In [1]:
```python
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score as ac ,confusion_matrix as cm ,classification_report as cr
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split as t
from sklearn.tree import DecisionTreeClassifier as dt
```

In [2]:
```python
df=pd.read_csv('heart.csv')
```

In [3]:
```python
df.head()
```

Out[3]:

	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

In [4]:
```python
df.corr()
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.096801	0.210013	-0.168814	0.276326	0.068001	-0.225439
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.141664	0.096093	-0.030711	0.118261	0.210041	-0.280937
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.394280	-0.149230	0.119717	-0.181053	-0.161736	0.433798
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	0.193216	-0.121475	0.101389	0.062210	-0.144931
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	0.053952	-0.004038	0.070511	0.098803	-0.085239
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.005747	-0.059894	0.137979	-0.032019	-0.028046
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-0.058770	0.093045	-0.072042	-0.011981	0.137230
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-0.344187	0.386784	-0.213177	-0.096439	0.421741
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	0.288223	-0.257748	0.115739	0.206754	-0.436757
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1.000000	-0.577537	0.222682	0.210244	-0.430696
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-0.577537	1.000000	-0.080155	-0.104764	0.345877
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.115739	0.222682	-0.080155	1.000000	0.151832	-0.391724
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.206754	0.210244	-0.104764	0.151832	1.000000	-0.344029
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-0.436757	-0.430696	0.345877	-0.391724	-0.344029	1.000000

In [ ]:

In [ ]:

In [ ]:

In [5]:
```python
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

In [26]:
```python
xtrain,xtest,ytrain,ytest=t(x,y,random_state=0)
```

In [38]:
```python
dtree=dt()
dtree.fit(xtrain,ytrain)
ypred=dtree.predict(xtest)
print(ac(ytest,ypred))
```

```
0.8026315789473685
```

In [44]:
```python
print('Training score')
dtree.score(xtrain,ytrain)
```

```
Training score
```

Out[44]: 1.0

In [45]:
```python
print('Testing score')
dtree.score(xtest,ytest)
```

```
Testing score
```

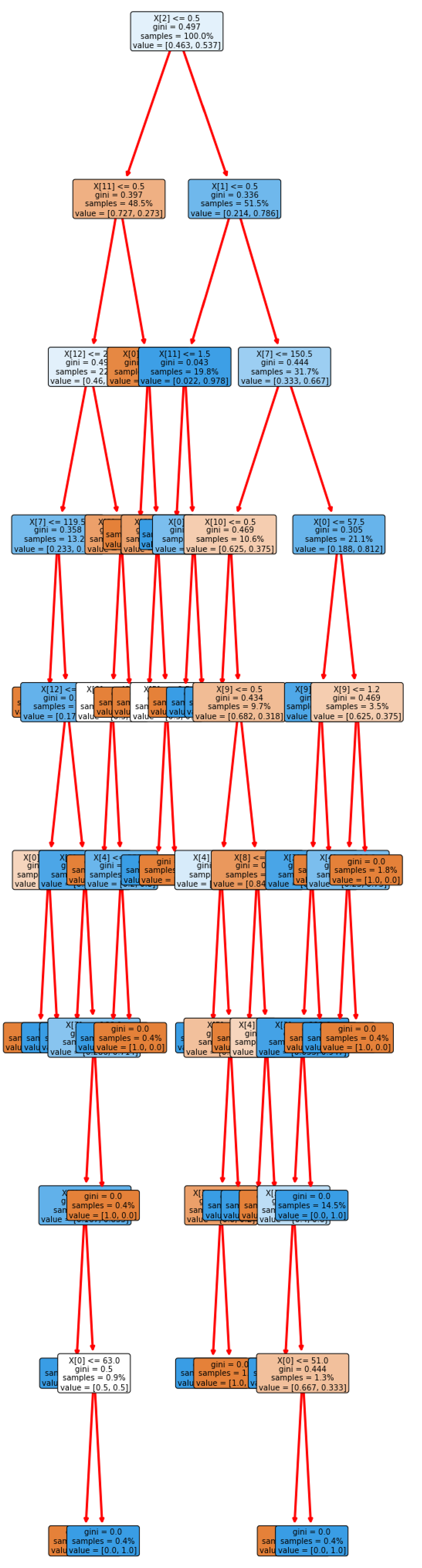
Out[45]: 0.8026315789473685

In [41]:
```python
from sklearn import tree
```

In the below the plot we have plotted the Decision Tree

We can the max depth is 10

In [43]:
```python
plt.figure(figsize=(8,40))
chart=tree.plot_tree(dtree,filled=True,fontsize=10,rounded=True,proportion=True)
for node in chart:
    arrow=node.arrow_patch
    if (arrow is not None):
        arrow.set_edgecolor('red')
        arrow.set_linewidth(3)
```



Below we have use Post pruning decision tree with cost complexity to overcome the situation of overfitting

In [ ]:
```python
path=dtree.cost_complexity_pruning_path(xtrain,ytrain)
```

In [13]:
```python
path.ccp_alphas
```

Out[13]:
```
array([0.        , 0.00367107, 0.00430739, 0.00437336, 0.00528634,
       0.00587372, 0.00612103, 0.00660793, 0.00704846, 0.0074551 ,
       0.00751217, 0.00763395, 0.00837004, 0.00991189, 0.0100487 ,
       0.01057269, 0.01109713, 0.01618943, 0.02529886, 0.03395007,
       0.05769323, 0.13176578])
```

In [14]:
```python
path.impurities
```

Out[14]:
```
array([0.        , 0.00734214, 0.01595693, 0.02470365, 0.03527634,
       0.04702377, 0.0531448 , 0.05975273, 0.06680119, 0.08171139,
       0.09673573, 0.11963757, 0.14474771, 0.1546596 , 0.17475701,
       0.19590238, 0.20699951, 0.22318894, 0.27378666, 0.30773674,
       0.36542997, 0.49719575])
```

In [15]:
```python
alphas,impure=path.ccp_alphas , path.impurities
```

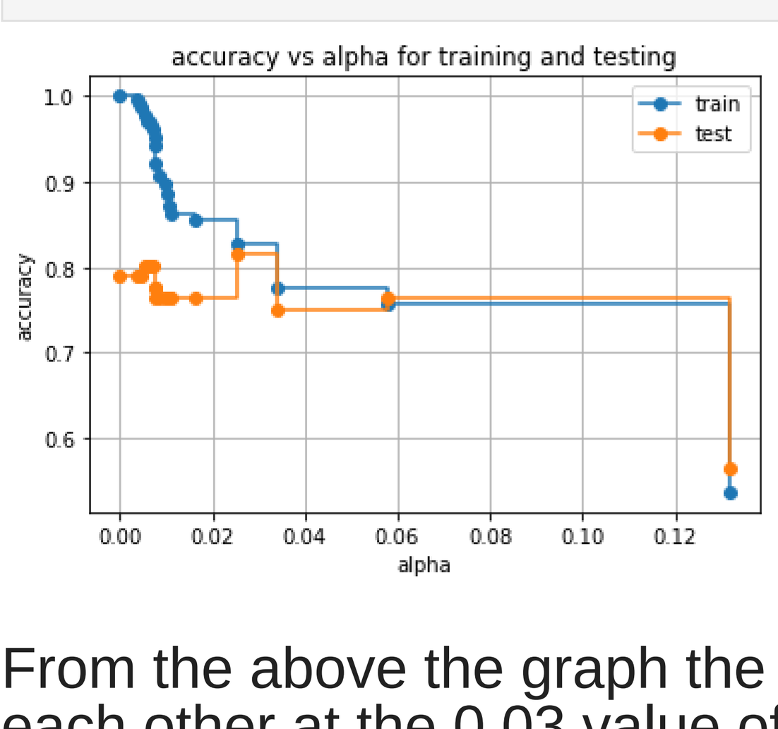
In [16]:
```python
clfs=[]
for i in alphas:
    i=dt(random_state=0,ccp_alpha=i)
    i.fit(xtrain,ytrain)
    clfs.append(i)
```

In [17]:
```python
print(f'Number of nodes  in the last tree  is {clfs[-1].tree_.node_count}   {alphas[-1]}')
```

```
Number of nodes  in the last tree  is 1   0.13176578489314994
```

In [18]:
```python
training_score=[clf.score(xtrain,ytrain) for clf in clfs]
testing_score=[clf.score(xtest,ytest) for clf in clfs]
```

In [19]:
```python
fig,ax=plt.subplots()
plt.grid()
ax.set_xlabel('alpha')
ax.set_ylabel('accuracy')
ax.set_title('accuracy vs alpha for training and testing')
ax.plot(alphas,training_score,marker='o',label='train',drawstyle='steps-post')
ax.plot(alphas,testing_score,marker='o',label='test',drawstyle='steps-post')
ax.plot()
ax.legend()
plt.show()
```



From the above the graph the training we can see the training and testing score are near to each other at the 0.03 value of aplha

In [48]:
```python
dtree=dt(ccp_alpha=0.03)
dtree.fit(xtrain,ytrain)
ypred=dtree.predict(xtest)
```

In [49]:
```python
dtree.score(xtrain,ytrain)
```

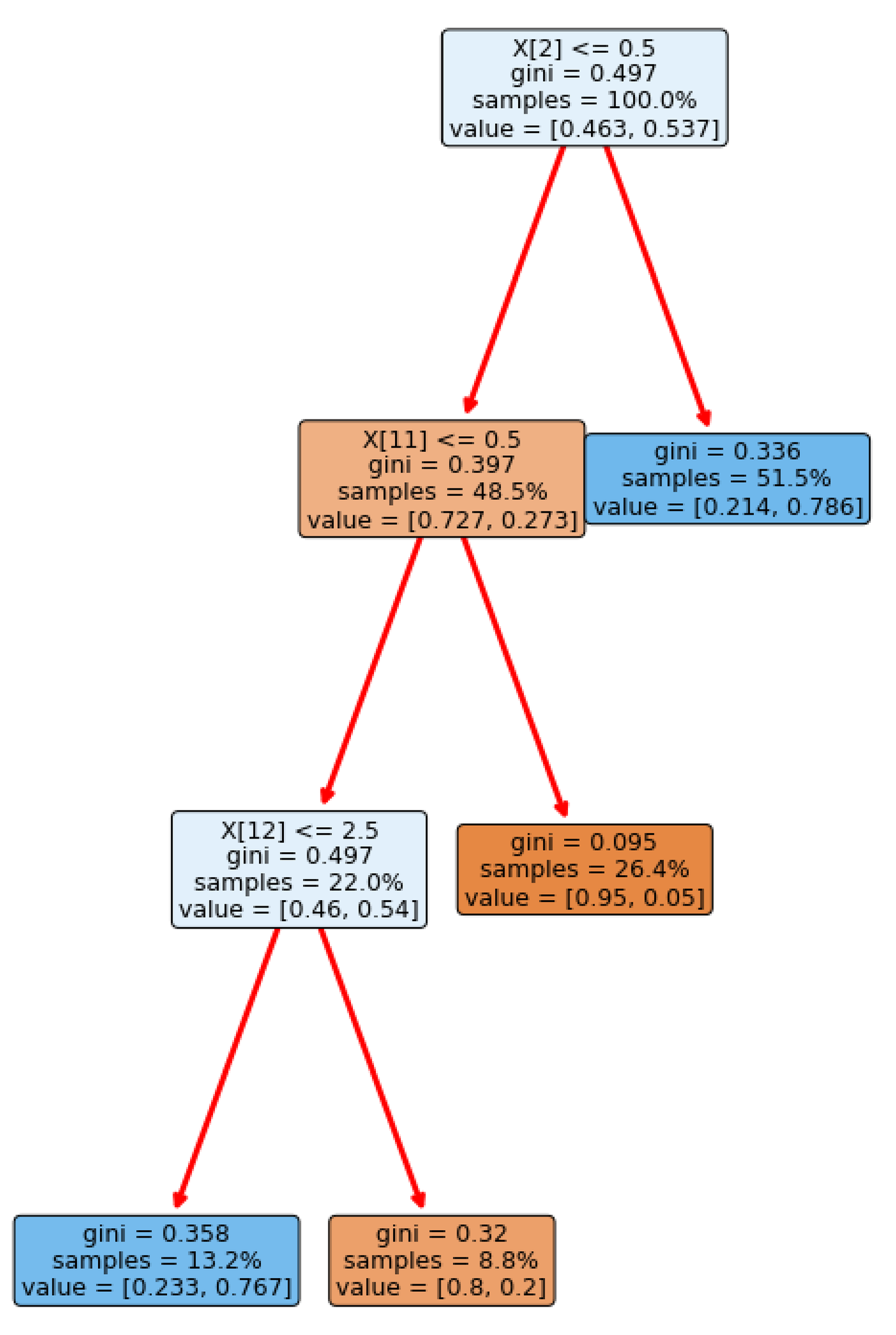
Out[49]: 0.8281938325991189

In [50]:
```python
dtree.score(xtest,ytest)
```

Out[50]: 0.8157894736842105

From the above the score we can see the it has overcomed the situation of overfitting to besting fitting

In [51]:
```python
plt.figure(figsize=(8,15))
chart=tree.plot_tree(dtree,filled=True,rounded=True,proportion=True)
for node in chart:
    arrow=node.arrow_patch
    if (arrow is not None):
        arrow.set_edgecolor('red')
        arrow.set_linewidth(3)
```



By comparing the above the decision plot to above plot we can see the max depth has been reduced to 3 from 10

In [ ]: