In [1]:

```
#Install any necessary libraries
# !pip freeze
# !pip3 install numpy
# !pip3 install pandas
# !pip3 install sklearn
# !pip3 install matplotlib
```

In [2]:

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

from sklearn.tree import export_graphviz
from sklearn import tree
# from sklearn.neural_network import MLPClassifier
# from sklearn.model_selection import GridSearchCV
```

Cleveland Dataset

Attribute Information:

Only 14 attributes used:

1. #3 (age)
2. #4 (sex)
3. #9 (cp)
4. #10 (trestbps)
5. #12 (chol)
6. #16 (fbs)
7. #19 (restecg)
8. #32 (thalach)
9. #38 (exang)
10. #40 (oldpeak)
11. #41 (slope)
12. #44 (ca)
13. #51 (thal)
14. #58 (num) (the predicted attribute)

In [3]:

```
attributes = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thala
cols = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',

df = pd.read_csv('cleaned_processed.cleveland.data', names=attributes)

X = df[cols]
Y = df.num

X_train1, X_test1, Y_train1, Y_test1 = train_test_split(X, Y, test_size=0.1)
X_train2, X_test2, Y_train2, Y_test2 = train_test_split(X, Y, test_size=0.2)
X_train3, X_test3, Y_train3, Y_test3 = train_test_split(X, Y, test_size=0.3)
X_train4, X_test4, Y_train4, Y_test4 = train_test_split(X, Y, test_size=0.4)
```

1. Decision tree

In [4]:

```
tree_entropy1 = []
d3_1 = DecisionTreeClassifier(random_state=0,criterion="entropy")
keys_d_1 = d3_1.cost_complexity_pruning_path(X_train1, Y_train1)['ccp_alphas']

trainAccuracy1 = []
testAccuracy1 = []
for alpha in keys_d_1:
    d3_1_Temp = DecisionTreeClassifier(random_state=0,criterion="entropy", ccp
    d3_1_Temp.fit(X_train1, Y_train1)
    tree_entropy1.append(d3_1_Temp)

    trainAccuracy1.append( d3_1_Temp.score(X_train1, Y_train1))
    testAccuracy1.append(d3_1_Temp.score(X_test1, Y_test1))

tree_entropy1 = tree_entropy1[:-1]
trainAccuracy1 = trainAccuracy1[:-1]
testAccuracy1 = testAccuracy1[:-1]
keys_d_1 = keys_d_1[:-1]
print("entropy,10%")
i1 = testAccuracy1.index(max(testAccuracy1))
print(trainAccuracy1[i1])
print(testAccuracy1[i1])

fig1, ax1 = plt.subplots()
ax1.set_xlabel("Alpha")
ax1.set_ylabel("Accuracy")
ax1.set_title("Training and test accuracy. Split Criterion: Entropy. Test Size
ax1.plot(keys_d_1, trainAccuracy1, label="training")
ax1.plot(keys_d_1, testAccuracy1, label="test")
ax1.legend()

tree_entropy2 = []
d3_2 = DecisionTreeClassifier(random_state=0, criterion="entropy")
keys_d_2 = d3_2.cost_complexity_pruning_path(X_train2, Y_train2)['ccp_alphas']

trainAccuracy2 = []
testAccuracy2 = []
for alpha in keys_d_2:
    d3_2_Temp = DecisionTreeClassifier(random_state=0,criterion="entropy", ccp
    d3_2_Temp.fit(X_train2, Y_train2)
    tree_entropy2.append(d3_2_Temp)

    trainAccuracy2.append( d3_2_Temp.score(X_train2, Y_train2))
    testAccuracy2.append(d3_2_Temp.score(X_test2, Y_test2))

tree_entropy2 = tree_entropy2[:-1]
trainAccuracy2 = trainAccuracy2[:-1]
testAccuracy2 = testAccuracy2[:-1]
keys_d_2 = keys_d_2[:-1]
print("entropy,20%")
i2 = testAccuracy2.index(max(testAccuracy2))
print(trainAccuracy2[i2])
print(testAccuracy2[i2])

fig2, ax2 = plt.subplots()
ax2.set_xlabel("Alpha")
ax2.set_ylabel("Accuracy")
ax2.set_title("Training and test accuracy. Split Criterion: Entropy. Test Size
trainAccuracy3 = []
testAccuracy3 = []
for alpha in keys_d_3:
    d3_3_Temp = DecisionTreeClassifier(random_state=0,criterion="entropy", ccp
    d3_3_Temp.fit(X_train3, Y_train3)
    tree_entropy3.append(d3_3_Temp)

    trainAccuracy3.append( d3_3_Temp.score(X_train3, Y_train3))
    testAccuracy3.append(d3_3_Temp.score(X_test3, Y_test3))

tree_entropy3 = tree_entropy3[:-1]
trainAccuracy3 = trainAccuracy3[:-1]
testAccuracy3 = testAccuracy3[:-1]
keys_d_3 = keys_d_3[:-1]
print("entropy,30%")
i3 = testAccuracy3.index(max(testAccuracy3))
print(trainAccuracy3[i3])
print(testAccuracy3[i3])

fig3, ax3 = plt.subplots()
ax3.set_xlabel("Alpha")
ax3.set_ylabel("Accuracy")
ax3.set_title("Training and test accuracy. Split Criterion: Entropy. Test Size
ax3.plot(keys_d_3, trainAccuracy3, label="training")
ax3.plot(keys_d_3, testAccuracy3, label="test")
ax3.legend()


tree_entropy4 = []
d3_4 = DecisionTreeClassifier(random_state=0,criterion="entropy")
keys_d_4 = d3_4.cost_complexity_pruning_path(X_train4, Y_train4)['ccp_alphas']

trainAccuracy4 = []
testAccuracy4 = []
for alpha in keys_d_4:
    d3_4_Temp = DecisionTreeClassifier(random_state=0,criterion="entropy", ccp
    d3_4_Temp.fit(X_train4, Y_train4)
    tree_entropy4.append(d3_4_Temp)

    trainAccuracy4.append( d3_4_Temp.score(X_train4, Y_train4))
    testAccuracy4.append(d3_4_Temp.score(X_test4, Y_test4))

tree_entropy4 = tree_entropy4[:-1]
trainAccuracy4 = trainAccuracy4[:-1]
testAccuracy4 = testAccuracy4[:-1]
keys_d_4 = keys_d_4[:-1]
print("entropy,40%")
i4 = testAccuracy4.index(max(testAccuracy4))
print(trainAccuracy4[i4])
print(testAccuracy4[i4])
```

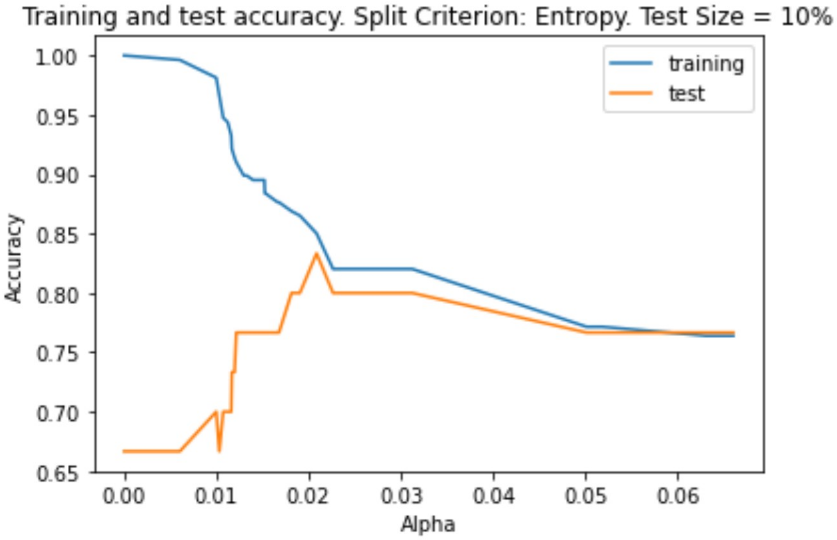
```
    d3_11_Temp = DecisionTreeClassifier(random_state=0,criterion="gini", ccp_a
    d3_11_Temp.fit(X_train1, Y_train1)
    tree_gini1.append(d3_11_Temp)

    trainAccuracy11.append( d3_11_Temp.score(X_train1, Y_train1))
    testAccuracy11.append(d3_1_Temp.score(X_test1, Y_test1))

tree_gini1 = tree_gini1[:-1]
trainAccuracy11 = trainAccuracy11[:-1]
testAccuracy11 = testAccuracy11[:-1]
keys_d_11 = keys_d_11[:-1]
print("gini,10%")
j1 = testAccuracy11.index(max(testAccuracy11))
print(trainAccuracy11[j1])
print(testAccuracy11[j1])

fig11, ax11 = plt.subplots()
ax11.set_xlabel("Alpha")
ax11.set_ylabel("Accuracy")
ax11.set_title("Training and test accuracy. Split Criterion: Gini. Test Size =
ax11.plot(keys_d_11, trainAccuracy11, label="training")
ax11.plot(keys_d_11, testAccuracy11, label="test")
ax11.legend()

tree_gini2 = []
d3_22 = DecisionTreeClassifier(random_state=0, criterion="gini")
keys_d_22 = d3_22.cost_complexity_pruning_path(X_train2, Y_train2)['ccp_alphas

trainAccuracy22 = []
testAccuracy22 = []
for alpha in keys_d_22:
    d3_22_Temp = DecisionTreeClassifier(random_state=0, criterion="gini", ccp_
    d3_22_Temp.fit(X_train2, Y_train2)
    tree_gini2.append(d3_22_Temp)

    trainAccuracy22.append( d3_22_Temp.score(X_train2, Y_train2))
    testAccuracy22.append(d3_22_Temp.score(X_test2, Y_test2))

tree_gini2 = tree_gini2[:-1]
trainAccuracy22 = trainAccuracy22[:-1]
testAccuracy22 = testAccuracy22[:-1]
keys_d_22 = keys_d_22[:-1]
j2 = testAccuracy22.index(max(testAccuracy22))
print("gini,20%")
print(trainAccuracy22[j2])
```

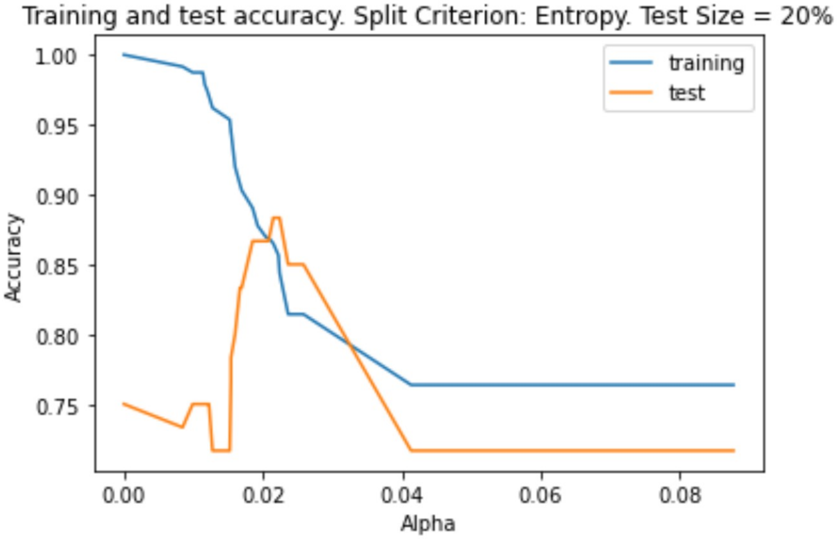
```
    trainAccuracy33.append( d3_33_Temp.score(X_train3, Y_train3))
    testAccuracy33.append(d3_33_Temp.score(X_test3, Y_test3))

tree_gini3 = tree_gini3[:-1]
trainAccuracy33 = trainAccuracy33[:-1]
testAccuracy33 = testAccuracy33[:-1]
keys_d_33 = keys_d_33[:-1]
print("gini,30%")
j3 = testAccuracy33.index(max(testAccuracy33))
print(trainAccuracy33[j3])
print(testAccuracy33[j3])

fig33, ax33 = plt.subplots()
ax33.set_xlabel("Alpha")
ax33.set_ylabel("Accuracy")
ax33.set_title("Training and test accuracy. Split Criterion: Gini. Test Size =
ax33.plot(keys_d_33, trainAccuracy33, label="training")
ax33.plot(keys_d_33, testAccuracy33, label="test")
ax33.legend()


tree_gini4 = []
d3_44 = DecisionTreeClassifier(random_state=0,criterion="gini")
keys_d_44 = d3_44.cost_complexity_pruning_path(X_train4, Y_train4)['ccp_alphas

trainAccuracy44 = []
testAccuracy44 = []
for alpha in keys_d_44:
    d3_44_Temp = DecisionTreeClassifier(random_state=0,criterion="gini", ccp_a
    d3_44_Temp.fit(X_train4, Y_train4)
    tree_gini4.append(d3_44_Temp)

    trainAccuracy44.append( d3_44_Temp.score(X_train4, Y_train4))
    testAccuracy44.append(d3_44_Temp.score(X_test4, Y_test4))

tree_gini4 = tree_gini4[:-1]
trainAccuracy44 = trainAccuracy44[:-1]
testAccuracy44 = testAccuracy44[:-1]
```

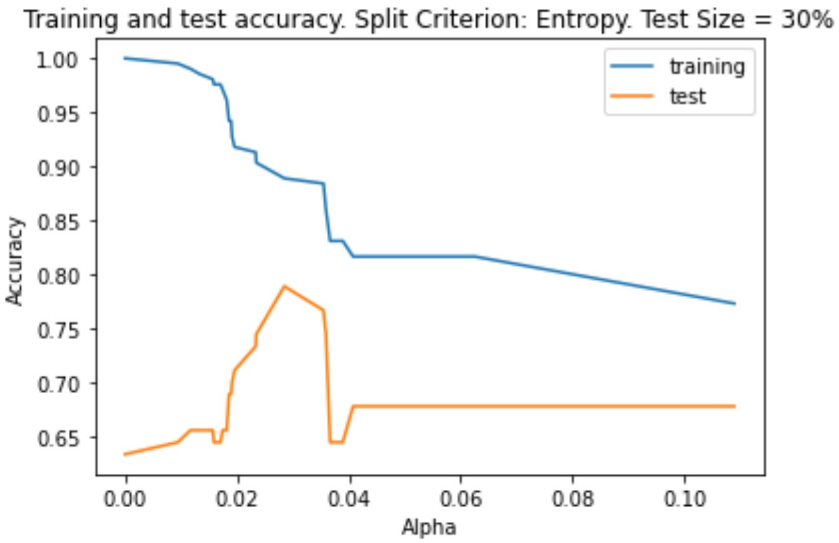
```
print(test_max_entropy.index(max(test_max_entropy)))

figmaxm2, axmax2 = plt.subplots()
axmax2.set_xlabel("Test data size")
axmax2.set_ylabel("Max Accuracy")
axmax2.set_title("Training and test accuracy vs Test Size. Split Criterion: G
axmax2.plot(test_size, training_max_gini, label="training")
axmax2.plot(test_size, test_max_gini, label="test")
axmax2.legend()
print("best split gini")
print(test_max_gini.index(max(test_max_gini)))
```

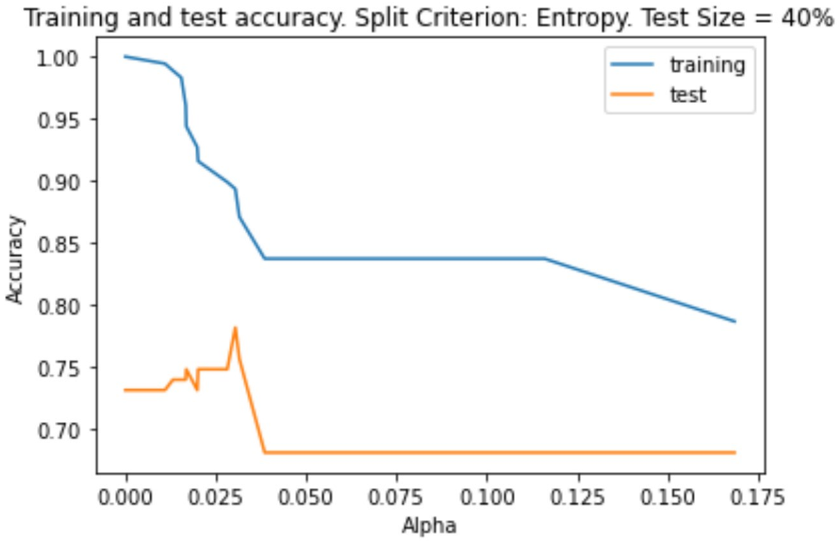
```
entropy,10%
0.850187265917603
0.8333333333333334
entropy,20%
0.8649789029535865
0.8833333333333333
entropy,30%
0.8888888888888888
0.7888888888888889
entropy,40%
0.8932584269662921
0.7815126050420168
gini,10%
1.0
0.5333333333333333
gini,20%
0.869198312236287
0.8833333333333333
```

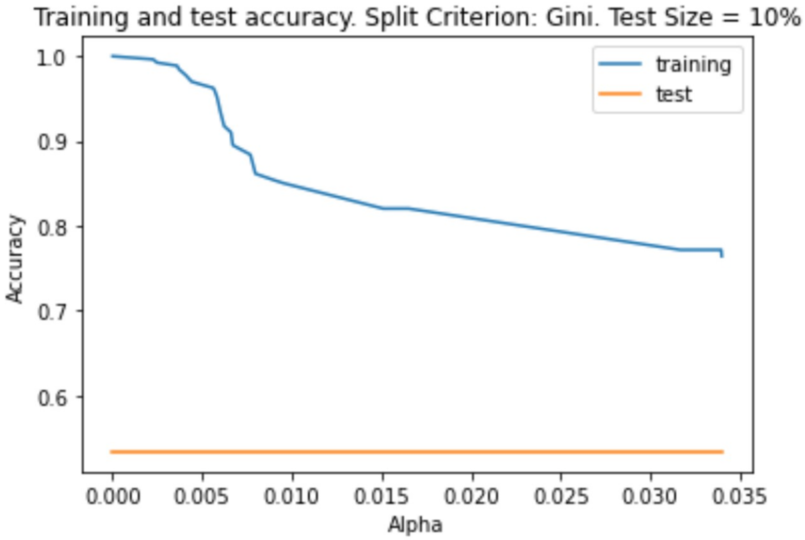
```
gini,30%
0.8695652173913043
0.8
gini,40%
0.8932584269662921
0.7815126050420168
best split entropy
2
best split gini
2
```

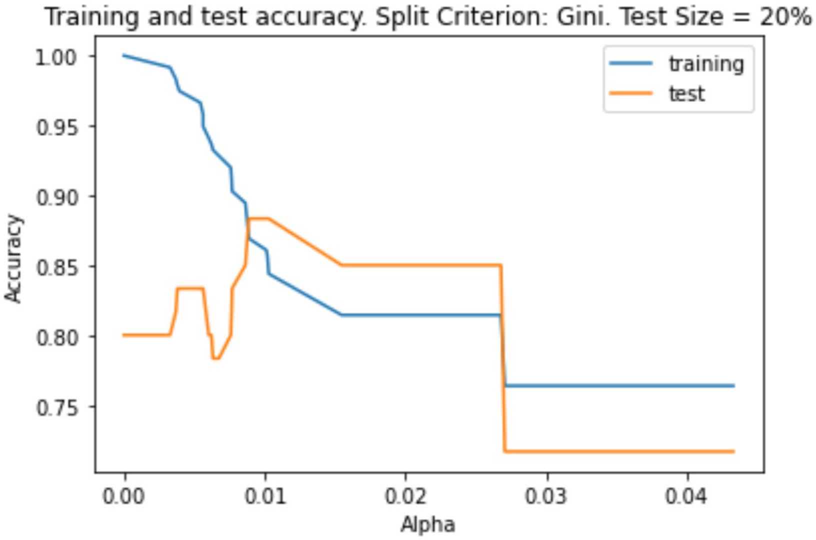


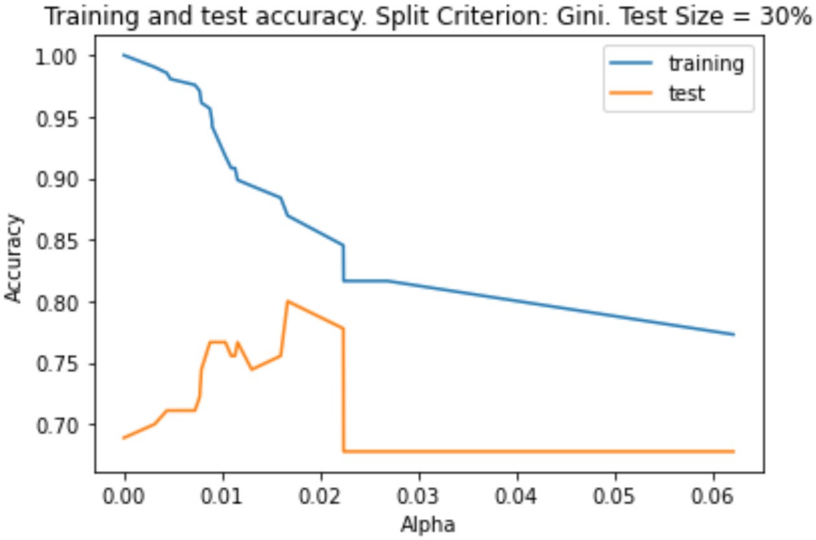


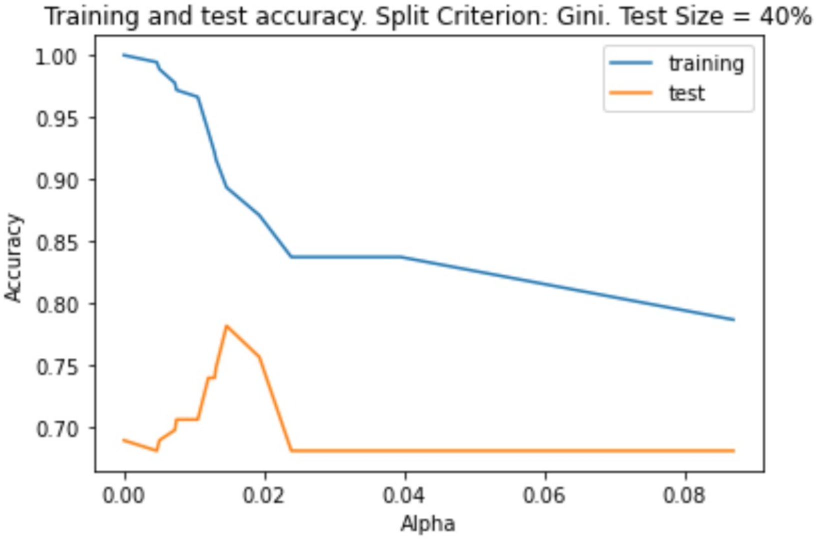

Training and test accuracy. Split Criterion: Entropy. Test Size = 30%
Accuracy
Alpha
training
test
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.00
0.02
0.04
0.06
0.08
0.10

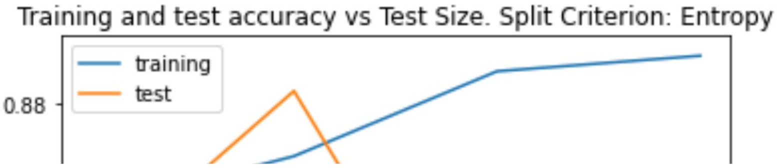

Training and test accuracy. Split Criterion: Entropy. Test Size = 40%
Accuracy
Alpha
training
test
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175


Training and test accuracy. Split Criterion: Gini. Test Size = 10%
Accuracy
Alpha
training
test
1.0
0.9
0.8
0.7
0.6
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035


Training and test accuracy. Split Criterion: Gini. Test Size = 20%
training
test
Accuracy
1.00
0.95
0.90
0.85
0.80
0.75
Alpha
0.00
0.01
0.02
0.03
0.04


Training and test accuracy. Split Criterion: Gini. Test Size = 30%
training
test
Accuracy
1.00
0.95
0.90
0.85
0.80
0.75
0.70
Alpha
0.00
0.01
0.02
0.03
0.04
0.05
0.06


Training and test accuracy. Split Criterion: Gini. Test Size = 40%
training
test
Accuracy
1.00
0.95
0.90
0.85
0.80
0.75
0.70
Alpha
0.00
0.02
0.04
0.06
0.08



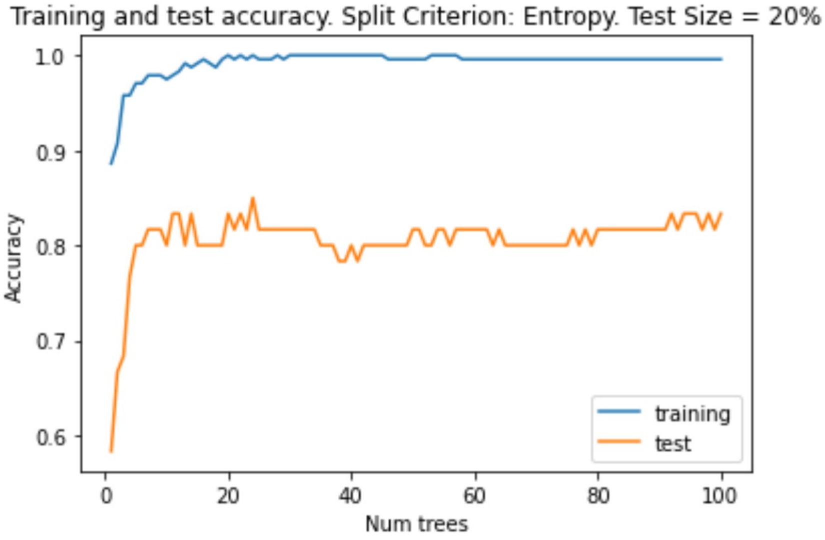
2. Random Forests

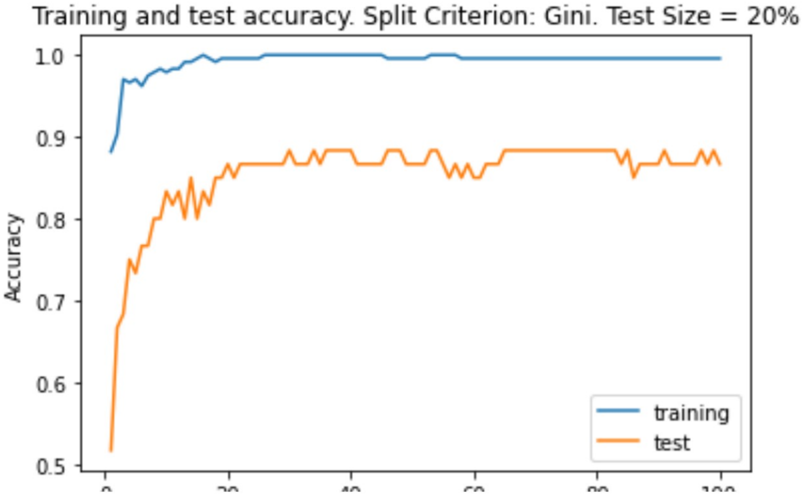
In [5]:

```
#num_trees = [1,10,20,30,40,50,60,70,80,90,100]
num_trees = np.arange(1,101)

trainAccuracyrf = []
testAccuracyrf = []
est, train, test= 0,0,0
for n in num_trees:
    model = RandomForestClassifier(criterion="entropy", n_estimators=n, max_fe
    model.fit(X_train2, Y_train2)
    trainAccuracyrf.append(model.score(X_train2, Y_train2))
    testAccuracyrf.append(model.score(X_test2, Y_test2))

    if(model.score(X_test2, Y_test2) > test):
        est = n
        train = model.score(X_train2, Y_train2)
        test = model.score(X_test2, Y_test2)

figrf, axrf = plt.subplots()
axrf.set_xlabel("Num trees")
axrf.set_ylabel("Accuracy")
axrf.set_title("Training and test accuracy. Split Criterion: Entropy. Test Si
axrf.plot(num_trees, trainAccuracyrf, label="training")
axrf.plot(num_trees, testAccuracyrf, label="test")
axrf.legend()

print("Entropy")
print(est)
print(train)
print(test)

trainAccuracyrf1 = []
testAccuracyrf1 = []
est1, train1, test1 = 0,0,0
for n in num_trees:
    model1 = RandomForestClassifier(criterion="gini", n_estimators=n, max_feat
    model1.fit(X_train2, Y_train2)
    trainAccuracyrf1.append(model1.score(X_train2, Y_train2))
    testAccuracyrf1.append(model1.score(X_test2, Y_test2))

    if(model1.score(X_test2, Y_test2) > test1):
        est1 = n
        train1 = model1.score(X_train2, Y_train2)
        test1 = model1.score(X_test2, Y_test2)

figrf1, axrf1 = plt.subplots()
axrf1.set_xlabel("Num trees")
axrf1.set_ylabel("Accuracy")
axrf1.set_title("Training and test accuracy. Split Criterion: Gini. Test Size
axrf1.plot(num_trees, trainAccuracyrf1, label="training")
axrf1.plot(num_trees, testAccuracyrf1, label="test")
axrf1.legend()

print("GINI")
print(est1)
print(train1)
print(test1)
```

```
        for d in depth:
            model2 = RandomForestClassifier(criterion="gini", n_estimators=n, max_
            model2.fit(X_train2, Y_train2)
            temp_train.append(model2.score(X_train2, Y_train2))
            temp_test.append(model2.score(X_test2, Y_test2))

            if(model2.score(X_test2, Y_test2) > test2):
                est2 = n
                train2 = model2.score(X_train2, Y_train2)
                test2 = model2.score(X_test2, Y_test2)
                depth2 = d
        trainAccuracyrf2.append(temp_train)
        testAccuracyrf2.append(temp_test)

print("GINI with depth")
print(est2)
print(depth2)
print(train2)
print(test2)
```

```
Entropy
24
1.0
0.85
GINI
30
1.0
0.8833333333333333
GINI with depth
26
2
0.8396624472573839
0.9166666666666666
```




Training and test accuracy. Split Criterion: Gini. Test Size = 20%
Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
training
test

In [ ]: