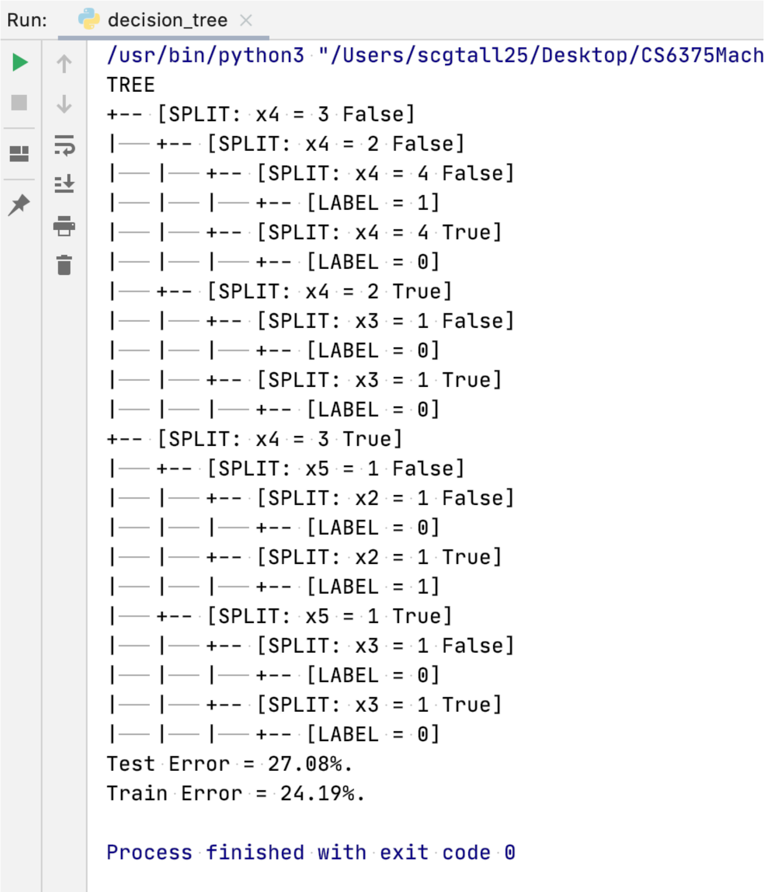
Assignment 2

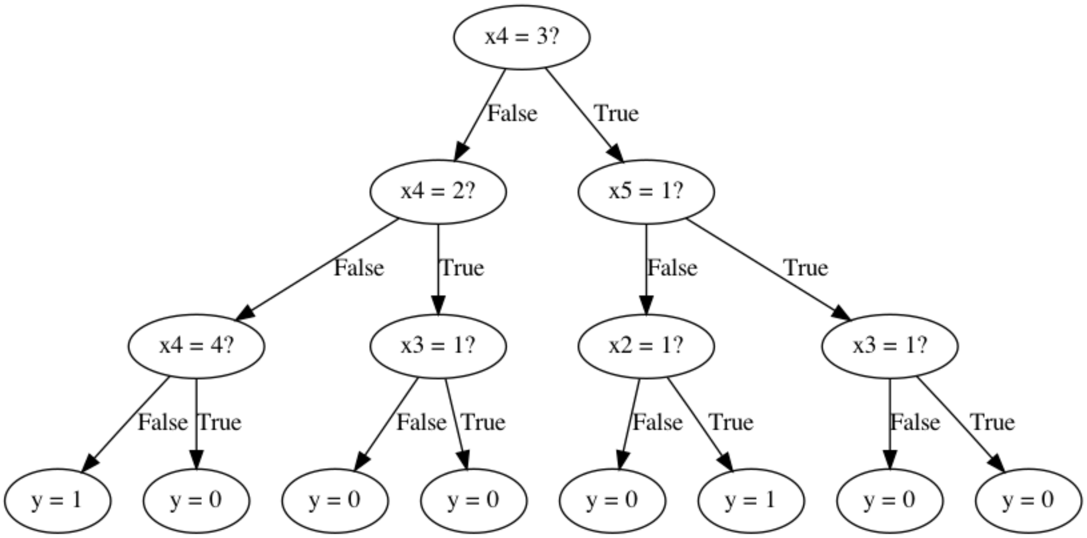
part a

Console Output:



```
/usr/bin/python3 "/Users/scgtall25/Desktop/CS6375Mach
TREE
+-- [SPLIT: x4 = 3 False]
|── +-- [SPLIT: x4 = 2 False]
|── |── +-- [SPLIT: x4 = 4 False]
|── |── |── +-- [LABEL = 1]
|── |── +-- [SPLIT: x4 = 4 True]
|── |── |── +-- [LABEL = 0]
|── +-- [SPLIT: x4 = 2 True]
|── |── +-- [SPLIT: x3 = 1 False]
|── |── |── +-- [LABEL = 0]
|── |── +-- [SPLIT: x3 = 1 True]
|── |── |── +-- [LABEL = 0]
+-- [SPLIT: x4 = 3 True]
|── +-- [SPLIT: x5 = 1 False]
|── |── +-- [SPLIT: x2 = 1 False]
|── |── |── +-- [LABEL = 0]
|── |── +-- [SPLIT: x2 = 1 True]
|── |── |── +-- [LABEL = 1]
|── +-- [SPLIT: x5 = 1 True]
|── |── +-- [SPLIT: x3 = 1 False]
|── |── |── +-- [LABEL = 0]
|── |── +-- [SPLIT: x3 = 1 True]
|── |── |── +-- [LABEL = 0]
Test Error = 27.08%.
Train Error = 24.19%.

Process finished with exit code 0
```

Visualized Learned Decision Tree:

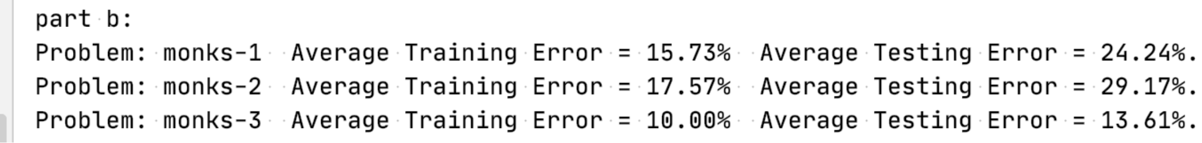


I coded with the given frame and designed my own ID3 Algorithm. In the following part b-e, I will import these functions directly from decision_tree.py.

For part b-e, you should execute Assignment2b-e.py instead.

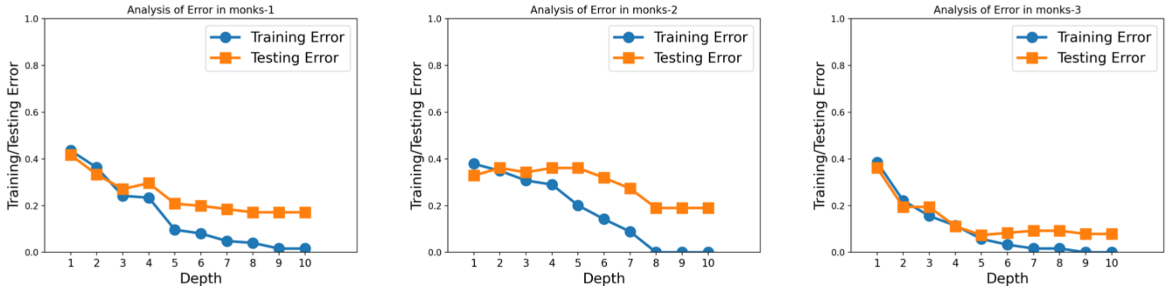
part b

Console Output:



```
part b:
Problem: monks-1  Average Training Error = 15.73%  Average Testing Error = 24.24%.
Problem: monks-2  Average Training Error = 17.57%  Average Testing Error = 29.17%.
Problem: monks-3  Average Training Error = 10.00%  Average Testing Error = 13.61%.
```

Plots implemented by matplotlib:

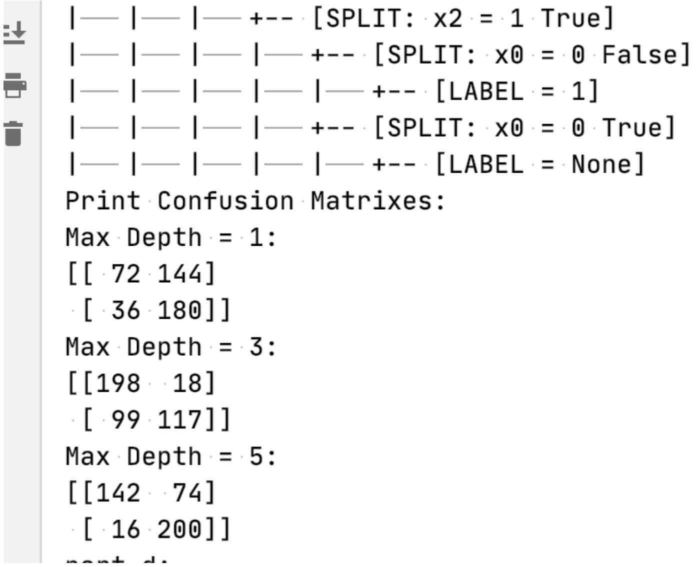


Max Depth	1	3	5
AVG Training Error	15.73%	17.57%	10.00%
AVG Testing Error	24.24%	29.17%	13.61%

From these three plots can we see, it is wise to choose the maximum depth with 3. Because a decision tree with depth more than 3 is often suffered from overfitting.

part c

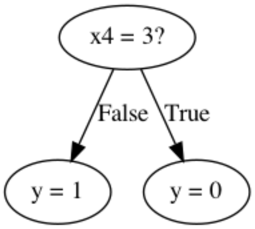
Console Output:



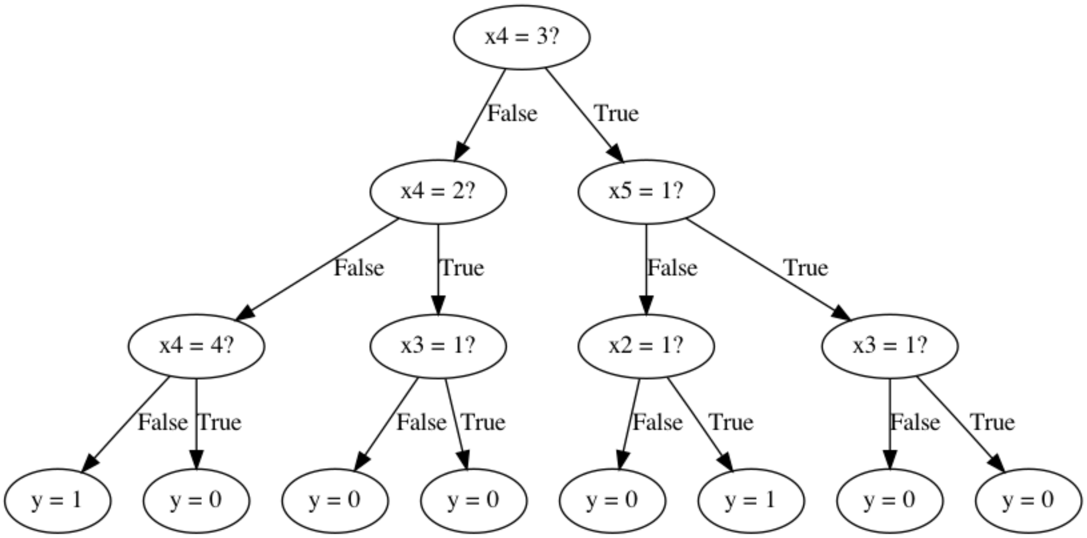
```
|—— |—— |—— +-- [SPLIT: x2 = 1 True]
|—— |—— |—— |—— +-- [SPLIT: x0 = 0 False]
|—— |—— |—— |—— |—— +-- [LABEL = 1]
|—— |—— |—— |—— +-- [SPLIT: x0 = 0 True]
|—— |—— |—— |—— |—— +-- [LABEL = None]
Print Confusion Matrixes:
Max Depth = 1:
[[ 72 144]
 [ 36 180]]
Max Depth = 3:
[[198  18]
 [ 99 117]]
Max Depth = 5:
[[142  74]
 [ 16 200]]
part d:
```

Visualized Learned Decision Tree:

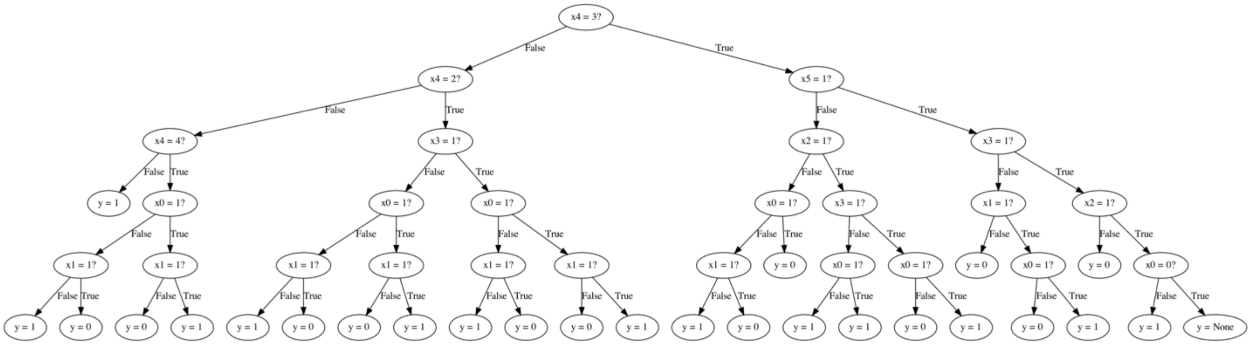
Max Depth = 1



Max Depth = 3

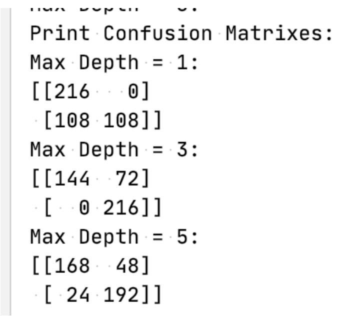


Max Depth = 5



part d

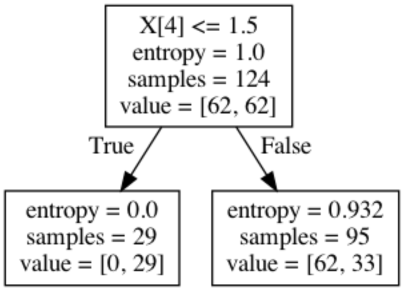
Console Output:



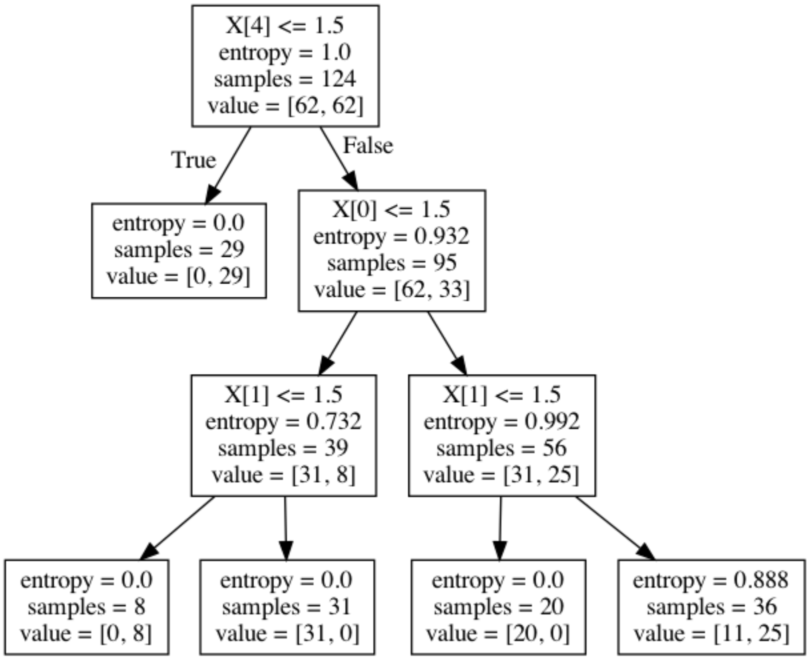
```
Max Depth = 5:
Print Confusion Matrixes:
Max Depth = 1:
[[216   0]
 [108 108]]
Max Depth = 3:
[[144  72]
 [  0 216]]
Max Depth = 5:
[[168  48]
 [ 24 192]]
```

Visualized Learned Decision Tree:

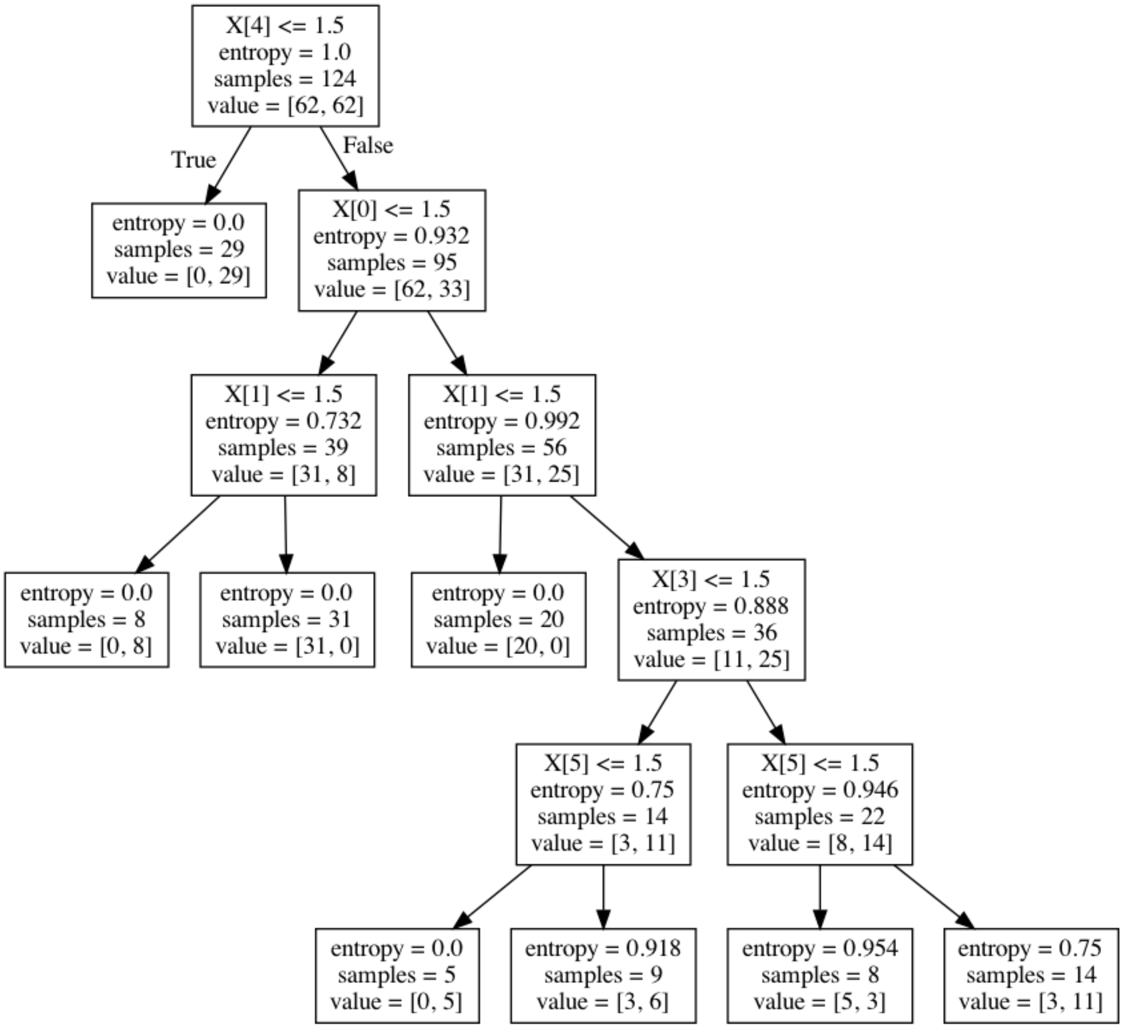
Max Depth = 1



Max Depth = 3


X[4] <= 1.5
entropy = 1.0
samples = 124
value = [62, 62]
True
False
entropy = 0.0
samples = 29
value = [0, 29]
X[0] <= 1.5
entropy = 0.932
samples = 95
value = [62, 33]
X[1] <= 1.5
entropy = 0.732
samples = 39
value = [31, 8]
X[1] <= 1.5
entropy = 0.992
samples = 56
value = [31, 25]
entropy = 0.0
samples = 8
value = [0, 8]
entropy = 0.0
samples = 31
value = [31, 0]
entropy = 0.0
samples = 20
value = [20, 0]
entropy = 0.888
samples = 36
value = [11, 25]

Max Depth = 5

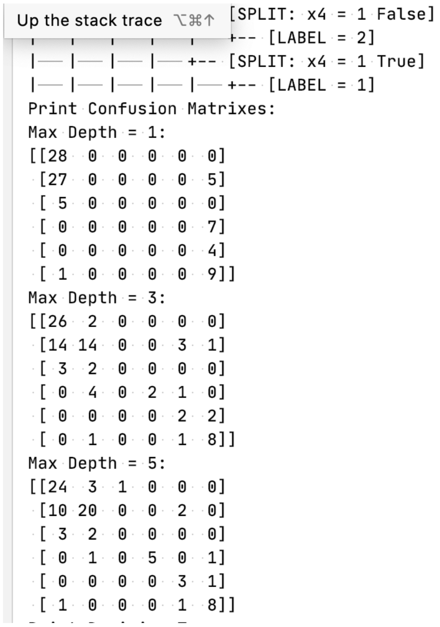


The result I get from part d is slightly different from part c because of the way that I split the nodes. But the overall shapes are same.

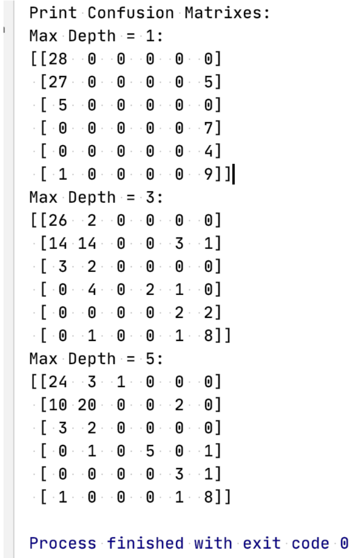
part e

ID3 on the left (top if too large) and Scikit-learn on the right (below).

Console Output:



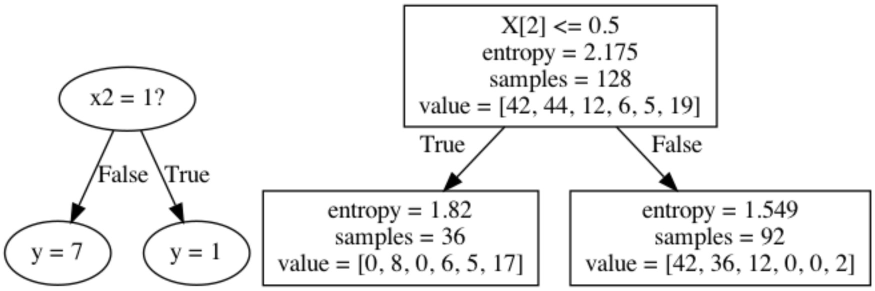
```
[SPLIT: x4 = 1 False]
+-- [LABEL = 2]
|—— |—— |—— |—— +-- [SPLIT: x4 = 1 True]
|—— |—— |—— |—— |—— +-- [LABEL = 1]
Print Confusion Matrixes:
Max Depth = 1:
[[28  0  0  0  0  0]
 [27  0  0  0  0  5]
 [ 5  0  0  0  0  0]
 [ 0  0  0  0  0  7]
 [ 0  0  0  0  0  4]
 [ 1  0  0  0  0  9]]
Max Depth = 3:
[[26  2  0  0  0  0]
 [14 14  0  0  3  1]
 [ 3  2  0  0  0  0]
 [ 0  4  0  2  1  0]
 [ 0  0  0  0  2  2]
 [ 0  1  0  0  1  8]]
Max Depth = 5:
[[24  3  1  0  0  0]
 [10 20  0  0  2  0]
 [ 3  2  0  0  0  0]
 [ 0  1  0  5  0  1]
 [ 0  0  0  0  3  1]
 [ 1  0  0  0  1  8]]
```



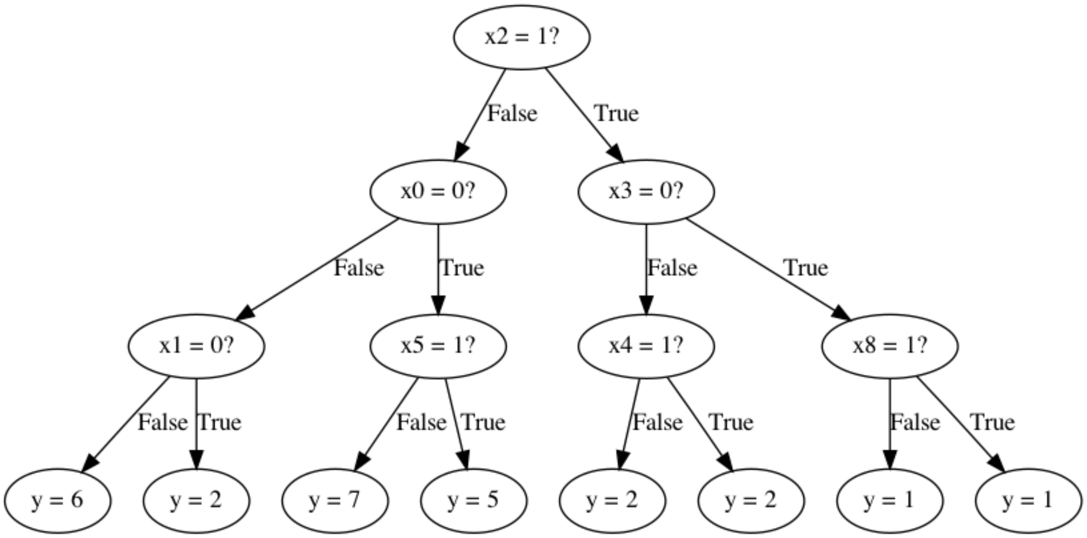
```
Print Confusion Matrixes:
Max Depth = 1:
[[28  0  0  0  0  0]
 [27  0  0  0  0  5]
 [ 5  0  0  0  0  0]
 [ 0  0  0  0  0  7]
 [ 0  0  0  0  0  4]
 [ 1  0  0  0  0  9]]
Max Depth = 3:
[[26  2  0  0  0  0]
 [14 14  0  0  3  1]
 [ 3  2  0  0  0  0]
 [ 0  4  0  2  1  0]
 [ 0  0  0  0  2  2]
 [ 0  1  0  0  1  8]]
Max Depth = 5:
[[24  3  1  0  0  0]
 [10 20  0  0  2  0]
 [ 3  2  0  0  0  0]
 [ 0  1  0  5  0  1]
 [ 0  0  0  0  3  1]
 [ 1  0  0  0  1  8]]

Process finished with exit code 0
```

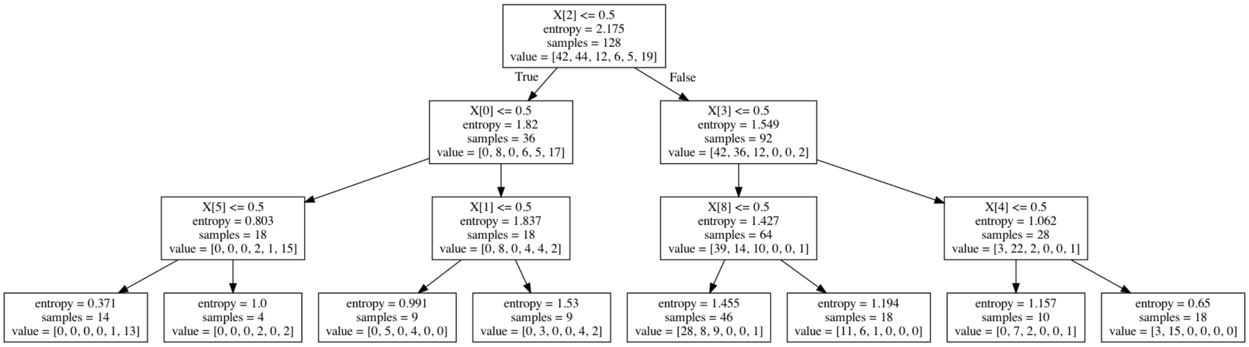
Visualized Learned Decision Tree:

Max Depth = 1

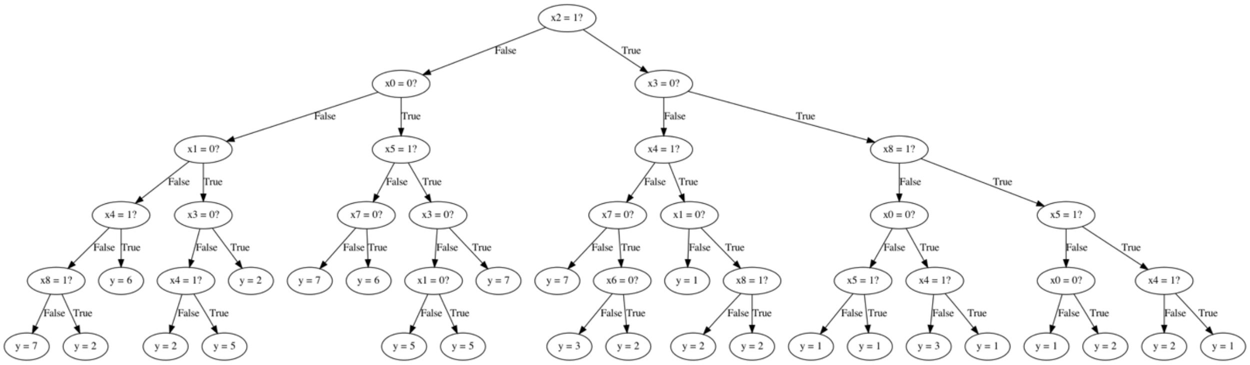


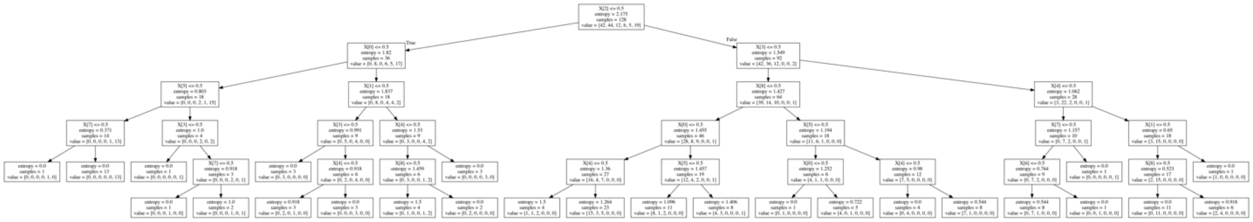
Max Depth = 3





Max Depth = 5





At the last part, I use the Glass Identification Data Set from UCI. I did not want to make y which had 7 different values into 0 and 1. I think that way is illogical. So I optimized a little with my code in decision_tree.py and made it work by only transform xi's values into 0 and 1.

I did not find a way to make Scikit-learn use equal to split. Hence, the learned decision trees obtained by ID3 have some differences from the ones by Scikit-learn. However, if you observe the confusion matrix, you would find the matrixes are similar correspondingly. I think this proved the rationality of decision tree learning algorithm.