها در حال تست پیش بینی می کنیم

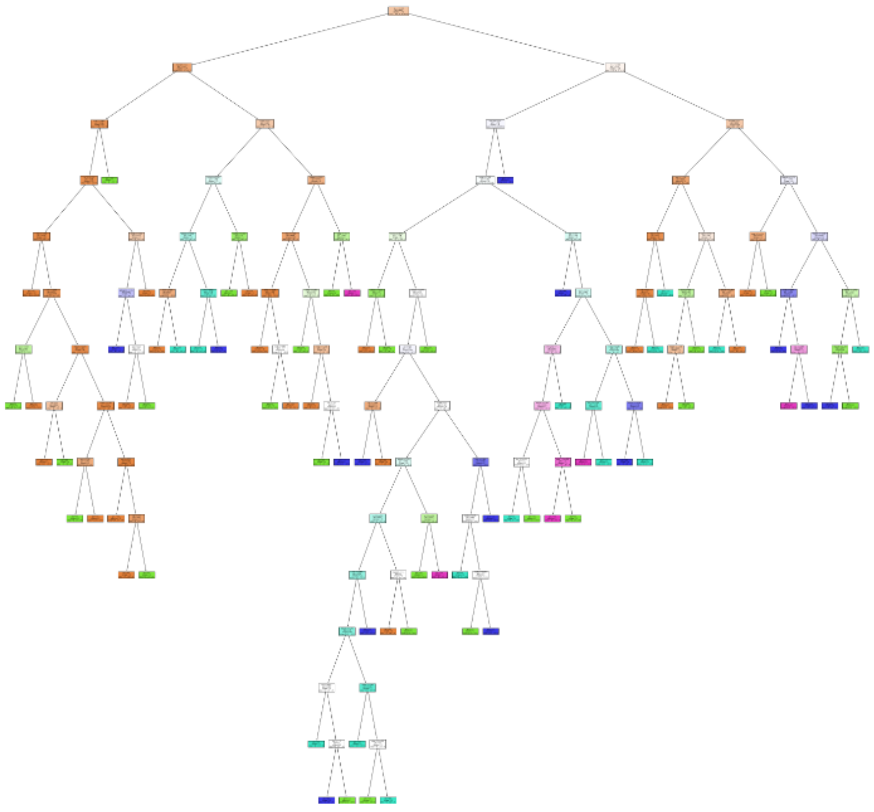
Build first Tree

In [8]:
```
from matplotlib import pyplot as plt
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

# Create Decision Tree classifer object
clf = DecisionTreeClassifier( random_state = 42)
# Train Decision Tree Classifer
clf = clf.fit(X_train_norm,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test_norm)
```

رسم ساختار درختی درخت اولیه:

۱- بوسیله pyplot

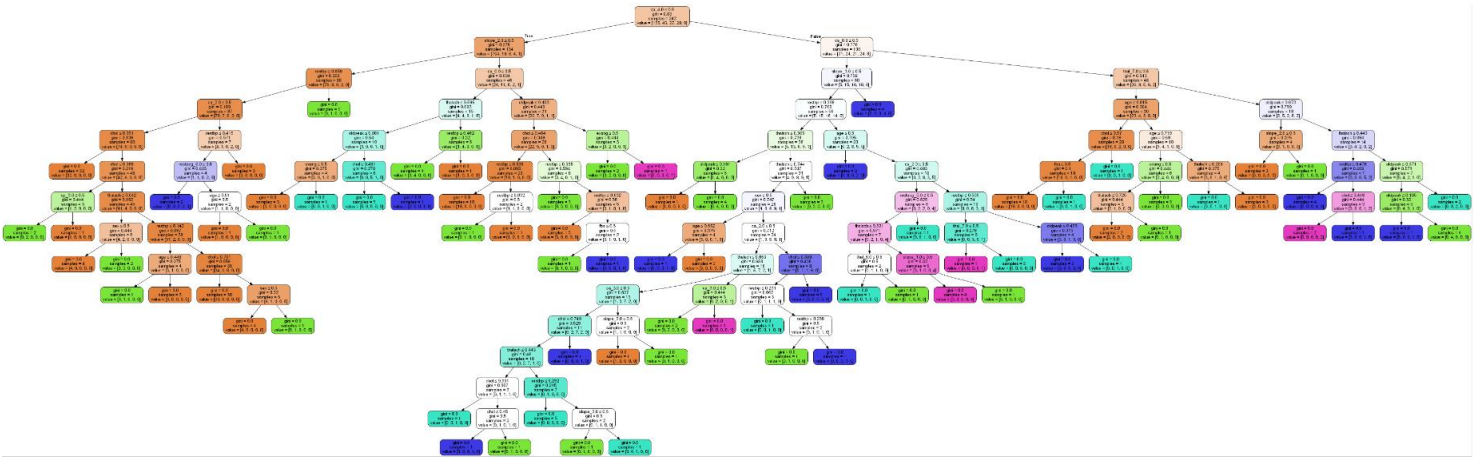
In [13]:
```
fig = plt.figure(figsize=(50,50))
_ = tree.plot_tree(clf, filled=True,feature_names=X_test_norm.columns)
plt.savefig('Desktop/tree.png')
```



۲-بوسیله Graphiz

(با توجه به بزرگ بودن درخت محتویات هر نود درخت در تصویر زیر قابل تشخیص نیست اما در کد که در پیوست گزارش قرار می گیرد قابل رویت است)

In [9]:
```
import graphviz
# DOT data

dot_data = tree.export_graphviz(clf, out_file=None,feature_names=X_test_norm.columns,
                                filled=True, rounded=True ,special_characters=True)

# Draw graph

graph = graphviz.Source(dot_data, format="png")
graph.render("decision_tree_graphivz")
graph
```



ساخت ماتریس درهم ریختگی:

In [16]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.45901639344262296
[[23  6  0  0  0]
 [ 4  2  5  1  0]
 [ 2  2  2  2  1]
 [ 1  3  3  0  0]
 [ 0  1  1  1  1]]
              precision    recall  f1-score   support

           0       0.77      0.79      0.78        29
           1       0.14      0.17      0.15        12
           2       0.18      0.22      0.20         9
           3       0.00      0.00      0.00         7
           4       0.50      0.25      0.33         4

    accuracy                           0.46        61
   macro avg       0.32      0.29      0.29        61
weighted avg       0.45      0.46      0.45        61
```

با توجه به اطلاعات بالا accuracy برابر ۰/۴۶ می باشد که قصد داریم آن را بالاتر ببریم. البته این درخت ابتدا با ۷۰ درصد داده آموزش داده شده بود که بعد از افزایش به ۸۰ درصد , accuracy مقداری افزایش یافت.

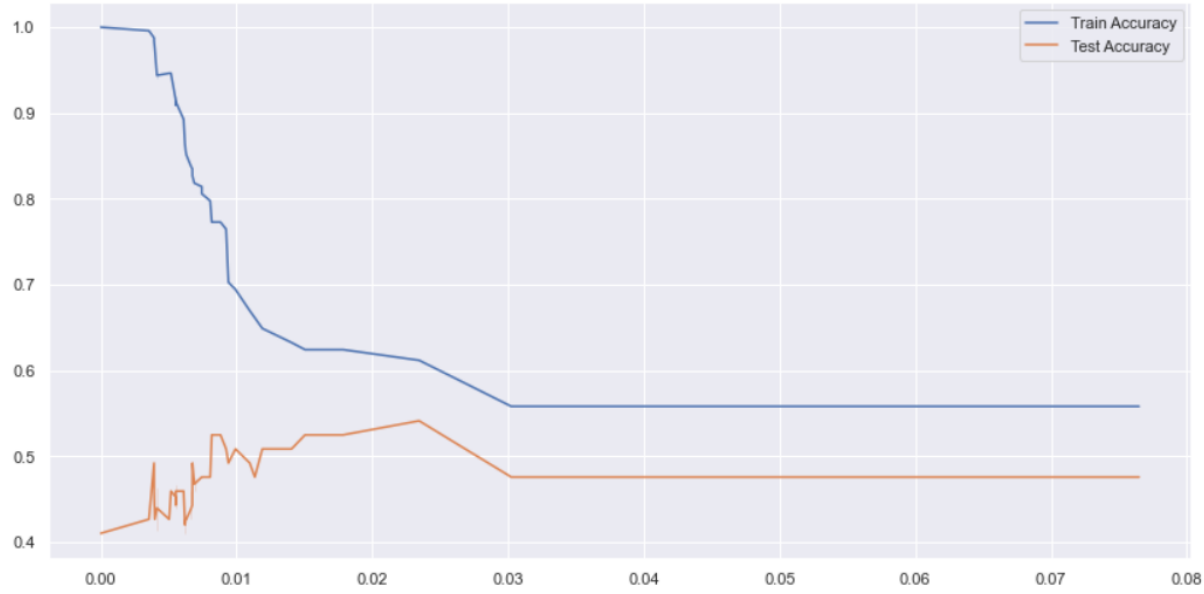
از روی هر درایه ماتریس در هم ریختگی نیز می توان میزان نزدیک بودن پیش بینی به عدد واقعی را دریافت و هرچه درایه های غیرقطری دارای صفر بیشتری باشند بهتر است و پیش بینی ما دقیق تر بوده که این ماتریس بعد از هرس بهبود می یابد.

۳) هرس کردن درخت تصمیم:

درخت تصمیم را هرس می کنیم و به ازای آلفاهای مختلف accuracy را اندازه می گیریم و در نمودار رسم می کنیم:

In [17]:
```
from sklearn.metrics import accuracy_score
import seaborn as sns
path=clf.cost_complexity_pruning_path(X_train_norm , y_train)
alphas=path['ccp_alphas']
print(alphas)
print(len(alphas))

accuracy_train,accuracy_test=[],[]
for i in alphas:
    tree=DecisionTreeClassifier(ccp_alpha=i)

    tree.fit(X_train_norm,y_train)
    y_train_pred=tree.predict(X_train_norm)
    y_test_pred=tree.predict(X_test_norm)

    accuracy_train.append(accuracy_score(y_train,y_train_pred))
    accuracy_test.append(accuracy_score(y_test,y_test_pred))


sns.set()
plt.figure(figsize=(14,7))
sns.lineplot(y=accuracy_train,x=alphas,label="Train Accuracy")
sns.lineplot(y=accuracy_test,x=alphas,label="Test Accuracy")
plt.show()
```

```
[0.         0.00354191 0.00392032 0.00392562 0.00397481 0.00413223
 0.00413223 0.00413223 0.00413223 0.00413223 0.00503381 0.00516529
 0.00550964 0.00550964 0.00550964 0.00550964 0.00550964 0.00550964
 0.00551927 0.00609892 0.00619835 0.00619835 0.00619835 0.00619835
 0.00619835 0.00625132 0.00629673 0.00661157 0.00661157 0.00661157
 0.00674024 0.00674931 0.00688705 0.00688705 0.00743802 0.00743802
 0.00805785 0.00817264 0.00881543 0.00922865 0.00939787 0.00991736
 0.01097009 0.01134396 0.01189794 0.01404959 0.0150441  0.01785336
 0.02345041 0.03023799 0.07648562]
51
```



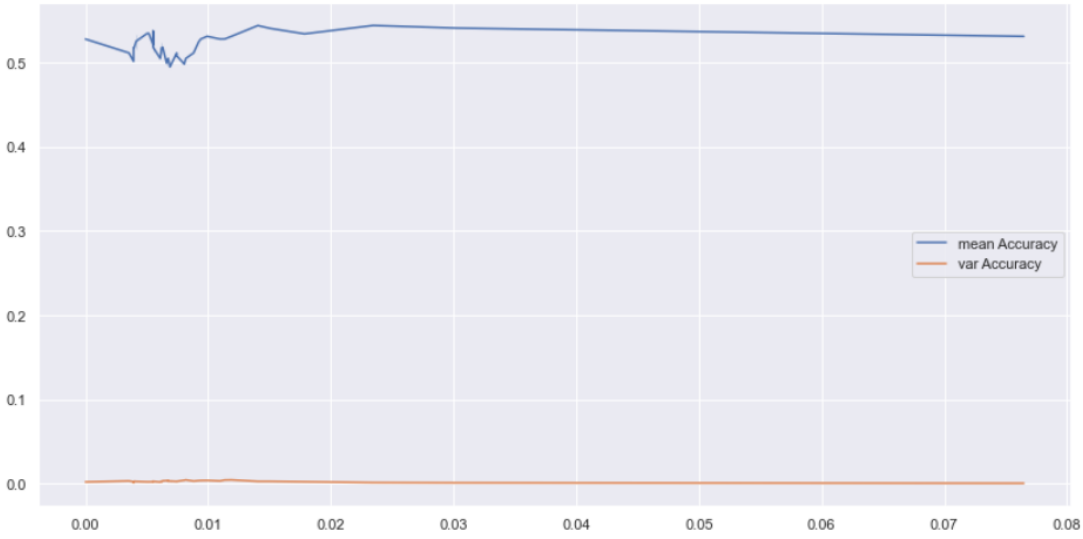
با توجه به نمودار بالا و test accuracy می توان دریافت که با یک آلفای بین ۰/۰۲ و ۰/۰۳ می توان به بهترین Accuracy رسید. که با توجه به لیست آلفاها در می یابیم که تنها آلفای این بازه ۰/۰۲۳۴۵۰۴۱ می باشد.

پس با توجه به نمودار بالا بهترین آلفا برابر است با: ۰/۰۲۳۴۵۰۴۱

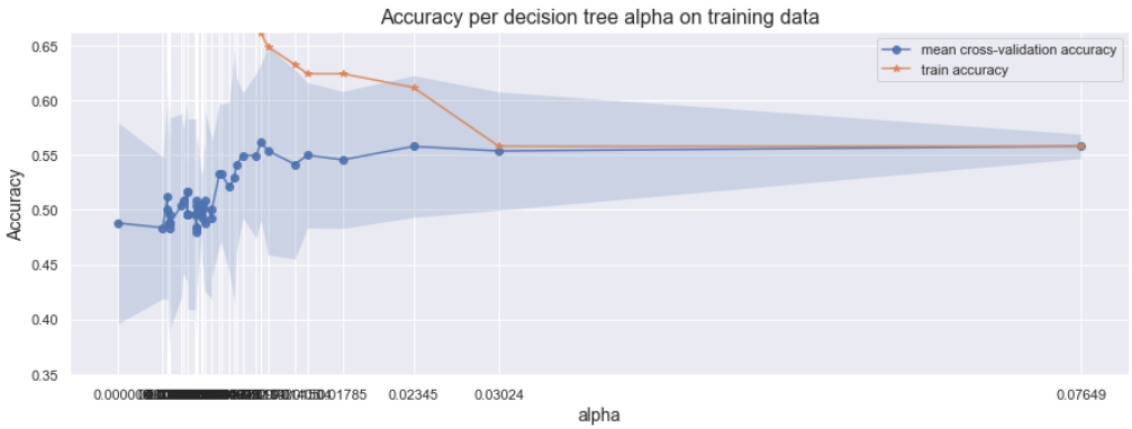
5-Fold cross validation : به ازای آلفاهای مختلف 5-fold cross validation را اجرامی کنیم و میانگین و واریانس Accuracy را محاسبه می کنیم و در نموداری نمایش می دهیم:

5_Fold

In [19]:
```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_score

mean_accuracy = []
var_accuracy = []
mean = []
var = []
for i in alphas:
    clf = DecisionTreeClassifier(ccp_alpha=i)
    scores = cross_val_score(estimator=clf, X=X, y=y, cv=5, scoring='accuracy')
    mean_accuracy.append((scores.mean()))
    var_accuracy.append((scores.var()))
    mean.append((i,scores.mean()))
    var.append((i,scores.var()))

print(mean)
print(var)


sns.set()
plt.figure(figsize=(14,7))
sns.lineplot(y=mean_accuracy,x=alphas,label="mean Accuracy")
sns.lineplot(y=var_accuracy,x=alphas,label="var Accuracy")
plt.show()
```



کد نمودار زیر در فایل کد بصورت دقیق قابل رویت است که بدلیل حجم بالا اینجا کپی نشده است:



میانگین Accuracy برای الفاهای مختلف:

```
[(0.0, 0.5281967213114754), (0.0035419126328217246, 0.5117486338797814), (0.003920322102140283, 0.5017486338797814), (0.0039256
19834710742, 0.5182513661202186), (0.003974813065722155, 0.5183060109289618), (0.004132231404958678, 0.5347540983606557), (0.00
4132231404958678, 0.518415300546448), (0.004132231404958678, 0.5347540983606558), (0.004132231404958678, 0.5315846994535518),
(0.004132231404958678, 0.5085245901639344), (0.005033809166040574, 0.5349180327868853), (0.005165289256198347, 0.53491803278688
53), (0.005509641873278236, 0.5150273224043715), (0.005509641873278236, 0.5117486338797814), (0.005509641873278236, 0.541530054
6448087), (0.005509641873278236, 0.5150819672131147), (0.005509641873278236, 0.5248633879781421), (0.005509641873278237, 0.5381
420765027322), (0.00551927411431543, 0.5183606557377048), (0.006098916757223966, 0.5050819672131148), (0.006198347107438017, 0.
5116939890710382), (0.006198347107438017, 0.5281420765027323), (0.006198347107438017, 0.5084153005464481), (0.00619834710743801
7, 0.5216393442622952), (0.006198347107438017, 0.5183060109289619), (0.0062513244331426215, 0.5183606557377048), (0.00629673356
9460843, 0.5184699453551913), (0.006611570247933882, 0.495191256830601 1), (0.006611570247933882, 0.5018032786885247), (0.006611
570247933882, 0.5018579234972678), (0.006740243702341192, 0.5051912568306011), (0.006749311294765841, 0.5051912568306011), (0.0
06887052341597796, 0.49513661202185794), (0.006887052341597796, 0.4951366120218579 4), (0.007438016528925621, 0.511639344262295
1), (0.007438016528925622, 0.508360655737705), (0.008057851239669421, 0.49841530054644806), (0.008172635445362718, 0.5050819672
131148), (0.008815426997245177, 0.5117486338797814), (0.009228650137741 05, 0.5249180327868853), (0.009397874852420302, 0.528196
7213114754), (0.009917355371900829, 0.5314754098360656), (0.01097009051554506, 0.5281967213114754), (0.011343959071231813, 0.52
82513661202186), (0.0118979383275571, 0.5314207650273224), (0.014049586776859503, 0.5442622950819672), (0.015044104180914064,
0.5409289617486339), (0.017853355637128088, 0.5343715846994536), (0.023450413223140495, 0.5443715846994536), (0.030237985919804
145, 0.5412568306010929), (0.07648562317086727, 0.5314207650273224)]
```

واریانس برای آلفاهای مختلف:

```
[(0.0, 0.00196883752874078), (0.0035419126328217246, 0.0031041834632267317), (0.003920322102140283, 0.0021908268386634415), (0.
0039256198347 10742, 0.001026080205440593), (0.003974813065722155, 0.0027282391232942146), (0.004132231404958678, 0.002412416017
1996776), (0.004132231404958678, 0.0031325390426707274), (0.004132231404958678, 0.002685222013198365), (0.004132231404958678,
0.0021104720953148796), (0.004132231404958678, 0.002261232046343576), (0.005033809166040574, 0.002066302367941712), (0.00516528
9256198347, 0.0020735286213383495), (0.005509641873278236, 0.00236417928274956), (0.005509641873278236, 0.002021977365702171),
(0.005509641873278236, 0.0019569410851324313), (0.005509641873278236, 0.0031814147929170775), (0.005509641873278236, 0.00132682
37331661146), (0.005509641873278237, 0.0028357251634865 19), (0.00551927411431543, 0.00264177491116486), (0.006098916757223966,
0.00180446116635313), (0.006198347107438017, 0.0021790856699214663), (0.006198347107438017, 0.0025505091223983994), (0.00619834
7107438017, 0.0025675117202663563), (0.006198347107438017, 0.003273952641165756), (0.006198347107438017, 0.0036523634626295 21),
(0.0062513244331426215, 0.0024231658156409564), (0.006296733569460843, 0.00320053151781182), (0.006611570247933882, 0.003249353
5190659603), (0.006611570247933882, 0.0038694974469228 71), (0.006611570247933882, 0.0032762519036101403), (0.00674024370234119
2, 0.0037455283824539393), (0.006749311294765841, 0.0027780465227388103), (0.006887052341597796, 0.0025446683985786 38), (0.0068
87052341597796, 0.0034046522738809767), (0.007438016528925621, 0.0025547970975544217), (0.007438016528925622, 0.002620370868046
225), (0.008057851239669421, 0.0039632834662121787), (0.008172635445362718, 0.0043844127922601 45), (0.008815426997245177, 0.0029
25318761384335), (0.00922865013774105, 0.0034871569769177957), (0.009397874852420302, 0.0036355041954074484), (0.00991735537190
0829, 0.0037623518170145 42), (0.01097009051554506, 0.0030980142733434864), (0.011343959071231813, 0.004250756964973572), (0.011
8979383275571, 0.0043985786377616 52), (0.014049586776859503, 0.002628594463853802), (0.015044104180914064, 0.0026347875421780277), (0.017853355637128088, 0.0021610797575323 23), (0.023450413223140495, 0.0012069813968765858), (0.030237985919804145, 0.00099
5371614560005), (0.07648562317086727, 0.00040205739795156704)]
```

در نمودار بالا که با استفاده از cross validation رسم شده می توان دریافت که میانگین accuracy در آلفایی بین ۰/۰۲ و ۰/۰۳ روی بیشـــترین مقدار قرار دارد و با توجه به این که در لیســـت آلفاهای ما فقط عدد ۰/۰۲۳۴۵۰۴۱ در این بازه قرار دارد ,این آلفا بهترین آلفای ما خواهد بود.

از روی زوج مرتب های آلفا و میانگین Accuracy هم می توان دریافت که این آلفا بیشترین میانگین Accuracyرا دارد.

بهترین آلفا با استفاده از5fold-cross validation برابر است با: ۰/۰۲۳۴۵۰۴۱

۴) ساخت و ارزیابی و رسم درخت تصمیم با آلفای مشخص:

با توجه به نمودارهایی که تحلیل شد بهترین آلفا ۰/۰۲۳۴۵۰۴۱ بود که می خواهیم با این آلفا درخت را هرس کرده و درخت را رسم کنیم و برای آن Accuracy و confusion matrix را هم حساب می کنیم:

Build final Tree

In [20]:
```
from matplotlib import pyplot as plt
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
# Create Decision Tree classifer object
clf2 = DecisionTreeClassifier(ccp_alpha=0.023450413223140495, random_state = 42)
# Train Decision Tree Classifer
clf2 = clf2.fit(X_train_norm,y_train)

#Predict the response for test dataset
y_test_pred = clf2.predict(X_test_norm)
print(accuracy_score(y_test,y_test_pred))
print(confusion_matrix(y_test, y_test_pred))
```

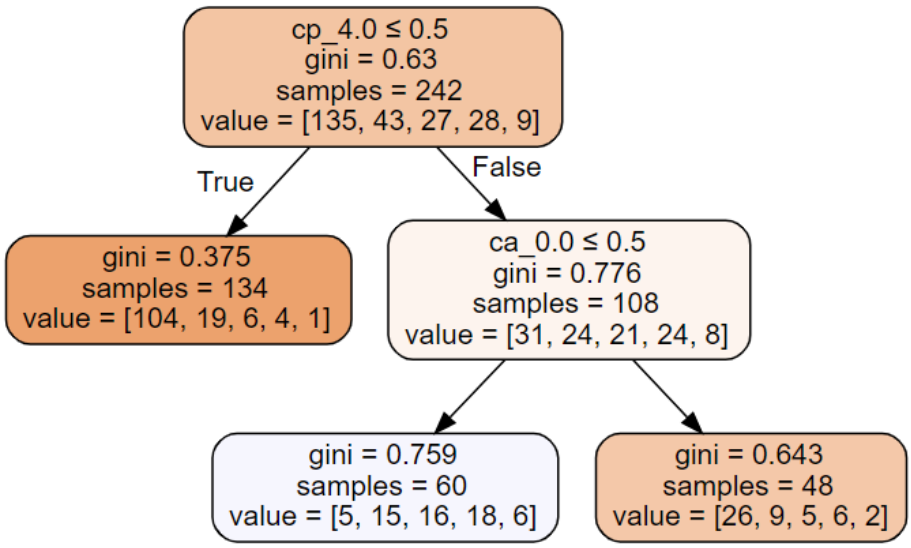
```
0.5409836065573771
[[29  0  0  0  0]
 [ 6  0  0  6  0]
 [ 4  0  0  5  0]
 [ 3  0  0  4  0]
 [ 1  0  0  3  0]]
```

با توجه به Accuracy و confusion matrix می توان دریافت که درخت بهتری داریم که accuracy از ۴۶ به ۵۴ رسیده و در ماتریس هم تعداد صفرهای غیرقطری افزایش یافته و پیش بینی های بهتری داشته ایم.

رسم ساختار درخت پایانی بعد از هرس:

In [30]:
```
import graphviz

dot2_data = tree.export_graphviz(clf2, out_file=None,feature_names=X_test_norm.columns,
                                 filled=True, rounded=True ,special_characters=True)

# Draw graph
graph = graphviz.Source(dot2_data, format="png")
graph
```

Out[30]:



حالت دوم تحلیل:

این دیتاست را می توان در حالتی که ستون برچسب فقط صفر و یک باشد هم بررسی کرد که تمام مراحل و کدها مانند حالت قبل می باشد فقط در این حالت hd های غیر صفر را برابر ۱ قرار می دهیم و فقط وجود یا عدم وجود بیماری قلبی را بررسی می کنیم.

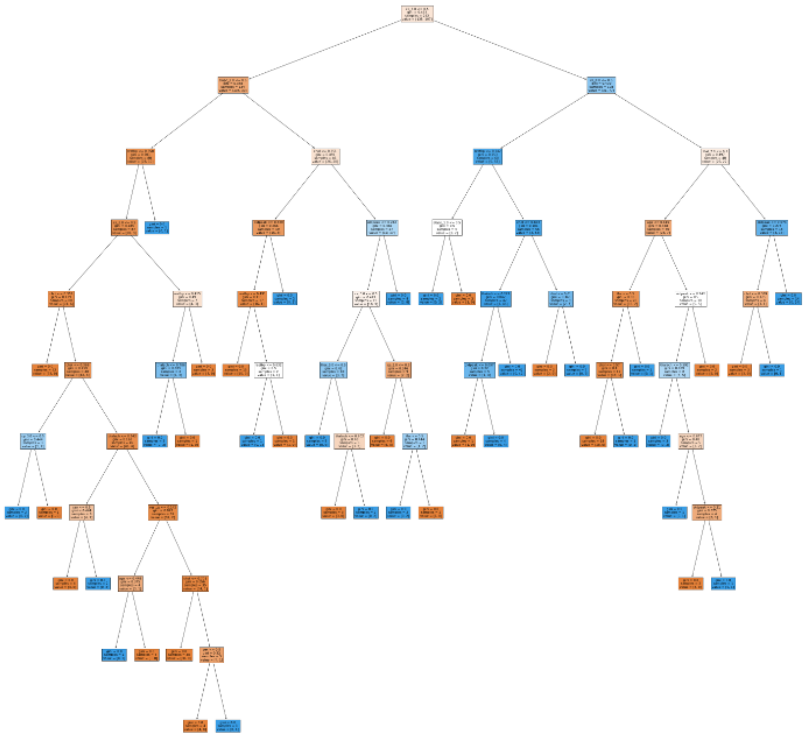
```
In [5]: y_not_zero=y>0
        y[y_not_zero]=1
        y
```

Out[5]:

	hd
0	0
1	1
2	1
3	0
4	0
...	...
298	1
299	1
300	1
301	1
302	0

303 rows × 1 columns

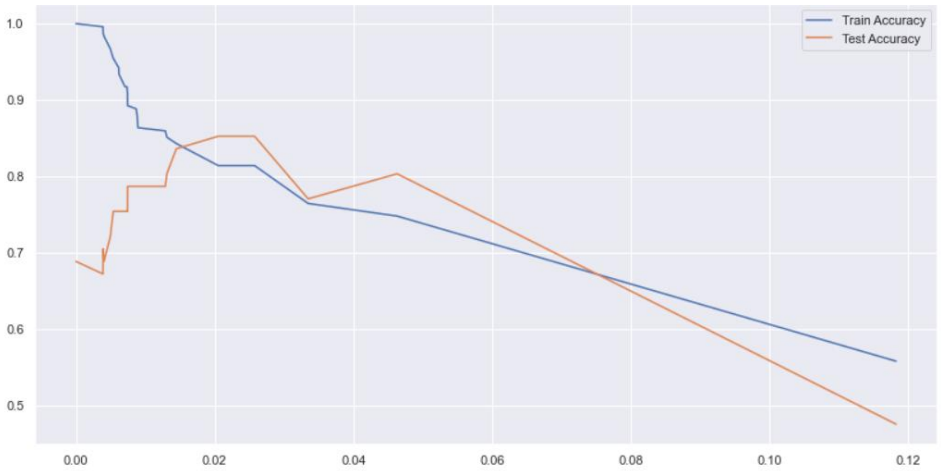
درخت اولیه در این حالت:



confusion matrix:

```
Accuracy: 0.7049180327868853
[[21  8]
 [10 22]]
```

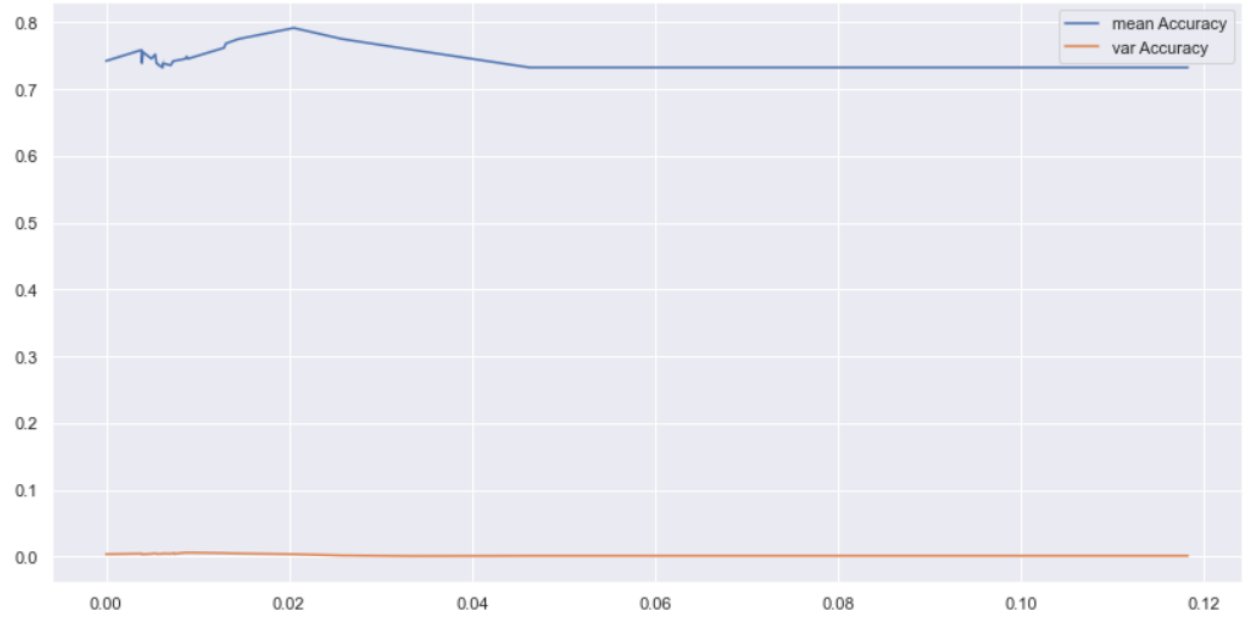
Pruning:

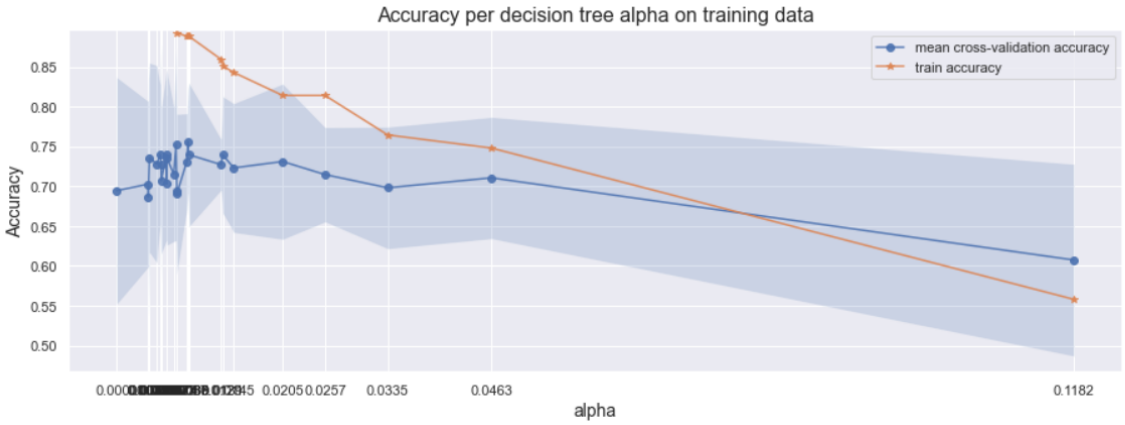


با توجه به نمودار بالا و test accuracy می توان دریافت که با یک آلفای بین ۰/۰۲ و ۰/۰۳ می توان به بهترین Accuracy رسید. که با توجه به لیست آلفاها در می یابیم که تنها آلفای این بازه ۰/۰۲۵۷۴۸۷۴ می باشد.

پس با توجه به نمودار بالا بهترین آلفا برابر است با: 0.02574874

5_fold cross validation:





میانگین Accuracy برای الفاهای مختلف:

```
[(0.0, 0.7422950819672132), (0.003889158969372874, 0.7587431693989071), (0.003920322102140283, 0.7389617486338798), (0.00404431
15878319, 0.7555191256830602), (0.004958677685950413, 0.7456284153005465), (0.005371900826446282, 0.7521311475409836), (0.00550
9641873278236, 0.7356284153005463), (0.005509641873278236, 0.7422404371584699), (0.006169209578300493, 0.7322950819672129), (0.
006198347107438017, 0.7455191256830602), (0.006198347107438017, 0.7322950819672129), (0.007062359128474829 6, 0.735683060109289
6), (0.007346189164370982 5, 0.7422950819672132), (0.007438016528925622, 0.7422404371584699), (0.007447541175781022, 0.742295081
9672132), (0.008677685950413223, 0.7455191256830602), (0.008815426997245177, 0.7488524590163935), (0.00893625247692234, 0.74551
91256830602), (0.012888051165563116, 0.7619125683060111), (0.013100324949466523, 0.7684699453551913), (0.014462809917355372, 0.
7750819672131148), (0.020510212729892 37, 0.7916393442622951), (0.025748742973792704, 0.7752459016393443), (0.03347107438016529,
0.7589071038251365), (0.046296296296335, 0.7324590163934426), (0.11822023721672725, 0.7324590163934426)]
```

واریانس برای آلفاهای مختلف:

```
[(0.0, 0.003991143360506432), (0.003889158969372874, 0.004774881304308877), (0.003920322102140283, 0.003776451969303351), (0.00
40443115878319, 0.003840735763982203), (0.004958677685950413, 0.004509719609423988), (0.005371900826446282, 0.00514420854608976
2), (0.005509641873278236, 0.00419175251575144), (0.005509641873278236, 0.0045636955418197), (0.006169209578300493, 0.004473719
728866192), (0.006198347107438017, 0.005436937501866283), (0.006198347107438017, 0.004473719728866192), (0.007062359128474829 6,
0.004580178566096328 5), (0.007346189164370982 5, 0.005281119173459944), (0.007438016528925622, 0.005423679417122044), (0.007447541
175781022, 0.004313637313744809), (0.008677685950413223, 0.006404419361581415), (0.008815426997245177, 0.006145402968138792),
(0.008936252476922 34, 0.006404419361581415), (0.012888051165563116, 0.005503227925587506), (0.013100324949466523, 0.005357747320
0155275), (0.014462809917355372, 0.004980124817104122), (0.020510212729892 37, 0.003963020693362), (0.025748742973792704, 0.001
968933082504705), (0.033471074380165 29, 0.001131284899519 2464), (0.046296296296335, 0.001469330227836 0057), (0.1182202372167
2725, 0.001469330227836 0057)]
```

از روی زوج مرتب های آلفا و میانگین Accuracy و همچنین نمودار رسم شده می توان دریافت که آلفای ۰/۰۲۰۵۱۰۲۱ بیشترین میانگین Accuracyرا دارد.

بهترین آلفا با استفاده از5fold-cross validation برابر است با: ۰/۰۲۰۵۱۰۲۱

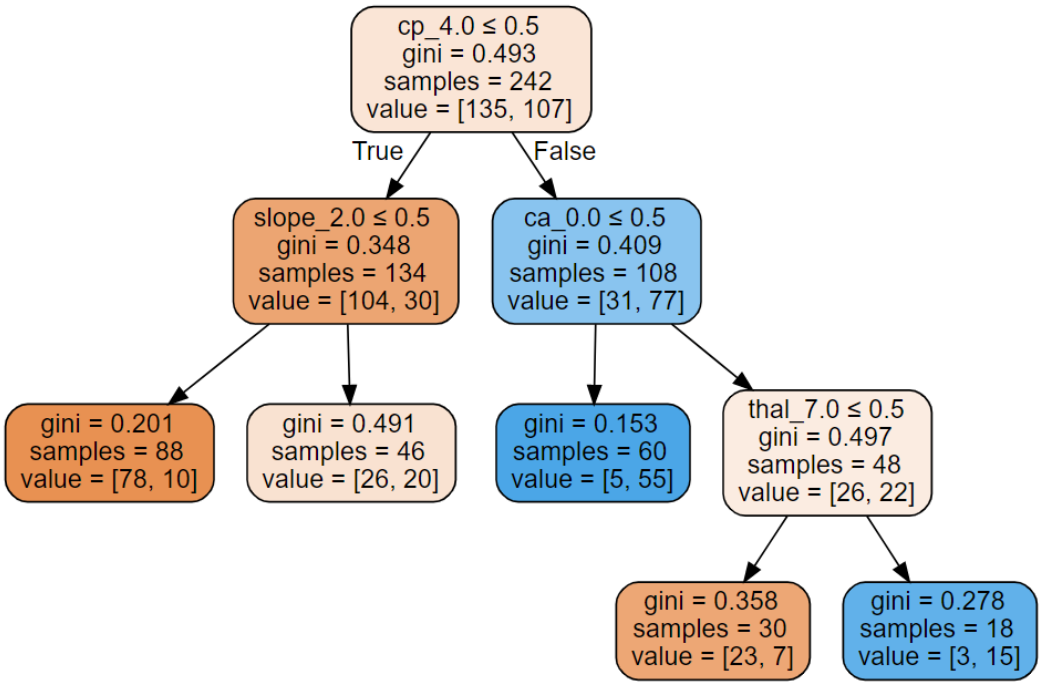
ساخت و ارزیابی و رسم درخت تصمیم با آلفای مشخص برای حالت دوم تحلیل:

با توجه به نمودارهایی که تحلیل شد بهترین آلفا ۰/۰۲۰۵۱۰۲۱ بود که می خواهیم با این آلفا درخت را هرس کرده و درخت را رسم کنیم و برای آن Accuracy و confusion matrix را هم حساب می کنیم

```
0.8524590163934426
[[27  2]
 [ 7 25]]
```

با توجه به Accuracy و confusion matrix می توان دریافت که درخت بهتری داریم که accuracy از ۷۰به ۸۵ رسیده است

رسم ساختار درخت پایانی بعد از هرس برای حالت دوم تحلیل:



دقت شود این تمرین در ۲ حالت تحلیل شد:

۱- حالتی که ستون بیماری قلبی (hd) دارای ۵ مقدار است(پیش فرض مسئله)

۲- حالتی که ستون بیماری قلبی دارای ۲ مقداراست(وجود یا عدم وجود بیماری)