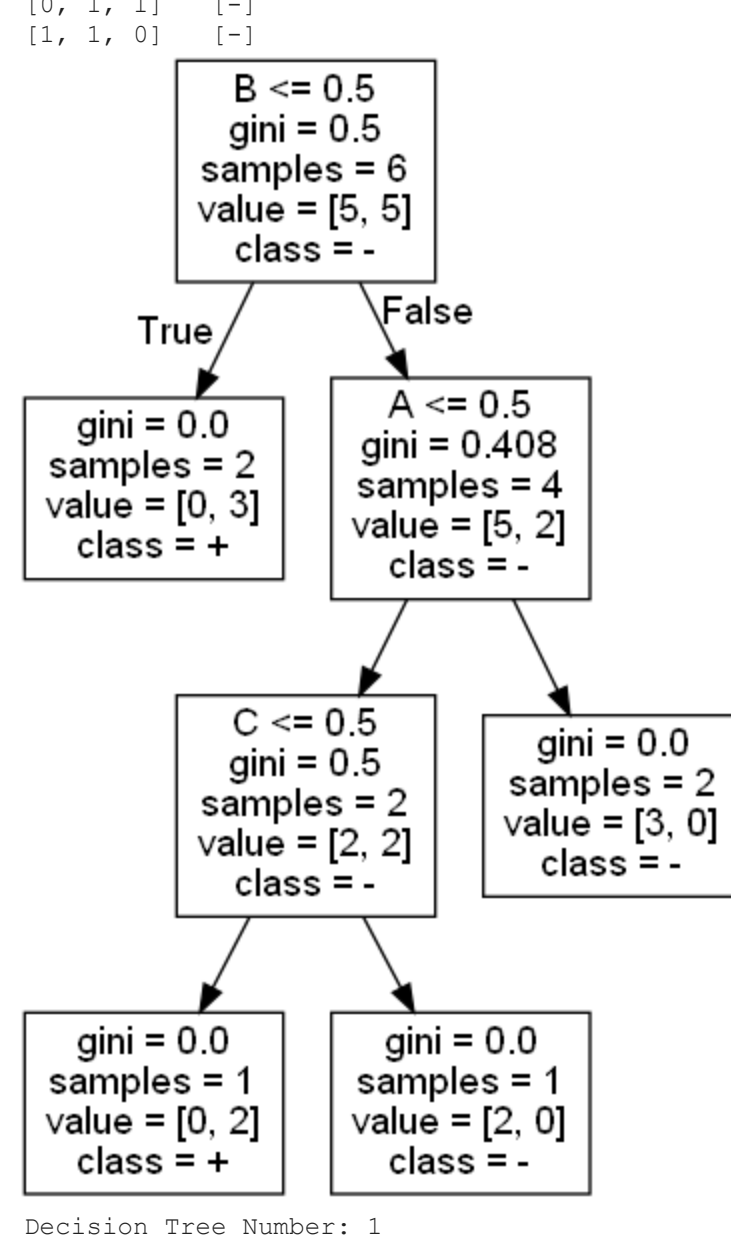
CS 5342 Project 4: Tyler Martin and Joel Fecke

Part 1: Bagging

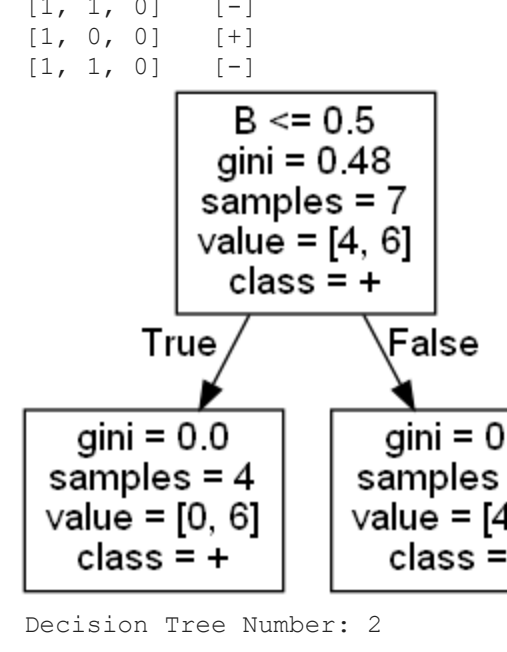
In [1]:
```python
import numpy as np
from multiprocessing import Process
import math
from array import *
from sklearn.datasets import make_classification
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn import tree
import graphviz
from IPython.display import Image
import matplotlib.pyplot as plt
```

In [2]:
```python
#Get the data from figure 3.36 to a 2D array for simplicity
#             A   B   C
TrainingAttr = [[0, 0, 0],
            [0, 0, 1],
            [0, 1, 0],
            [0, 1, 1],
            [1, 0, 0],
            [1, 0, 0],
            [1, 1, 0],
            [1, 0, 1],
            [1, 1, 0],
            [1, 1, 0]
            ]

TrainingClass = [1, 1, 1, -1, 1, 1, -1, 1 ,-1 ,-1]

#                 A   B   C
ValidationAttr =[[[0, 0, 0]],
                [[0, 1, 1]],
                [[1, 1, 0]],
                [[1, 0, 1]],
                [[1, 0, 0]],
                ]

ValidationClass = [1, 1, 1, -1, 1]

classifiers = ['=', '+', '-']

attr = ['A','B','C']

#==============================================================================

# Function to generate the Bagging Ensemble model
def baggingModel():

    # make the model
    clf = BaggingClassifier(base_estimator=None, n_estimators=10, random_state=0)
    clf.fit(TrainingAttr,TrainingClass)

    #get the decision trees that are part of this ensemble
    decisionTrees = getattr(clf, 'estimators_')

    #get the data that was used in each tree
    decisionTreeData = getattr(clf, 'estimators_samples_')

    #print the decision trees along with the classes
    print("Bagging Ensemble Individual Decision Trees:")
    itt = 0
    for x in decisionTrees:

        #print the data that went into each data set
        print("Decision Tree Number: " + str(itt) + "\n")
        print("[A, B, C] [Class]")
        for j in range(10):

            #format attributes into printable set
            dset = ", ".join(str(m) for m in TrainingAttr[decisionTreeData[itt][j]])

            #print with formatting
            print('[' +  dset +  ']   [' +
                classifiers[TrainingClass[decisionTreeData[itt][j]]] + ']')

        #display the decision tree
        d_data = tree.export_graphviz(x, feature_names=attr,  class_names=['-','+'])
        graph = graphviz.Source(d_data, format="png")
        graph.render('Tree',view=False)
        graph
        display(Image(filename='Tree.png'))
        itt += 1

    #return variables for predictions
    return clf, decisionTrees, decisionTreeData

clf, decisionTrees, decisionTreeData = baggingModel()
```

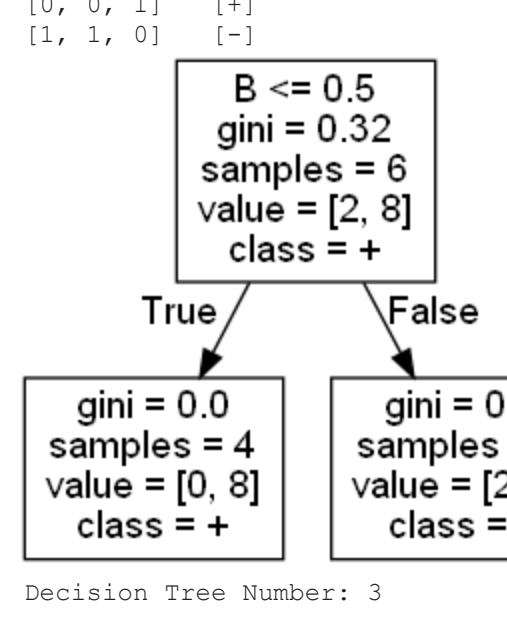
```
Bagging Ensemble Individual Decision Trees:
Decision Tree Number: 0

[A, B, C] [Class]
[0, 1, 1]   [-]
[0, 1, 0]   [+]
[0, 0, 0]   [+]
[0, 0, 0]   [+]
[0, 1, 0]   [+]
[1, 1, 0]   [-]
[1, 1, 0]   [-]
[1, 0, 1]   [+]
[0, 1, 1]   [-]
[1, 1, 0]   [-]
```



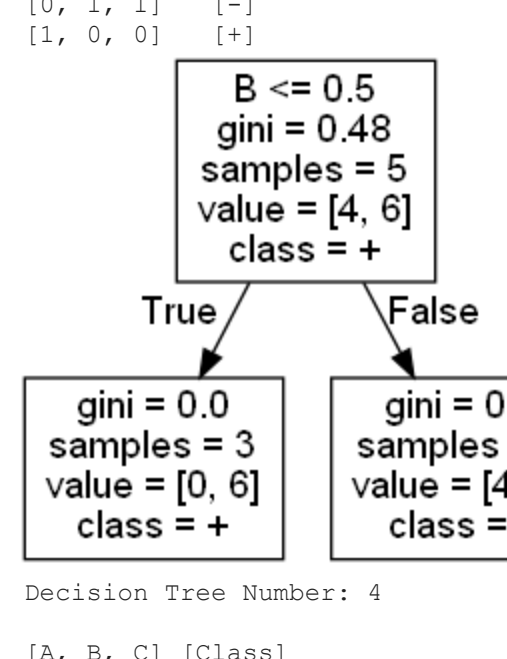

```
Decision Tree Number: 1

[A, B, C] [Class]
[0, 1, 1]   [-]
[0, 0, 0]   [+]
[0, 0, 0]   [+]
[0, 0, 1]   [+]
[1, 1, 0]   [-]
[0, 0, 1]   [+]
[1, 0, 1]   [+]
[1, 1, 0]   [-]
[1, 0, 0]   [+]
[1, 1, 0]   [-]
```



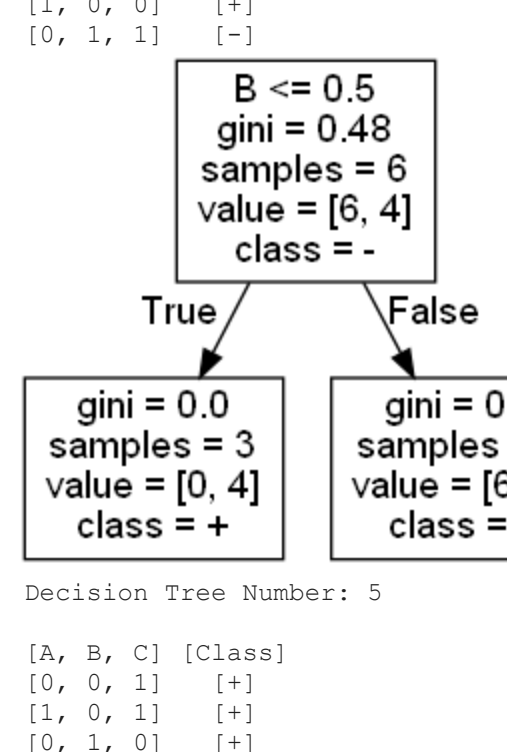

```
Decision Tree Number: 2

[A, B, C] [Class]
[0, 1, 1]   [-]
[1, 0, 0]   [+]
[1, 0, 0]   [+]
[1, 0, 0]   [+]
[1, 0, 1]   [+]
[0, 0, 1]   [+]
[1, 0, 1]   [+]
[1, 0, 1]   [+]
[0, 0, 1]   [+]
[1, 1, 0]   [-]
```



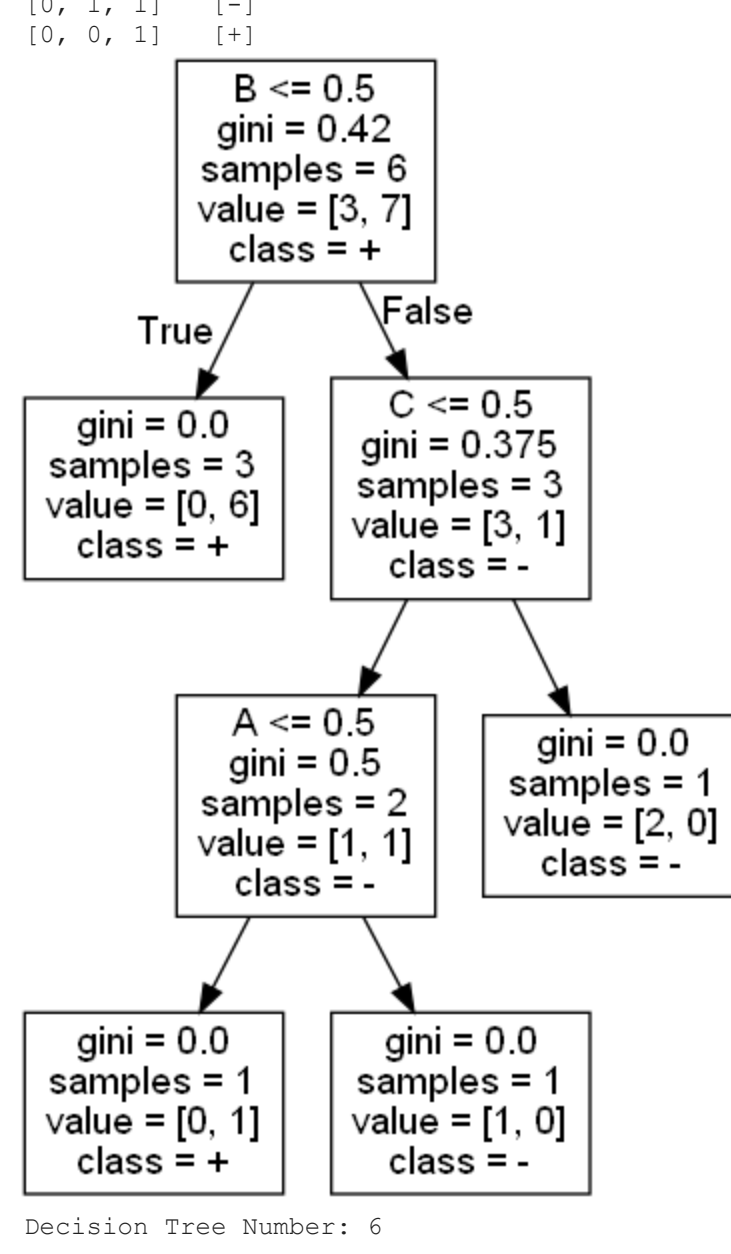

```
Decision Tree Number: 3

[A, B, C] [Class]
[1, 0, 0]   [+]
[1, 0, 0]   [+]
[0, 1, 1]   [-]
[0, 0, 0]   [+]
[1, 1, 0]   [-]
[0, 1, 1]   [-]
[0, 0, 0]   [+]
[1, 0, 0]   [+]
[0, 1, 1]   [-]
[1, 0, 0]   [+]
```



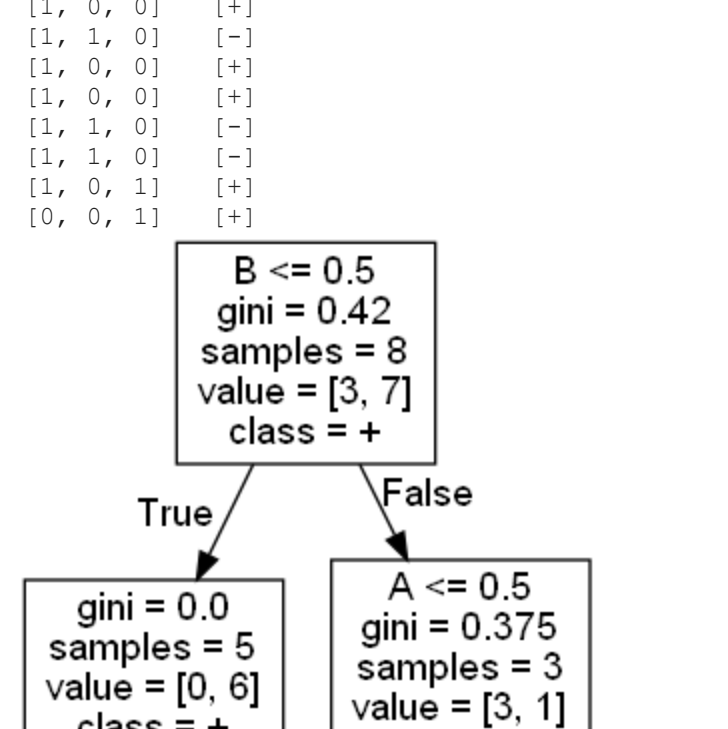

```
Decision Tree Number: 4

[A, B, C] [Class]
[0, 0, 1]   [+]
[1, 0, 1]   [+]
[1, 1, 0]   [-]
[1, 0, 0]   [+]
[0, 1, 1]   [-]
[1, 1, 0]   [-]
[0, 1, 1]   [-]
[1, 1, 0]   [-]
[1, 0, 0]   [+]
[0, 1, 1]   [-]
```



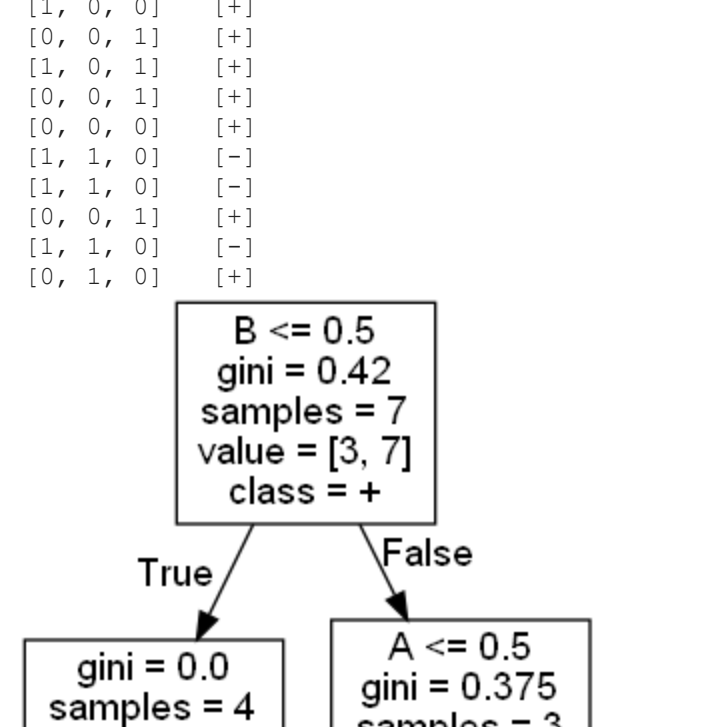

```
Decision Tree Number: 5

[A, B, C] [Class]
[0, 0, 1]   [+]
[1, 0, 1]   [+]
[0, 1, 0]   [+]
[1, 0, 1]   [+]
[0, 0, 0]   [+]
[1, 1, 0]   [-]
[0, 0, 1]   [+]
[0, 1, 1]   [-]
[0, 1, 1]   [-]
[0, 0, 1]   [+]
```



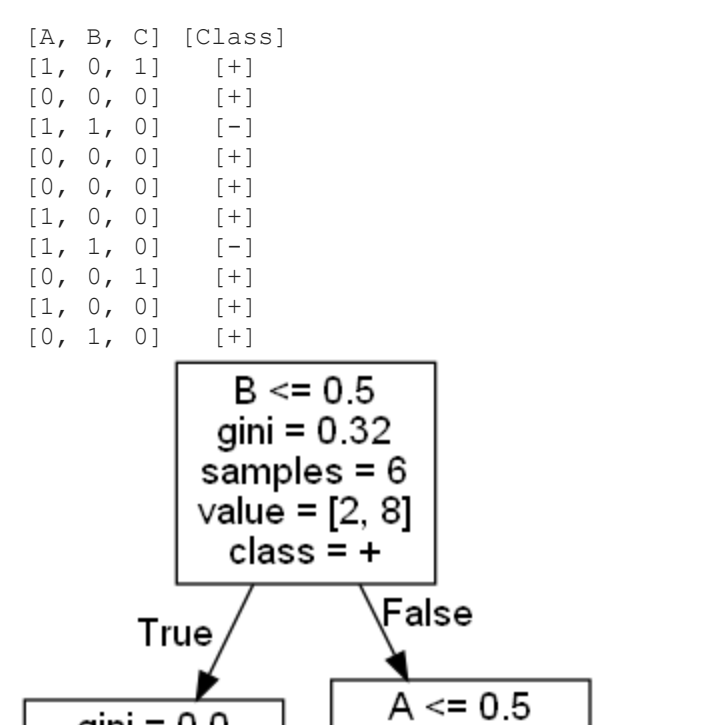

```
Decision Tree Number: 6

[A, B, C] [Class]
[0, 1, 0]   [+]
[0, 0, 0]   [+]
[1, 0, 0]   [+]
[1, 1, 0]   [-]
[1, 0, 0]   [+]
[1, 0, 0]   [+]
[1, 1, 0]   [-]
[1, 1, 0]   [-]
[1, 0, 1]   [+]
[0, 0, 1]   [+]
```



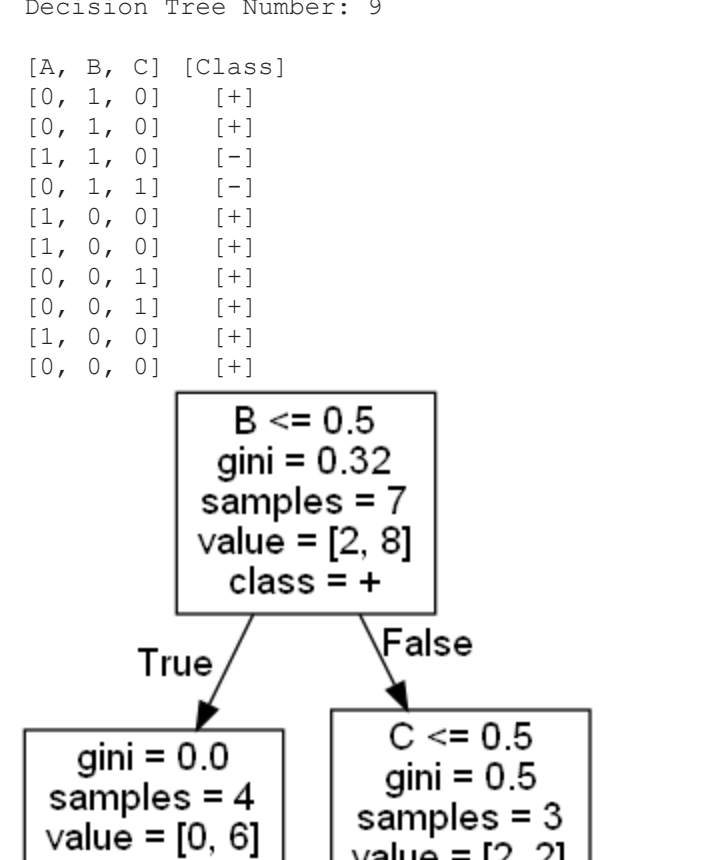

```
Decision Tree Number: 7

[A, B, C] [Class]
[1, 0, 0]   [+]
[0, 0, 1]   [+]
[1, 0, 1]   [+]
[0, 0, 1]   [+]
[0, 0, 0]   [+]
[1, 1, 0]   [-]
[1, 1, 0]   [-]
[0, 0, 1]   [+]
[1, 1, 0]   [-]
[0, 1, 0]   [+]
```




```
Decision Tree Number: 8

[A, B, C] [Class]
[1, 0, 1]   [+]
[0, 0, 0]   [+]
[1, 1, 0]   [-]
[0, 0, 0]   [+]
[0, 0, 0]   [+]
[1, 0, 0]   [+]
[1, 1, 0]   [-]
[0, 0, 1]   [+]
[1, 0, 0]   [+]
[0, 1, 0]   [+]
```




```
Decision Tree Number: 9

[A, B, C] [Class]
[0, 1, 0]   [+]
[0, 1, 0]   [+]
[1, 1, 0]   [-]
[0, 1, 1]   [-]
[1, 0, 0]   [+]
[1, 0, 0]   [+]
[0, 0, 1]   [+]
[0, 0, 1]   [+]
[1, 0, 0]   [+]
[0, 0, 0]   [+]
```



Making Predictions using Bagging

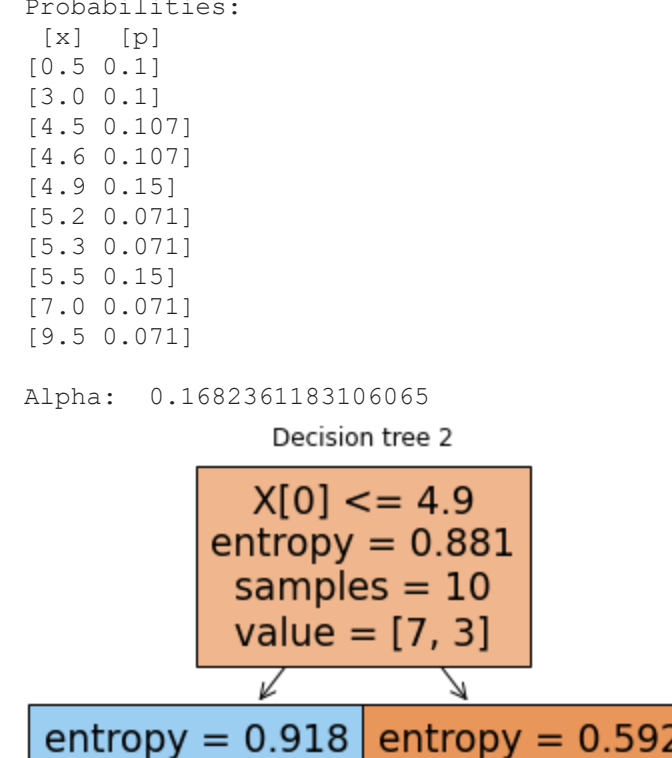
In [3]:
```python
#print the predictions based on the Validation sets
print("Bagging Ensemble Run on Validation Data Set:")
for i in ValidationAttr:
    prediction = clf.predict(np.array(i).reshape(1, -1))
    dset = ", ".join(str(l) for l in i)
    print('Predicted Class of [' + dset + ']: ' + str(prediction[0]))
```

```
Bagging Ensemble Run on Validation Data Set:
Predicted Class of [[0, 0, 0]]: 1
Predicted Class of [[0, 1, 1]]: -1
Predicted Class of [[1, 1, 0]]: -1
Predicted Class of [[1, 0, 1]]: 1
Predicted Class of [[1, 0, 0]]: 1
```

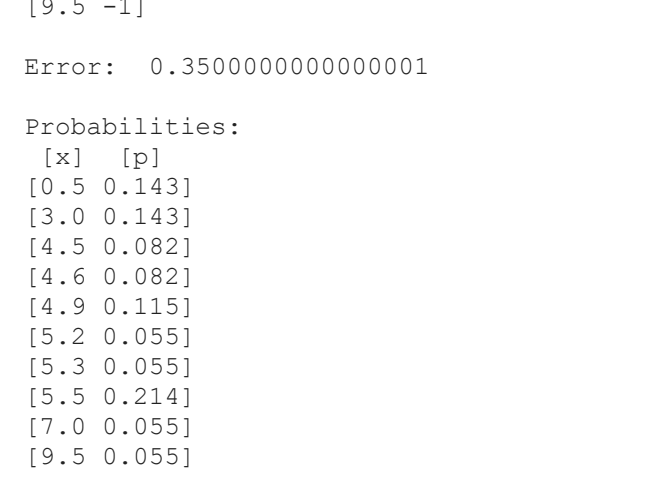
Part 2: Adaboost

In [4]:
```python
# Setting Intial and Validation data sets
x = np.array([0.5,3.0,4.5, 4.6, 4.9, 5.2, 5.3, 5.5, 7.0, 9.5]).reshape(-1, 1)
y = np.array([-1, -1, 1, 1, 1, -1, -1, 1, -1, -1]).reshape(-1, 1)
validation = np.array([1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0])
validation = validation.reshape(-1, 1)

#Setting up initial weights
n = len(x)
weights = np.array([1/n for i in x])
weights_int = np.array([1/n for i in x])

#Creating Decision Stump
Tree_model = DecisionTreeClassifier(criterion="entropy",max_depth=1)

#Inital Values for iterating
iteration = 1
iteration_max = 20;

#Empty Lists to store each model and the corresponding alpha score
alphas = []
models = []

#While loop to find best model using AdaBoost Alogrith
while iteration < iteration_max:
    # Create a training dataset, D(t)
    # Using weights for distribution, get numbers for sampling with replacement
    myList = np.random.choice(np.arange(0, n), n, p=weights)
    trainingX = np.array([x[i] for i in myList]).reshape(-1, 1)
    trainingY = np.array([y[i] for i in myList]).reshape(-1, 1)

    # Create model M(t)
    model = Tree_model.fit(trainingX,trainingY,sample_weight=None)
    models.append(model)

    # Calculate the error of M(t) on D
    prediction = model.predict(x)
    error = 0

    for m, pred in enumerate(prediction):
        if prediction[m] != y[m]:
            error += weights[m]

    # Find model M(t)'s weight
    alpha = 0.5*(math.log((1-error)/error))
    alphas.append(alpha)

    # update the probabilities of the records in D
    if error < 0.5:
        for j, pred in enumerate(prediction):
            if pred == y[j]:
                weights[j] = weights[j]/(2*(1-error))
            else:
                weights[j] = weights[j]/(2*error)

        #Normalizing Error
        weights = weights/sum(weights)

    else:
        #Resetting Weights
        weights = weights_int.copy()

    #print the table, error, and alpha for each model
    print("Decision Tree Number: " + str(iteration) + "\n")
    print("[x] ", "[y]" )
    for l,m in enumerate(x):
        print("[" + str(m[0]) + f" {'' if prediction[l] < 0 else ' '}" +
            str(prediction[l]) + "]")

    print(" ")
    print("Error: ", error)
    print(" ")

    print("Probabilities: ")
    print(" [x] ", "[p]" )
    for l,m in enumerate(weights):
        print("[" + str(x[l][0]) + " " + str(round(m,3)) + "]")

    print(" ")
    print("Alpha: ", alpha)

    #display the decision tree
    plt.figure()
    plot_tree(model, filled=True)
    plt.title("Decision tree " + str(iteration))
    plt.show()
    print("----------------------------")


    #Iteration
    iteration += 1
```

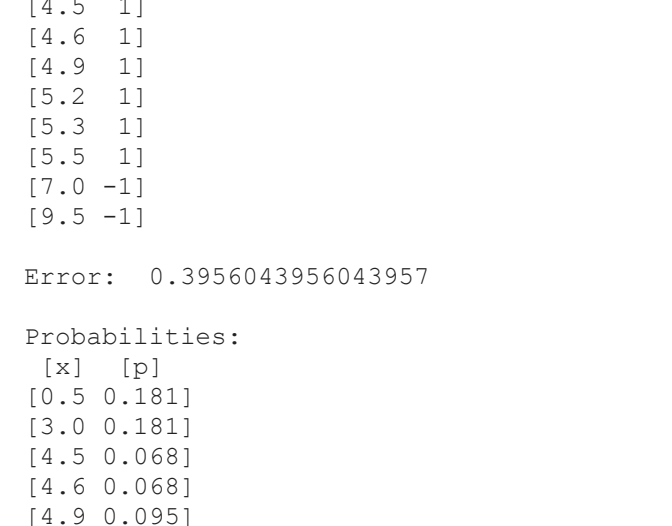
```
Decision Tree Number: 1

[x]  [y]
[0.5 -1]
[3.0 -1]
[4.5 -1]
[4.6 -1]
[4.9 -1]
[5.2 -1]
[5.3 -1]
[5.5 -1]
[7.0 -1]
[9.5 -1]

Error:  0.4

Probabilities:
 [x]  [p]
[0.5 0.083]
[3.0 0.083]
[4.5 0.125]
[4.6 0.125]
[4.9 0.125]
[5.2 0.083]
[5.3 0.083]
[5.5 0.125]
[7.0 0.083]
[9.5 0.083]

Alpha:  0.2027325540540821
```



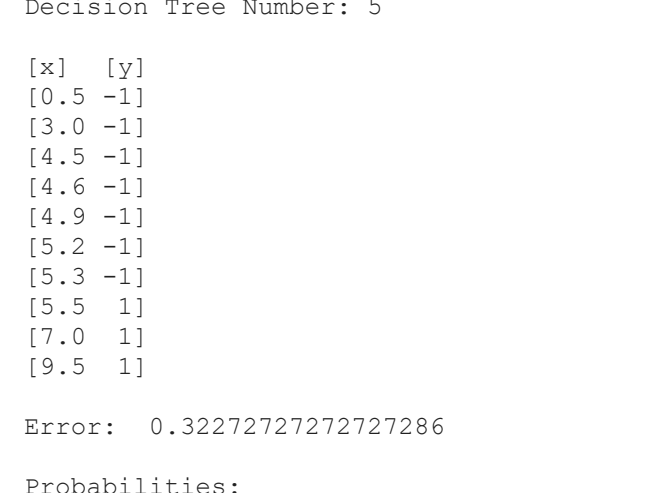

```
----------------------------
Decision Tree Number: 2

[x]  [y]
[0.5  1]
[3.0  1]
[4.5  1]
[4.6  1]
[4.9 -1]
[5.2 -1]
[5.3 -1]
[5.5 -1]
[7.0 -1]
[9.5 -1]

Error:  0.41666666666666663

Probabilities:
 [x]  [p]
[0.5 0.1]
[3.0 0.1]
[4.5 0.107]
[4.6 0.107]
[4.9 0.15]
[5.2 0.071]
[5.3 0.071]
[5.5 0.15]
[7.0 0.071]
[9.5 0.071]

Alpha:  0.1682361183106065
```



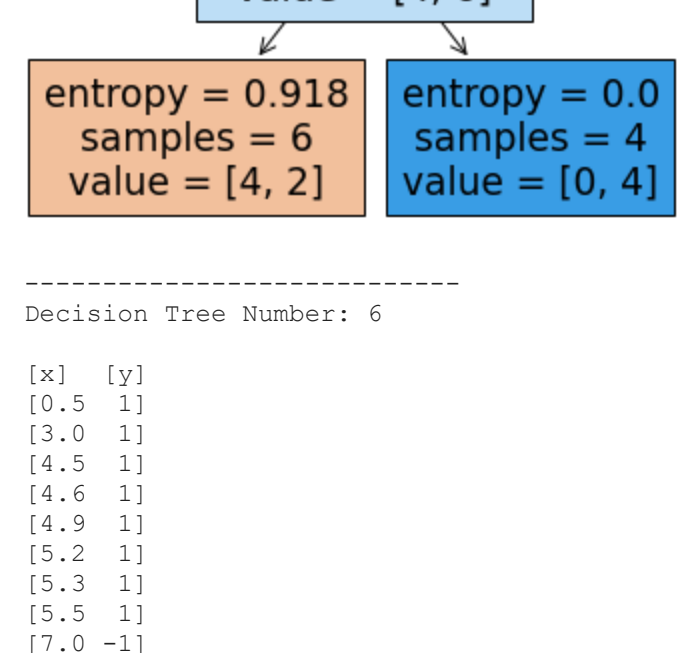

```
----------------------------
Decision Tree Number: 3

[x]  [y]
[0.5  1]
[3.0  1]
[4.5  1]
[4.6  1]
[4.9  1]
[5.2 -1]
[5.3 -1]
[5.5 -1]
[7.0 -1]
[9.5 -1]

Error:  0.3500000000000001

Probabilities:
 [x]  [p]
[0.5 0.143]
[3.0 0.143]
[4.5 0.082]
[4.6 0.082]
[4.9 0.115]
[5.2 0.055]
[5.3 0.055]
[5.5 0.214]
[7.0 0.055]
[9.5 0.055]

Alpha:  0.3095196042031115
```



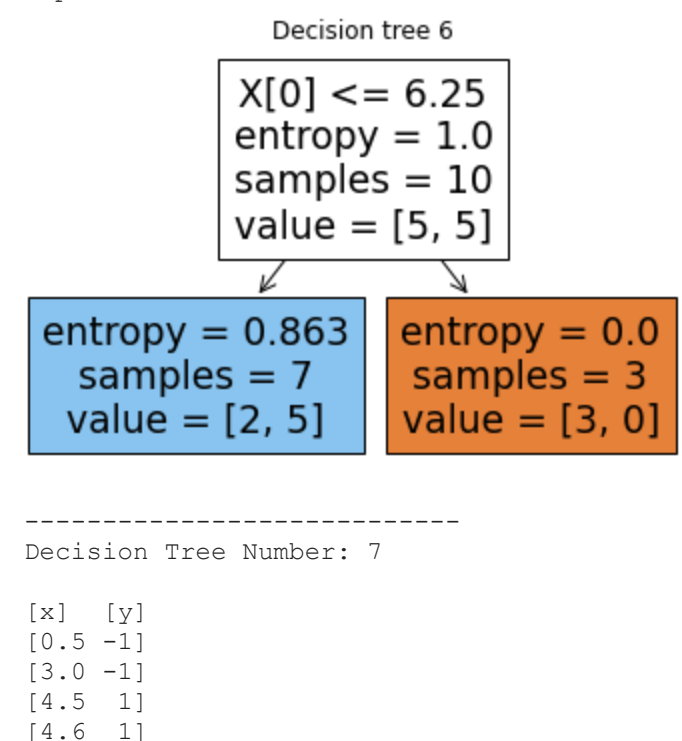

```
----------------------------
Decision Tree Number: 4

[x]  [y]
[0.5  1]
[3.0  1]
[4.5  1]
[4.6  1]
[4.9  1]
[5.2  1]
[5.3  1]
[5.5  1]
[7.0 -1]
[9.5 -1]

Error:  0.3956043956043957

Probabilities:
 [x]  [p]
[0.5 0.181]
[3.0 0.181]
[4.5 0.068]
[4.6 0.068]
[4.9 0.095]
[5.2 0.069]
[5.3 0.069]
[5.5 0.177]
[7.0 0.045]
[9.5 0.045]

Alpha:  0.2119071233881802
```



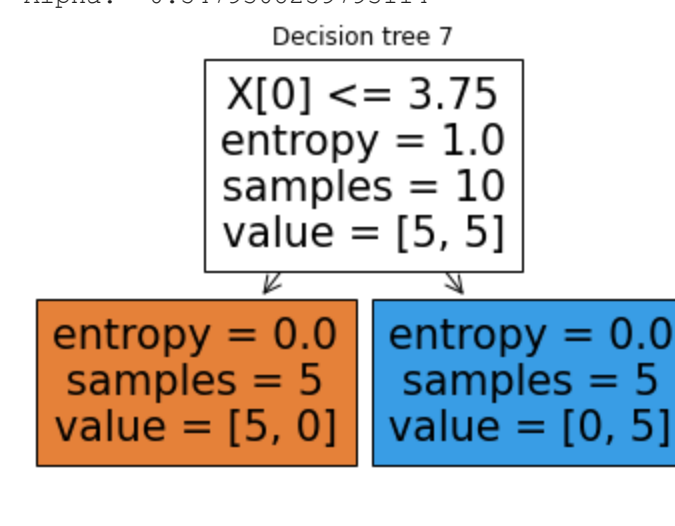

```
----------------------------
Decision Tree Number: 5

[x]  [y]
[0.5 -1]
[3.0 -1]
[4.5 -1]
[4.6 -1]
[4.9 -1]
[5.2 -1]
[5.3 -1]
[5.5  1]
[7.0  1]
[9.5  1]

Error:  0.32272727272727286

Probabilities:
 [x]  [p]
[0.5 0.133]
[3.0 0.133]
[4.5 0.106]
[4.6 0.106]
[4.9 0.148]
[5.2 0.051]
[5.3 0.051]
[5.5 0.131]
[7.0 0.07]
[9.5 0.07]

Alpha:  0.37063321445207154
```




```
----------------------------
Decision Tree Number: 6

[x]  [y]
[0.5  1]
[3.0  1]
[4.5  1]
[4.6  1]
[4.9  1]
[5.2  1]
[5.3  1]
[5.5  1]
[7.0 -1]
[9.5 -1]

Error:  0.36912751677852357

Probabilities:
 [x]  [p]
[0.5 0.181]
[3.0 0.181]
[4.5 0.084]
[4.6 0.084]
[4.9 0.117]
[5.2 0.069]
[5.3 0.069]
[5.5 0.104]
[7.0 0.056]
[9.5 0.056]

Alpha:  0.2679807985187664
```

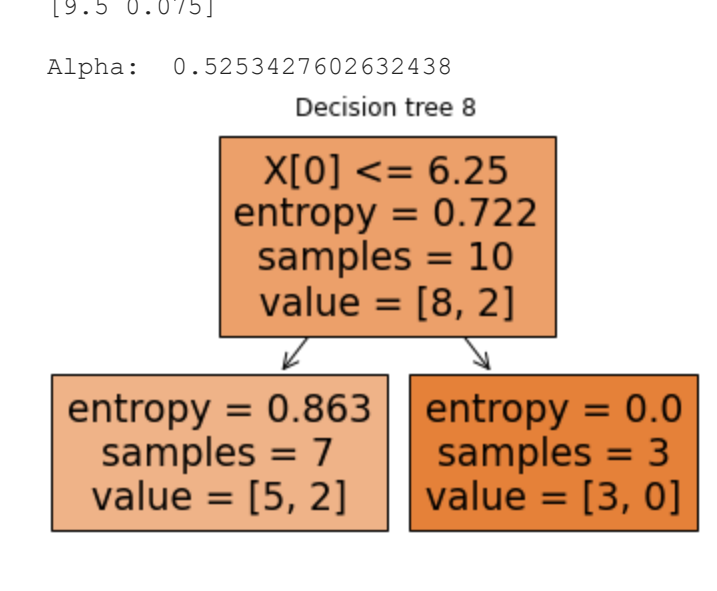



```
----------------------------
Decision Tree Number: 7

[x]  [y]
[0.5 -1]
[3.0 -1]
[4.5  1]
[4.6  1]
[4.9  1]
[5.2  1]
[5.3  1]
[5.5  1]
[7.0  1]
[9.5  1]

Error:  0.2505160989578131

Probabilities:
 [x]  [p]
[0.5 0.12]
[3.0 0.12]
[4.5 0.056]
[4.6 0.056]
[4.9 0.078]
[5.2 0.139]
[5.3 0.139]
[5.5 0.069]
[7.0 0.111]
[9.5 0.111]

Alpha:  0.5479308259793114
```

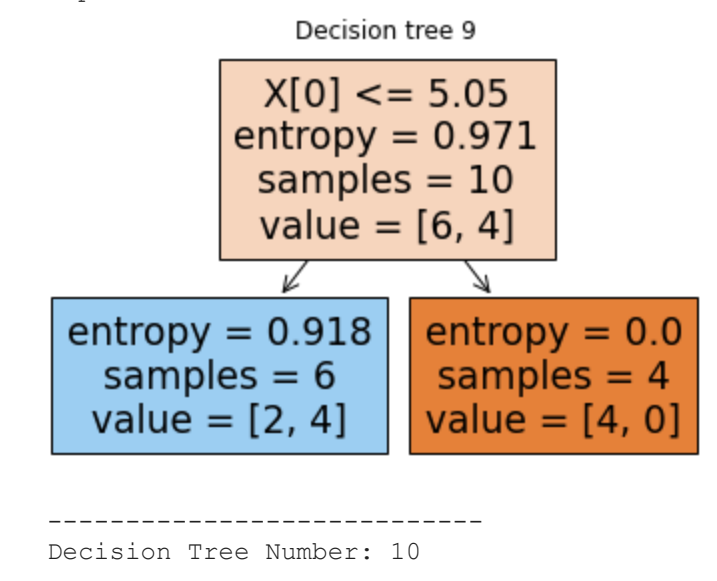



```
----------------------------
```

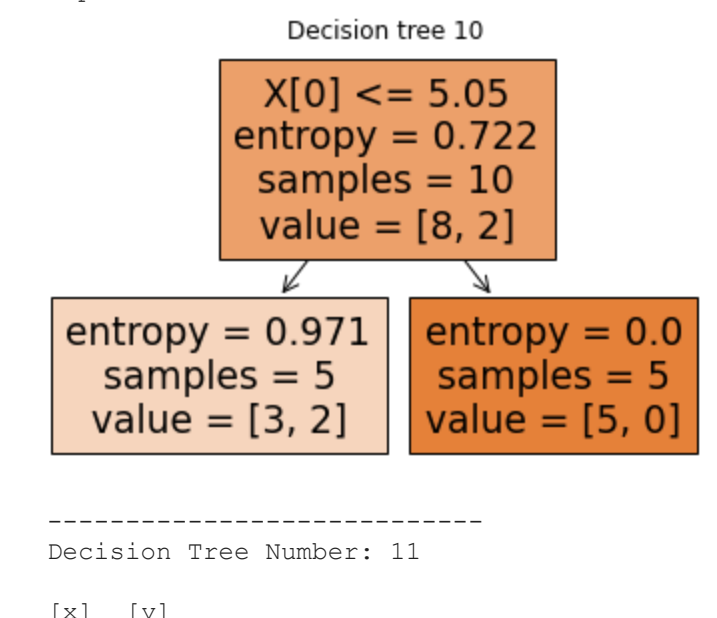
```
Decision Tree Number: 8

[x]  [y]
[0.5 -1]
[3.0 -1]
[4.5 -1]
[4.6 -1]
[4.9 -1]
[5.2 -1]
[5.3 -1]
[5.5 -1]
[7.0 -1]
[9.5 -1]

Error:  0.2590934837899975

Probabilities:
 [x]  [p]
[0.5 0.081]
[3.0 0.081]
[4.5 0.108]
[4.6 0.108]
[4.9 0.151]
[5.2 0.094]
[5.3 0.094]
[5.5 0.134]
[7.0 0.075]
[9.5 0.075]

Alpha:  0.5253427602632438
```



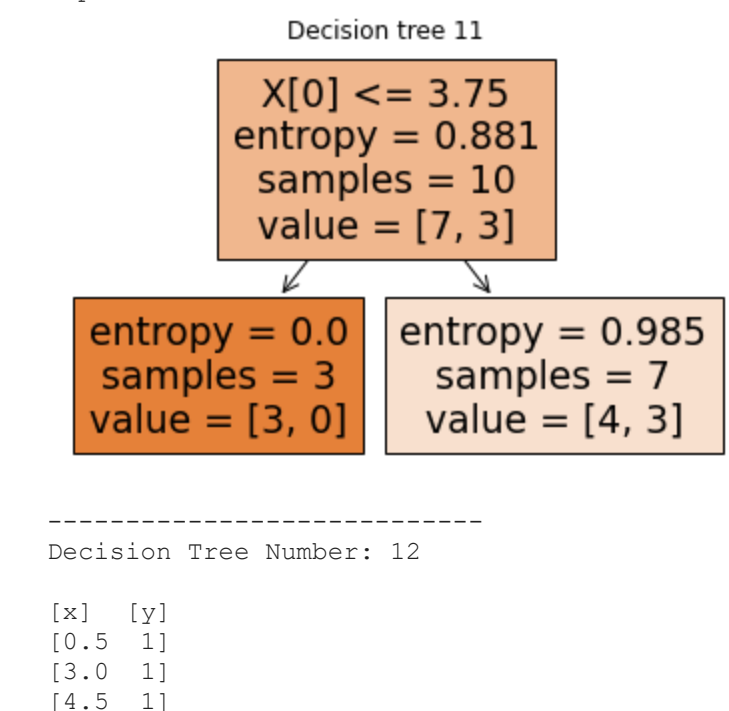

```
----------------------------
Decision Tree Number: 9

[x]  [y]
[0.5  1]
[3.0  1]
[4.5  1]
[4.6  1]
[4.9  1]
[5.2 -1]
[5.3 -1]
[5.5 -1]
[7.0 -1]
[9.5 -1]

Error:  0.2961113941778915

Probabilities:
 [x]  [p]
[0.5 0.137]
[3.0 0.137]
[4.5 0.077]
[4.6 0.077]
[4.9 0.107]
[5.2 0.066]
[5.3 0.066]
[5.5 0.225]
[7.0 0.053]
[9.5 0.053]

Alpha:  0.4329421990304272
```



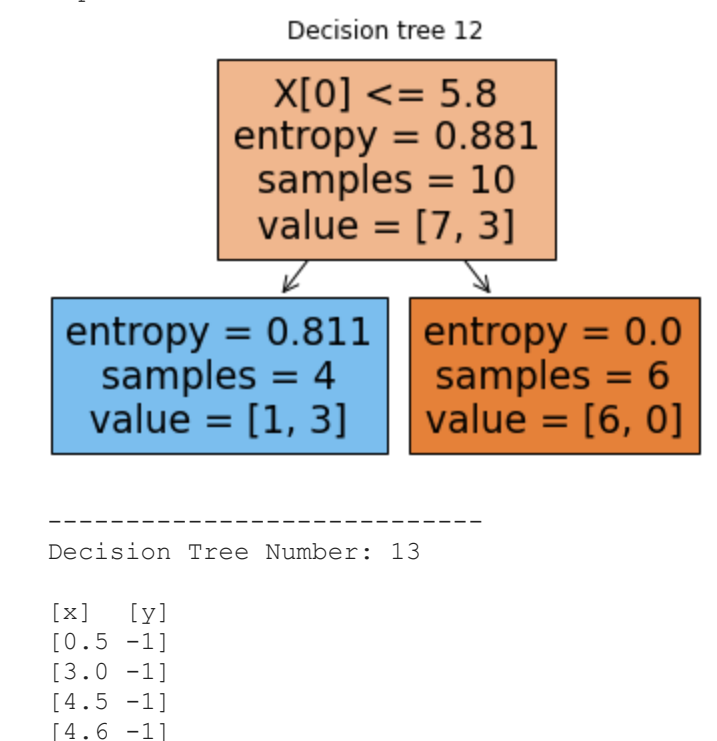

```
----------------------------
Decision Tree Number: 10

[x]  [y]
[0.5 -1]
[3.0 -1]
[4.5 -1]
[4.6 -1]
[4.9 -1]
[5.2 -1]
[5.3 -1]
[5.5 -1]
[7.0 -1]
[9.5 -1]

Error:  0.4857965103707136

Probabilities:
 [x]  [p]
[0.5 0.133]
[3.0 0.133]
[4.5 0.079]
[4.6 0.079]
[4.9 0.11]
[5.2 0.065]
[5.3 0.065]
[5.5 0.232]
[7.0 0.052]
[9.5 0.052]

Alpha:  0.028414624025551817
```



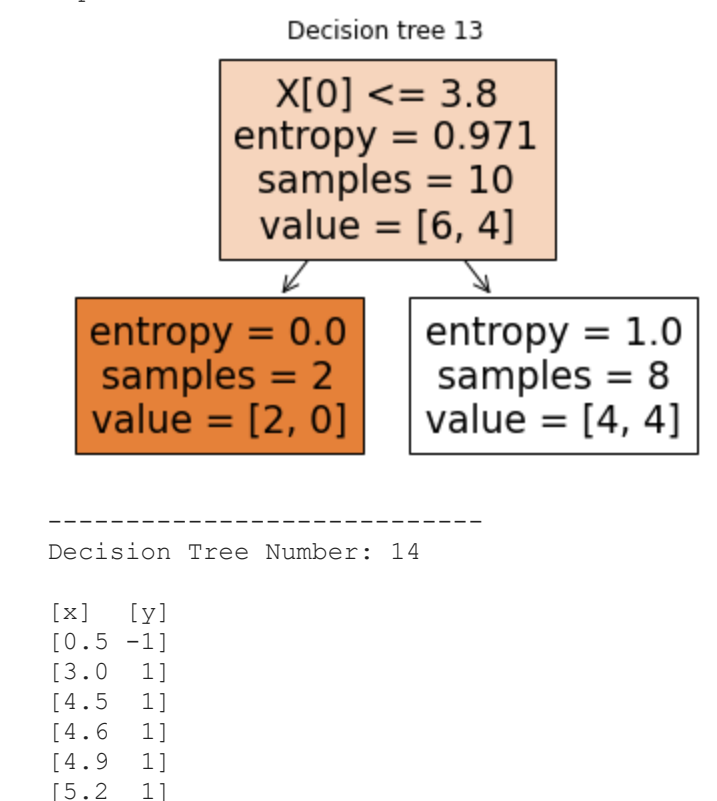

```
----------------------------
Decision Tree Number: 11

[x]  [y]
[0.5 -1]
[3.0 -1]
[4.5 -1]
[4.6 -1]
[4.9 -1]
[5.2 -1]
[5.3 -1]
[5.5 -1]
[7.0 -1]
[9.5 -1]

Error:  0.5

Probabilities:
 [x]  [p]
[0.5 0.1]
[3.0 0.1]
[4.5 0.1]
[4.6 0.1]
[4.9 0.1]
[5.2 0.1]
[5.3 0.1]
[5.5 0.1]
[7.0 0.1]
[9.5 0.1]

Alpha:  0.0
```



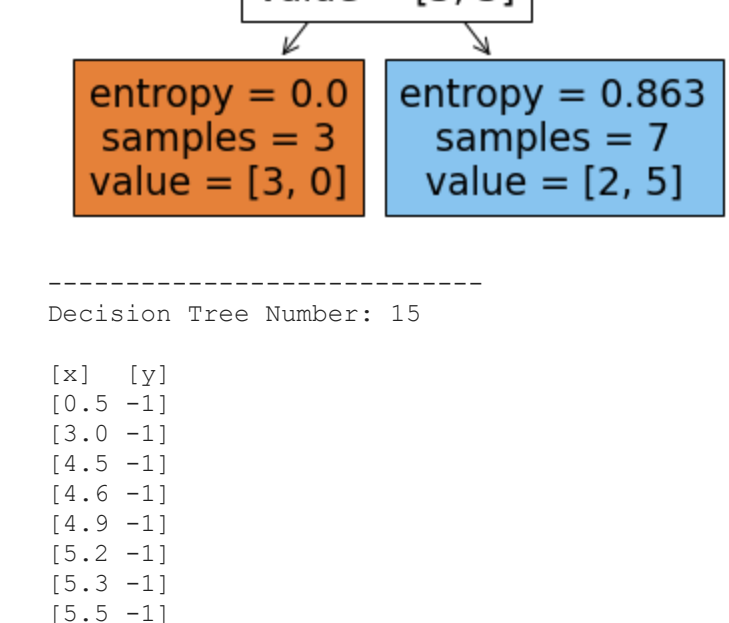

```
----------------------------
Decision Tree Number: 12

[x]  [y]
[0.5  1]
[3.0  1]
[4.5  1]
[4.6  1]
[4.9  1]
[5.2  1]
[5.3  1]
[5.5  1]
[7.0 -1]
[9.5 -1]

Error:  0.4

Probabilities:
 [x]  [p]
[0.5 0.125]
[3.0 0.125]
[4.5 0.083]
[4.6 0.083]
[4.9 0.083]
[5.2 0.125]
[5.3 0.125]
[5.5 0.083]
[7.0 0.083]
[9.5 0.083]

Alpha:  0.2027325540540821
```



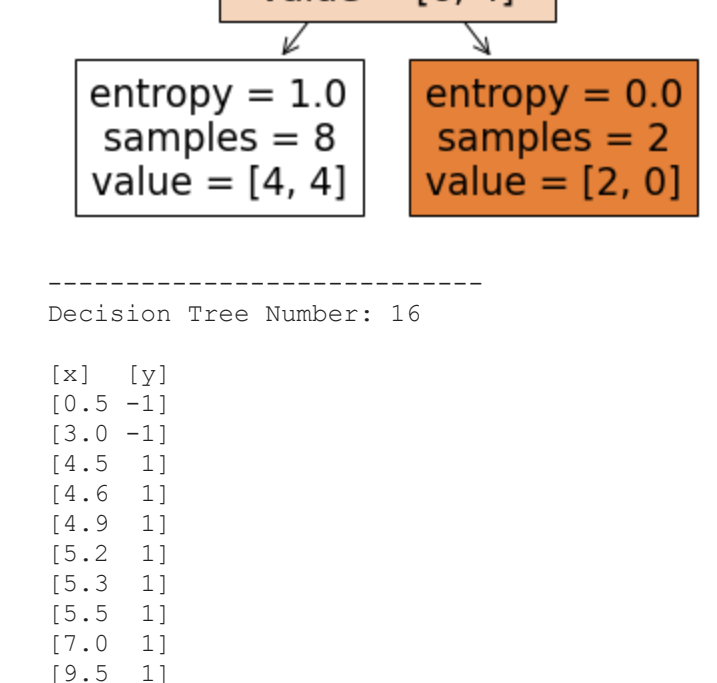

```
----------------------------
Decision Tree Number: 13

[x]  [y]
[0.5 -1]
[3.0 -1]
[4.5 -1]
[4.6 -1]
[4.9 -1]
[5.2 -1]
[5.3 -1]
[5.5 -1]
[7.0 -1]
[9.5 -1]

Error:  0.3333333333333333

Probabilities:
 [x]  [p]
[0.5 0.094]
[3.0 0.094]
[4.5 0.125]
[4.6 0.125]
[4.9 0.125]
[5.2 0.094]
[5.3 0.094]
[5.5 0.125]
[7.0 0.062]
[9.5 0.062]

Alpha:  0.34657359027997275
```



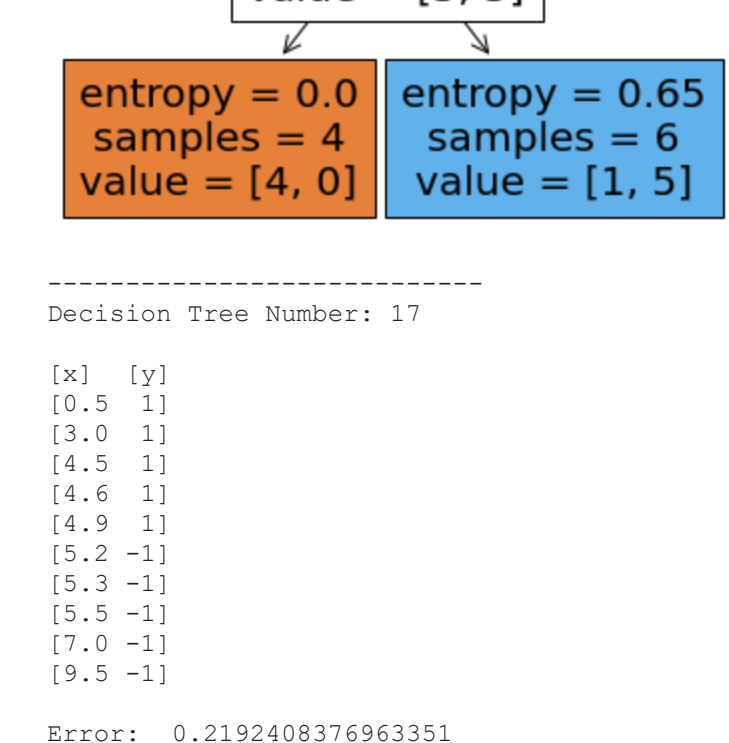

```
----------------------------
Decision Tree Number: 14

[x]  [y]
[0.5 -1]
[3.0  1]
[4.5  1]
[4.6  1]
[4.9  1]
[5.2  1]
[5.3  1]
[5.5  1]
[7.0  1]
[9.5  1]

Error:  0.40624999999999999

Probabilities:
 [x]  [p]
[0.5 0.079]
[3.0 0.115]
[4.5 0.105]
[4.6 0.105]
[4.9 0.105]
[5.2 0.115]
[5.3 0.115]
[5.5 0.105]
[7.0 0.077]
[9.5 0.077]

Alpha:  0.18974481085245207
```



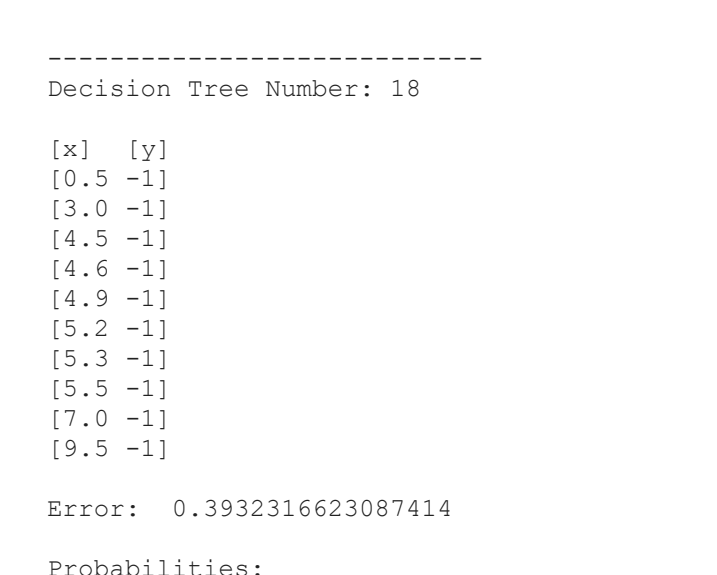

```
----------------------------
Decision Tree Number: 15

[x]  [y]
[0.5 -1]
[3.0 -1]
[4.5 -1]
[4.6 -1]
[4.9 -1]
[5.2 -1]
[5.3 -1]
[5.5 -1]
[7.0 -1]
[9.5 -1]

Error:  0.4210526315789473

Probabilities:
 [x]  [p]
[0.5 0.068]
[3.0 0.1]
[4.5 0.125]
[4.6 0.125]
[4.9 0.125]
[5.2 0.1]
[5.3 0.1]
[5.5 0.125]
[7.0 0.066]
[9.5 0.066]

Alpha:  0.15922686555926738
```



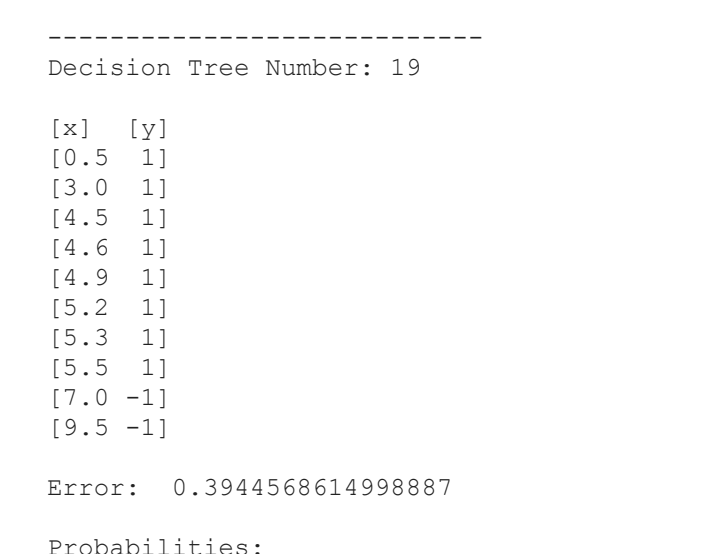

```
----------------------------
Decision Tree Number: 16

[x]  [y]
[0.5 -1]
[3.0 -1]
[4.5  1]
[4.6  1]
[4.9  1]
[5.2  1]
[5.3  1]
[5.5  1]
[7.0  1]
[9.5  1]

Error:  0.3321678321678322

Probabilities:
 [x]  [p]
[0.5 0.051]
[3.0 0.075]
[4.5 0.094]
[4.6 0.094]
[4.9 0.094]
[5.2 0.15]
[5.3 0.15]
[5.5 0.094]
[7.0 0.1]
[9.5 0.1]

Alpha:  0.3491982682230444
```



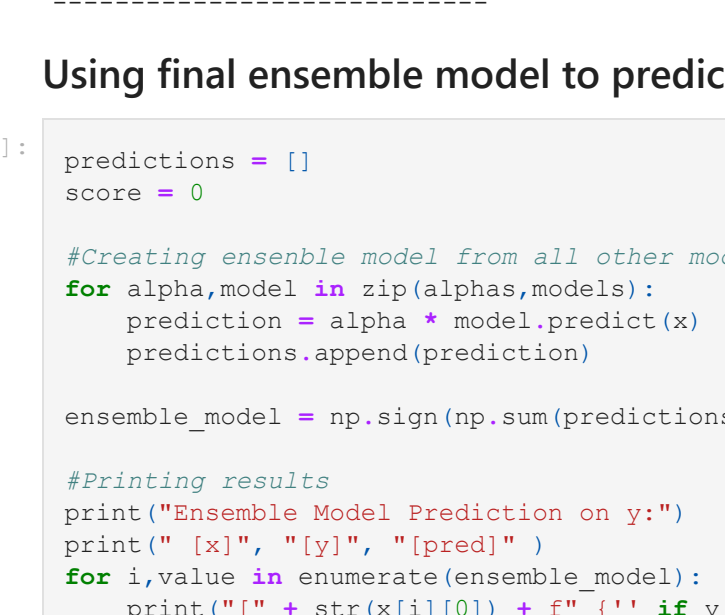

```
----------------------------
Decision Tree Number: 17

[x]  [y]
[0.5  1]
[3.0  1]
[4.5  1]
[4.6  1]
[4.9  1]
[5.2 -1]
[5.3 -1]
[5.5 -1]
[7.0 -1]
[9.5 -1]

Error:  0.2192408376963351

Probabilities:
 [x]  [p]
[0.5 0.116]
[3.0 0.17]
[4.5 0.06]
[4.6 0.06]
[4.9 0.06]
[5.2 0.096]
[5.3 0.096]
[5.5 0.213]
[7.0 0.064]
[9.5 0.064]

Alpha:  0.6350479451364248
```




```
----------------------------
Decision Tree Number: 18

[x]  [y]
[0.5 -1]
[3.0 -1]
[4.5 -1]
[4.6 -1]
[4.9 -1]
[5.2 -1]
[5.3 -1]
[5.5 -1]
[7.0 -1]
[9.5 -1]

Error:  0.3932316623087414

Probabilities:
 [x]  [p]
[0.5 0.096]
[3.0 0.14]
[4.5 0.076]
[4.6 0.076]
[4.9 0.076]
[5.2 0.079]
[5.3 0.079]
[5.5 0.271]
[7.0 0.053]
[9.5 0.053]

Alpha:  0.2168740786197276
```




```
----------------------------
Decision Tree Number: 19

[x]  [y]
[0.5  1]
[3.0  1]
[4.5  1]
[4.6  1]
[4.9  1]
[5.2  1]
[5.3  1]
[5.5  1]
[7.0 -1]
[9.5 -1]

Error:  0.3944568614998887

Probabilities:
 [x]  [p]
[0.5 0.122]
[3.0 0.178]
[4.5 0.063]
[4.6 0.063]
[4.9 0.063]
[5.2 0.1]
[5.3 0.1]
[5.5 0.224]
[7.0 0.044]
[9.5 0.044]

Alpha:  0.21430801019251342
```




```
----------------------------
```

Using final ensemble model to predict X

In [5]:

```
predictions = []
score = 0

#Creating ensenble model from all other models
for alpha,model in zip(alphas,models):
    prediction = alpha * model.predict(x)
    predictions.append(prediction)

ensemble_model = np.sign(np.sum(predictions,axis=0))

#Printing results
print("Ensemble Model Prediction on y:")
print(" [x]", "[y]", "[pred]" )
for i,value in enumerate(ensemble_model):
    print("[" + str(x[i][0]) + f" {'' if y[i] < 0 else ' '}"
          + str(y[i][0]) + f" {'' if value < 0 else ' '}" + str(value) + "]")

    #Computing accuracy
    if y[i] == value:
        score += 1

#Computing accuracy
accuracy = score/len(y)

#Printing Accuracy
print("Accuracy: ", accuracy)
```

```
Ensemble Model Prediction on y:
 [x] [y] [pred]
[0.5 -1  1.0]
[3.0 -1  1.0]
[4.5  1  1.0]
[4.6  1  1.0]
[4.9  1  1.0]
[5.2 -1  1.0]
[5.3 -1  1.0]
[5.5  1  1.0]
[7.0 -1 -1.0]
[9.5 -1 -1.0]
Accuracy:  0.6
```

Using final ensemble model to predict validation set

In [6]:

```
predictions = []

#Creating ensenble model from all other models
for alpha,model in zip(alphas,models):
    prediction = alpha * model.predict(validation)
    predictions.append(prediction)

ensemble_model = np.sign(np.sum(predictions,axis=0))

#Printing Results
print("[x] ", "[y]" )
for l,m in enumerate(zip(validation, ensemble_model)):
    print("[" + str(l) + f" {'' if m[1] < 0 else ' '}" + str(m[1]) + "]")
```

```
[x]  [y]
[0  1.0]
[1  1.0]
[2  1.0]
[3  1.0]
[4  1.0]
[5  1.0]
[6 -1.0]
[7 -1.0]
[8 -1.0]
[9 -1.0]
```