CS 6375.001 Machine Learning

By

Prof. Anjum Chida

ASSIGNMENT – 1

Fixed-Length Decision Tree

VEDANT PARESH SHAH VXS200021

Problem: Implement a fixed-depth decision tree algorithm, that is, the input to the ID3 algorithm will include the training data and maximum depth of the tree to be learned. The code skeleton as well as data sets for this assignment can be found on e-Learning.

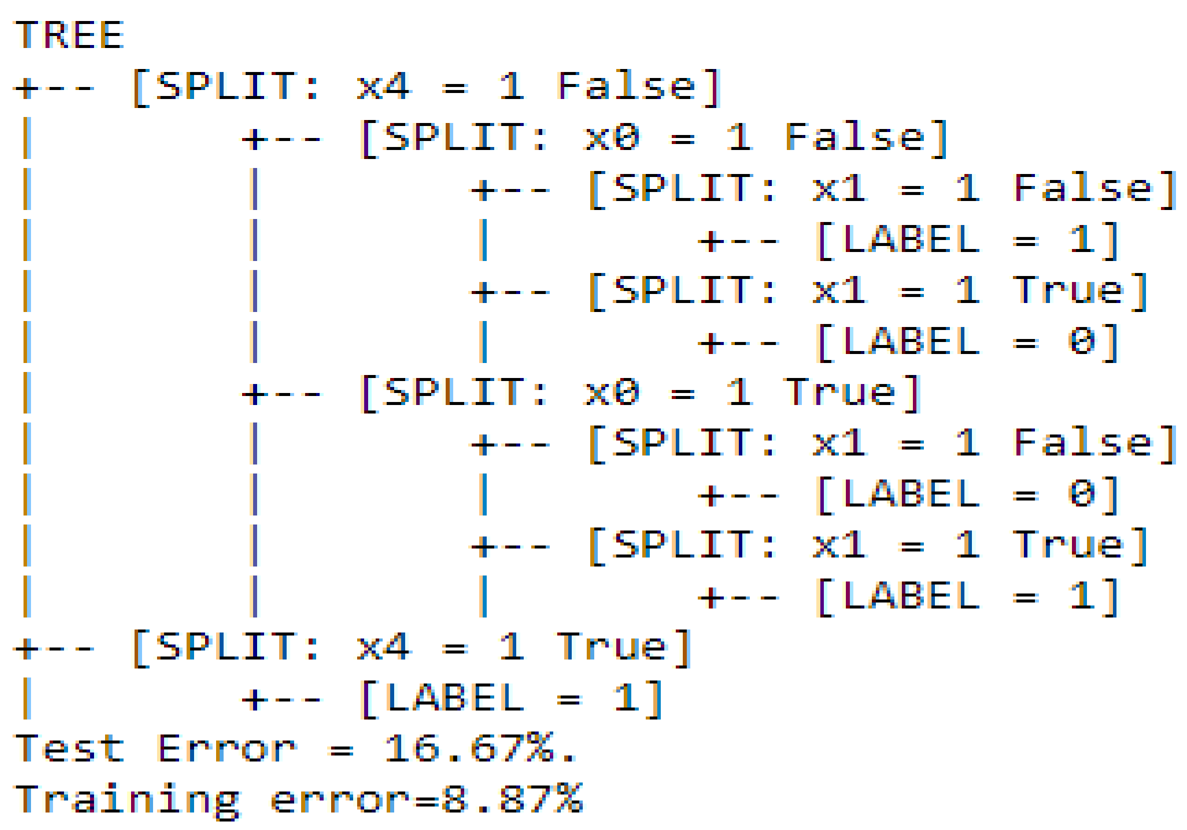
Data Sets: The MONK's Problems were the basis of a first international comparison of learning algorithms [1]. The training and test files for the three problems are named monks-X.train and monks-X.test. There are six attributes/features (columns 2–7), and binary class labels (column 1). See monks.names for more details.

Visualization: The code skeleton provided contains a function render_dot_file(), which can be used to generate .png images of the trees learned by both scikit-learn and your code. See the documentation for render_dot_file() for additional details on usage.

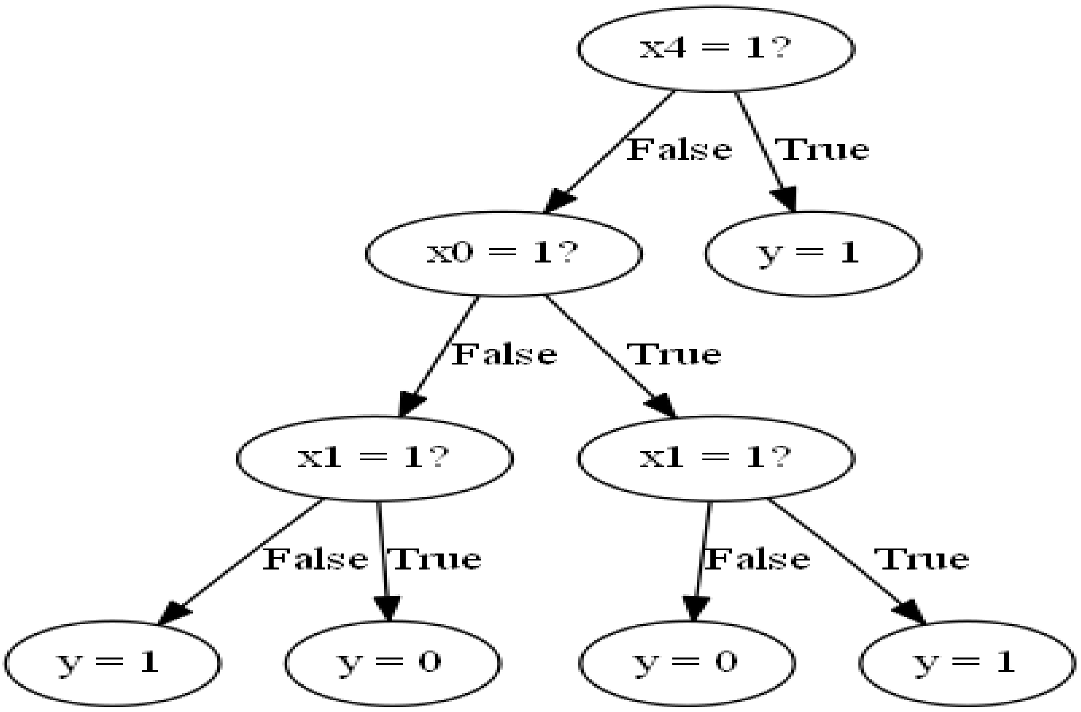
a. (Autograder Score, 20 points) Your code will be auto-graded and cross-checked with other submissions. The auto-grader will evaluate your code on several different data sets to perform a sanity check. In order to ensure that your code passes the auto-grader, ensure that you do not modify the function headers. In addition, do not hard code any values (such as y = 0 and 1) and make your code as general as possible.

Answer:

Monk-1 Dataset:

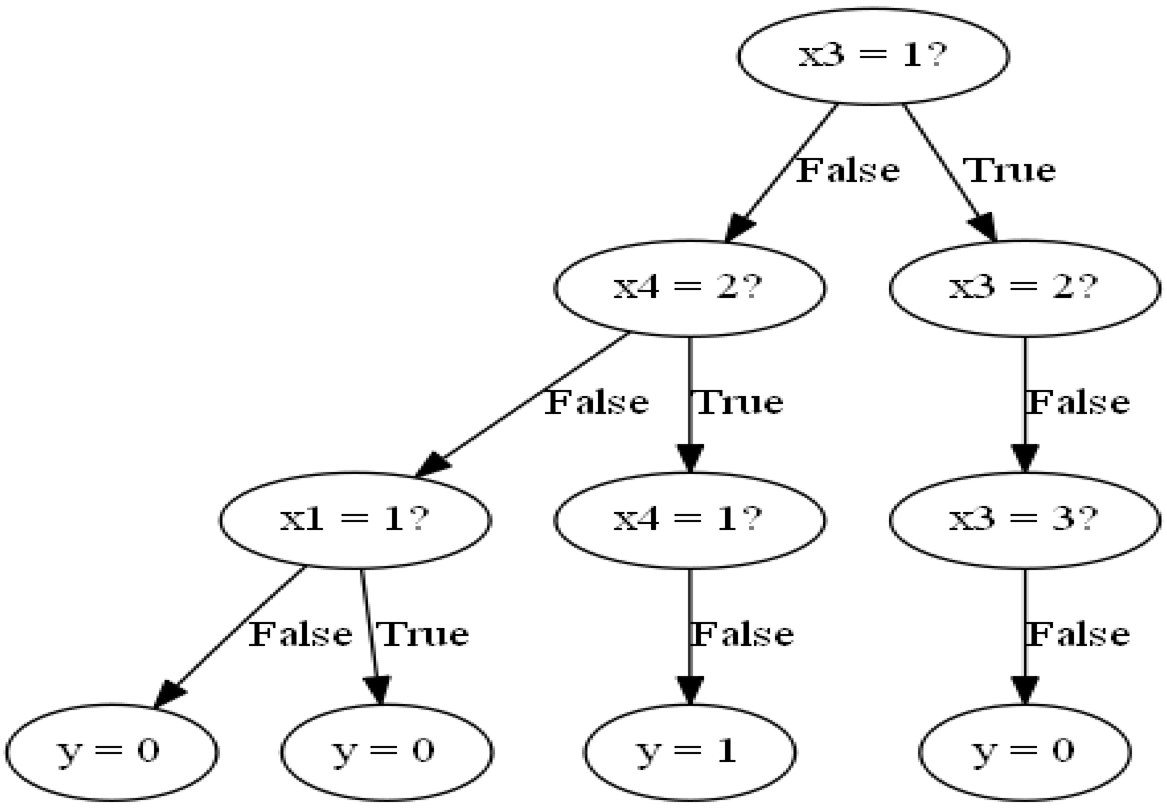


```
TREE
+-- [SPLIT: x4 = 1 False]
|       +-- [SPLIT: x0 = 1 False]
|       |       +-- [SPLIT: x1 = 1 False]
|       |       |       +-- [LABEL = 1]
|       |       +-- [SPLIT: x1 = 1 True]
|       |       |       +-- [LABEL = 0]
|       +-- [SPLIT: x0 = 1 True]
|       |       +-- [SPLIT: x1 = 1 False]
|       |       |       +-- [LABEL = 0]
|       |       +-- [SPLIT: x1 = 1 True]
|       |       |       +-- [LABEL = 1]
+-- [SPLIT: x4 = 1 True]
|       +-- [LABEL = 1]
Test Error = 16.67%.
Training error=8.87%
```



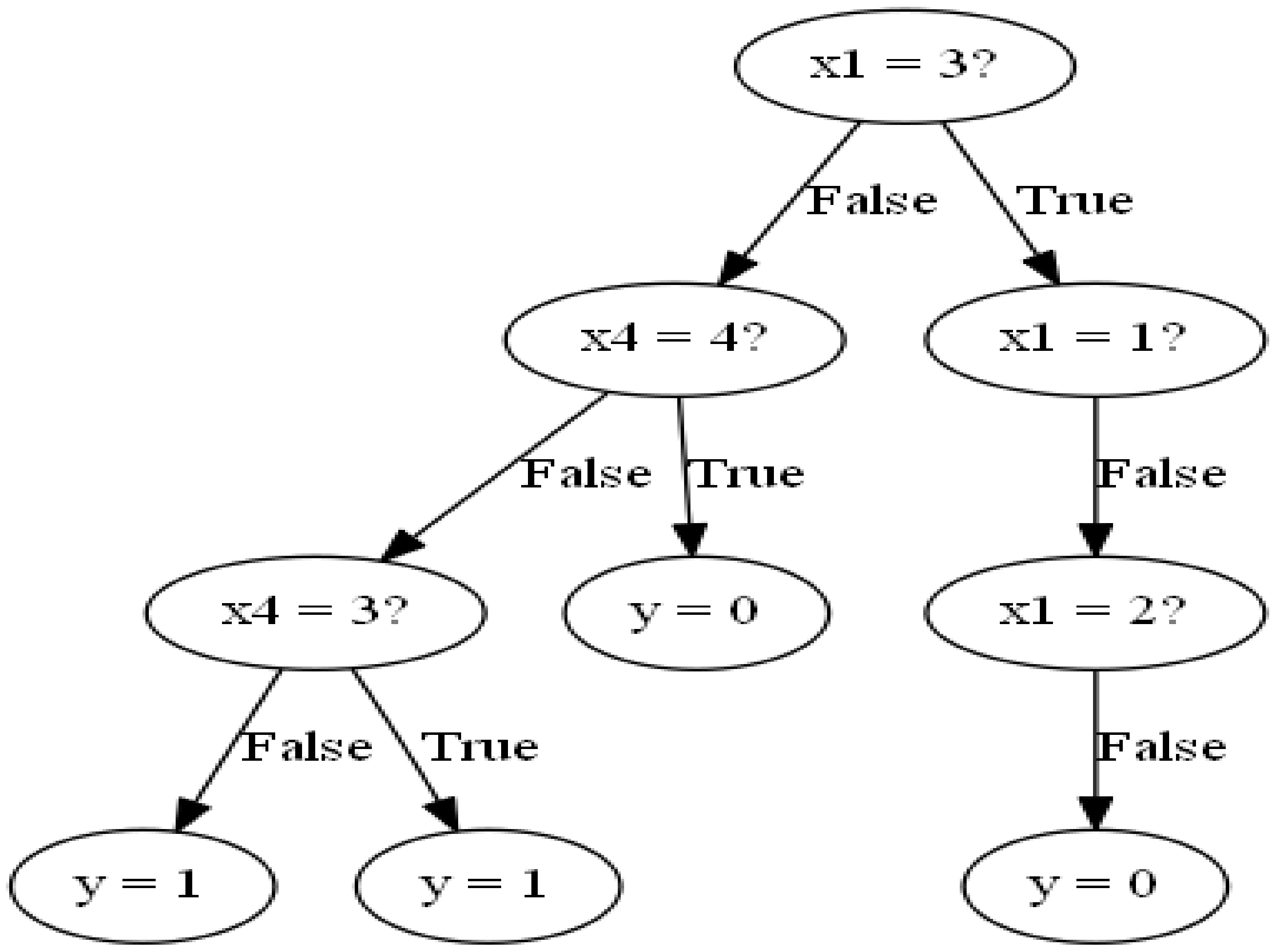
Monk-2 Dataset:

```
TREE
+-- [SPLIT: x3 = 1 False]
|       +-- [SPLIT: x4 = 2 False]
|       |       +-- [SPLIT: x1 = 1 False]
|       |       |       +-- [LABEL = 0]
|       |       +-- [SPLIT: x1 = 1 True]
|       |       |       +-- [LABEL = 0]
|       +-- [SPLIT: x4 = 2 True]
|       |       +-- [SPLIT: x4 = 1 False]
|       |       |       +-- [LABEL = 1]
+-- [SPLIT: x3 = 1 True]
|       +-- [SPLIT: x3 = 2 False]
|       |       +-- [SPLIT: x3 = 3 False]
|       |       |       +-- [LABEL = 0]
Test Error = 37.50%.
Training error=36.09%
```



Monk-3 Dataset:

```
TREE
+--- [SPLIT: x1 = 3 False]
|        +--- [SPLIT: x4 = 4 False]
|        |        +--- [SPLIT: x4 = 3 False]
|        |        |        +--- [LABEL = 1]
|        |        +--- [SPLIT: x4 = 3 True]
|        |        |        +--- [LABEL = 1]
|        +--- [SPLIT: x4 = 4 True]
|        |        +--- [LABEL = 0]
+--- [SPLIT: x1 = 3 True]
|        +--- [SPLIT: x1 = 1 False]
|        |        +--- [SPLIT: x1 = 2 False]
|        |        |        +--- [LABEL = 0]
Test Error = 2.78%.
Training error=6.56%
```



Observation:

Here we have created Decision trees for 3 Datasets Monk-1, Monk-2, and Monk-3. We have shown Decision trees in both Formats with Testing Error and Training Error of above given Datasets.

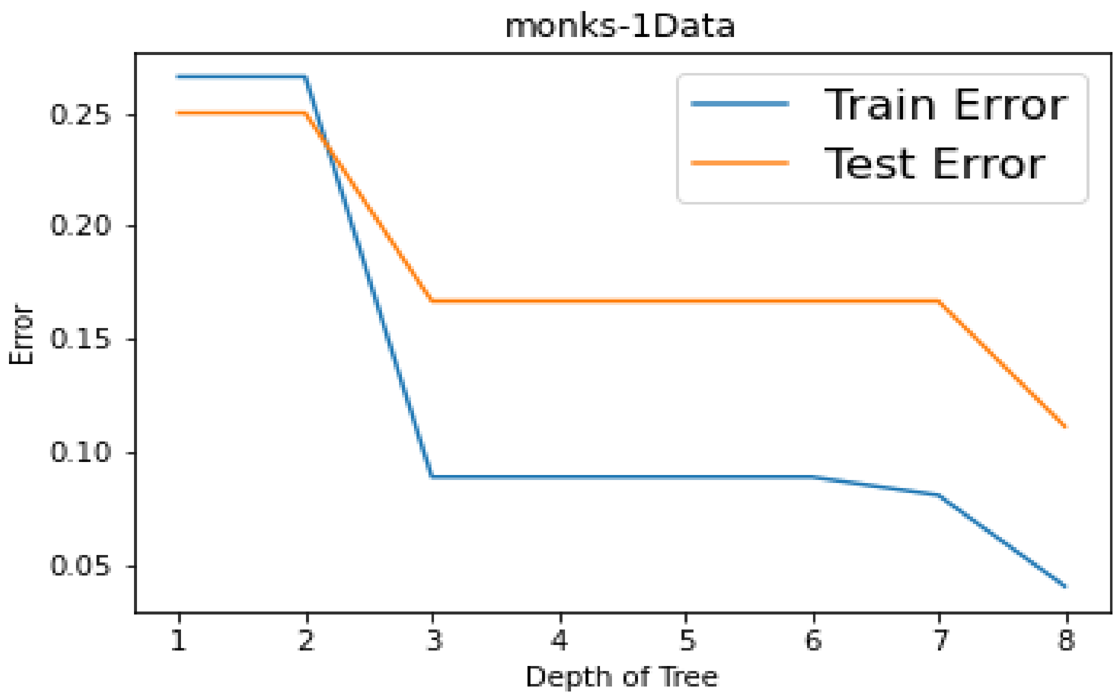
b. (Learning Curves, 20 points) For depth = 1, …, 10, learn decision trees and compute the average training and test errors on each of the three MONK's problems. Make three plots, one for each of the MONK's problem sets, plotting training and testing error curves together for each problem, with tree depth on the x-axis and error on the y-axis.

Answer:

Monk-1 dataset:

Training error Average=12.60%

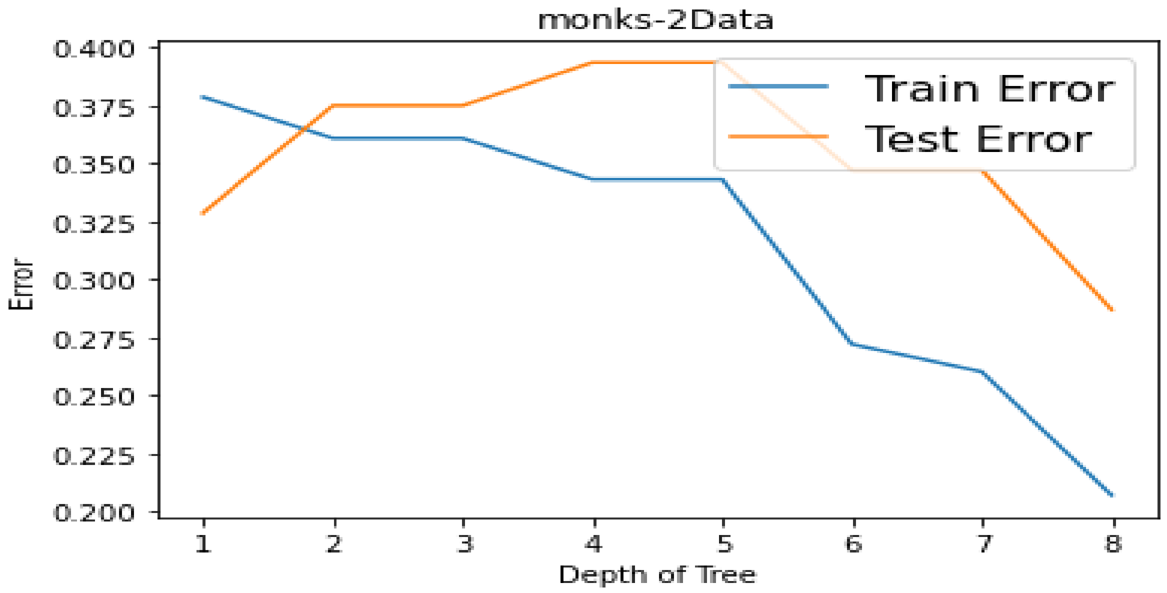
Testing Error Average= 18.06%



Monk-2 dataset:

Training error Average=31.58%

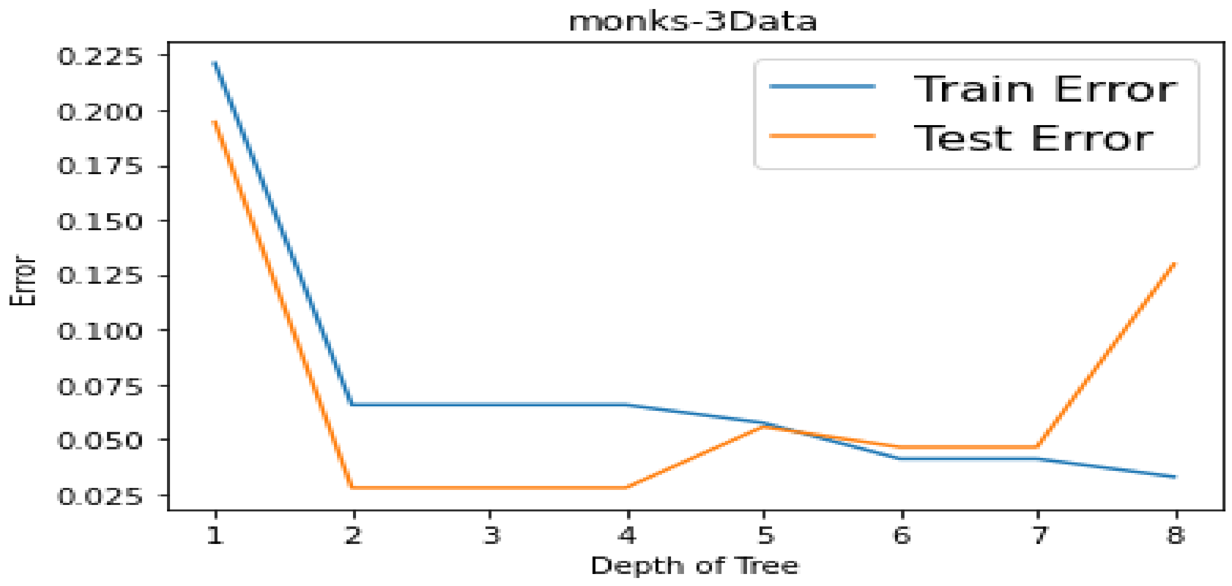
Testing Error Average= 35.59%



Monk-3 dataset:

Training error Average=7.38%

Testing Error Average= 6.94%



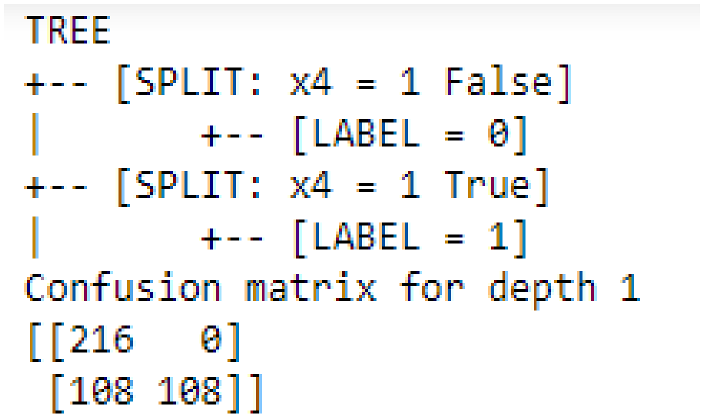
Observation:

As we can see from the above plots for the 3 datasets and from observing the Testing Error and Training Error, we can say that Monk-2 has the highest Error and after that Monk-1 has the Second Highest Error and Monk-3 has the Lowest Error.

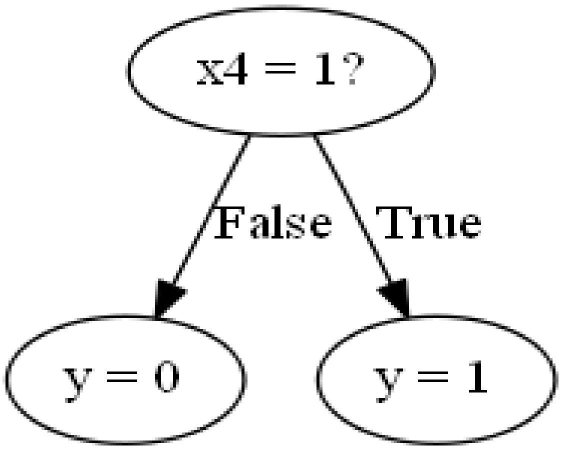
c. (Weak Learners, 20 points) For monks-1, report the visualized learned decision tree and the confusion matrix on the test set for depth = 1, 3, 5. You may use scikit-learns's confusion matrix() function [2].

Answer:

For Depth=1



```
TREE
+-- [SPLIT: x4 = 1 False]
|       +-- [LABEL = 0]
+-- [SPLIT: x4 = 1 True]
|       +-- [LABEL = 1]
Confusion matrix for depth 1
[[216   0]
 [108 108]]
```

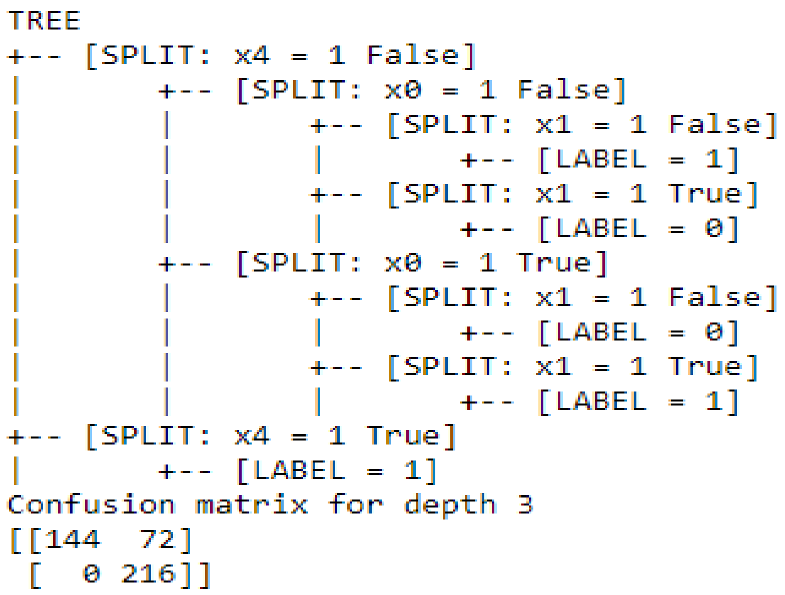


Confusion Matrix:

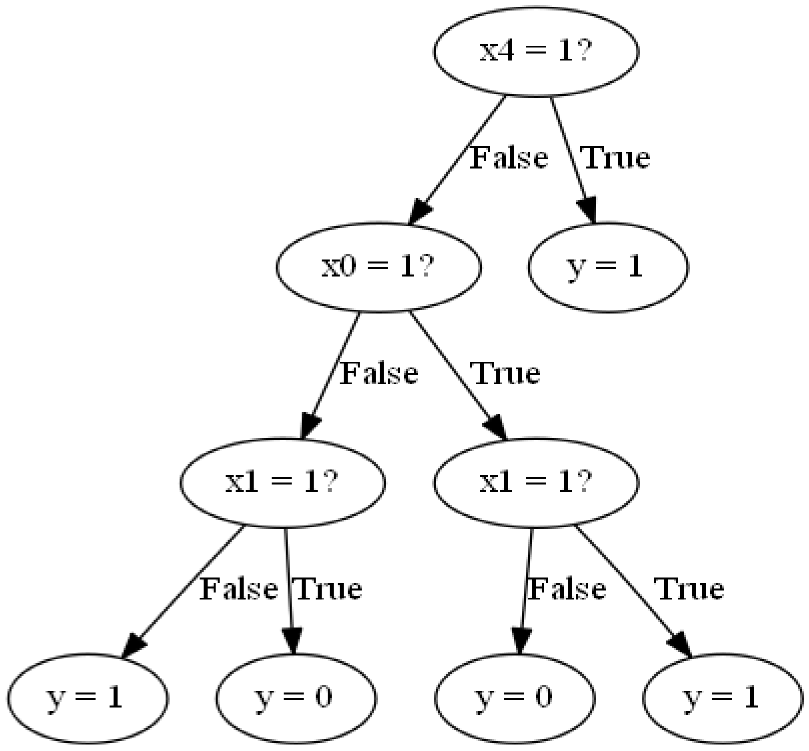
[[216   0]

 [108 108]]

For Depth=3



```
TREE
+-- [SPLIT: x4 = 1 False]
|       +-- [SPLIT: x0 = 1 False]
|       |       +-- [SPLIT: x1 = 1 False]
|       |       |       +-- [LABEL = 1]
|       |       +-- [SPLIT: x1 = 1 True]
|       |       |       +-- [LABEL = 0]
|       +-- [SPLIT: x0 = 1 True]
|       |       +-- [SPLIT: x1 = 1 False]
|       |       |       +-- [LABEL = 0]
|       |       +-- [SPLIT: x1 = 1 True]
|       |       |       +-- [LABEL = 1]
+-- [SPLIT: x4 = 1 True]
|       +-- [LABEL = 1]
Confusion matrix for depth 3
[[144  72]
 [  0 216]]
```



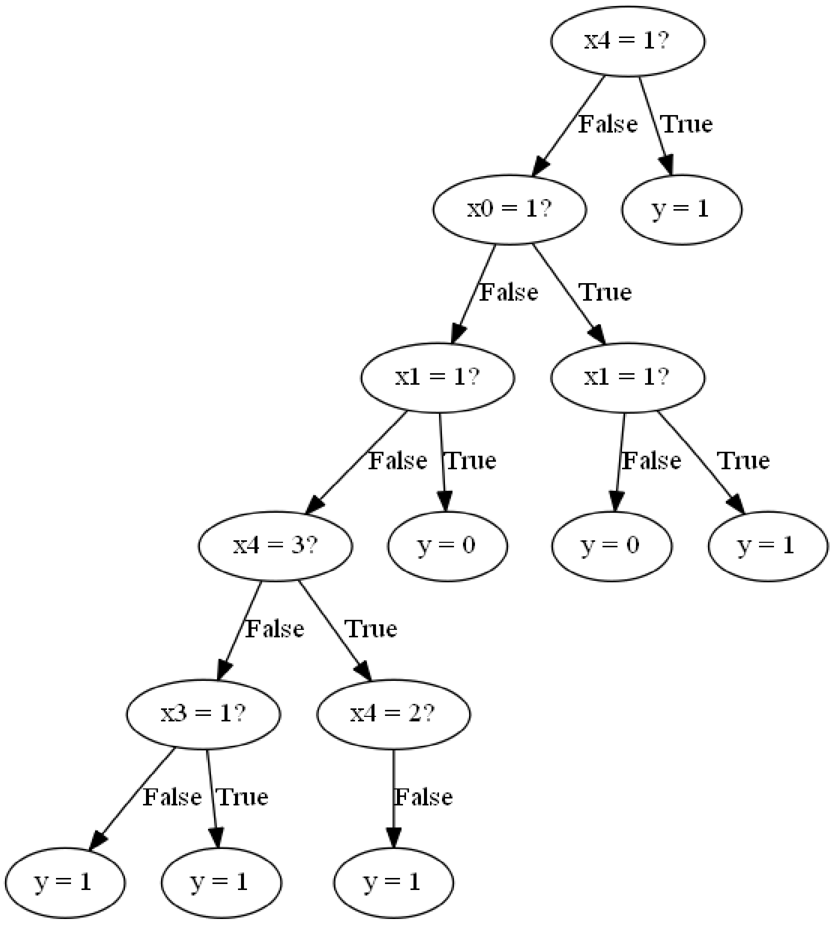
Confusion Matrix:

[[144 72]

 [0 216]]

For Depth=5

```
+-- [SPLIT: x4 = 1 False]
|       +-- [SPLIT: x0 = 1 False]
|       |       +-- [SPLIT: x1 = 1 False]
|       |       |       +-- [SPLIT: x4 = 3 False]
|       |       |       |       +-- [SPLIT: x3 = 1 False]
|       |       |       |       |       +-- [LABEL = 1]
|       |       |       |       +-- [SPLIT: x3 = 1 True]
|       |       |       |       |       +-- [LABEL = 1]
|       |       |       +-- [SPLIT: x4 = 3 True]
|       |       |       |       +-- [SPLIT: x4 = 2 False]
|       |       |       |       |       +-- [LABEL = 1]
|       |       +-- [SPLIT: x1 = 1 True]
|       |       |       +-- [LABEL = 0]
|       +-- [SPLIT: x0 = 1 True]
|       |       +-- [SPLIT: x1 = 1 False]
|       |       |       +-- [LABEL = 0]
|       |       +-- [SPLIT: x1 = 1 True]
|       |       |       +-- [LABEL = 1]
+-- [SPLIT: x4 = 1 True]
|       +-- [LABEL = 1]
Confusion matrix for depth 5
[[144  72]
 [  0 216]]
```



Confusion Matrix:

```
[[144  72]
 [  0 216]]
```

Observation:

Here we have created decision Trees for Specific Depth 1, 3, and 5 for Dataset Monk-1.

And for these depths we also have created a Confusion Matrix using

scikit-learns's confusion matrix().

The Format of Confusion Matrix is as Follows:

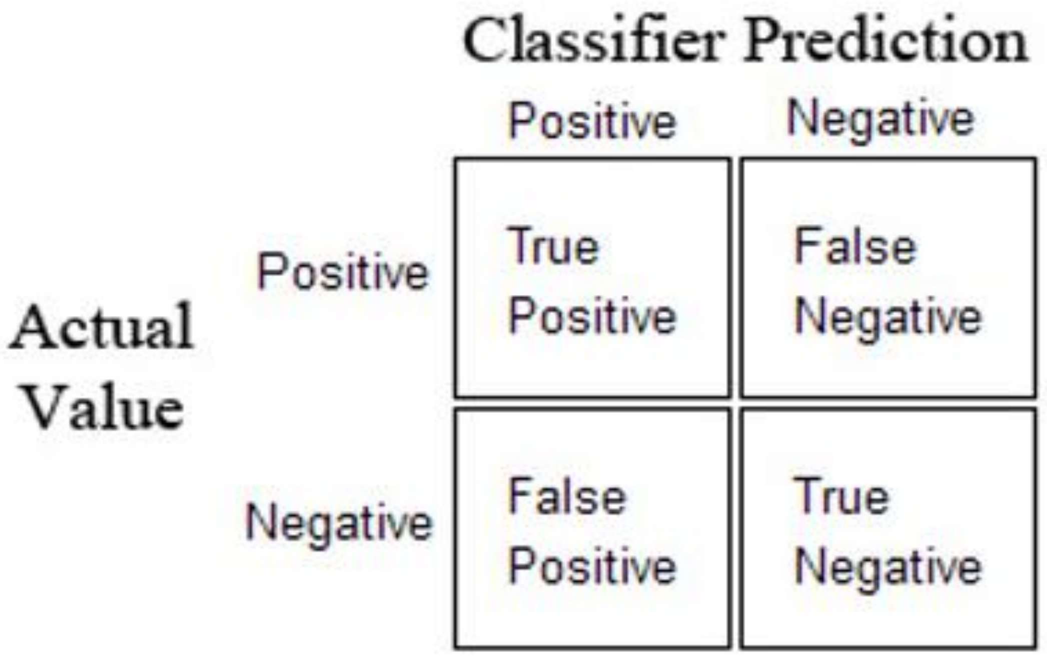
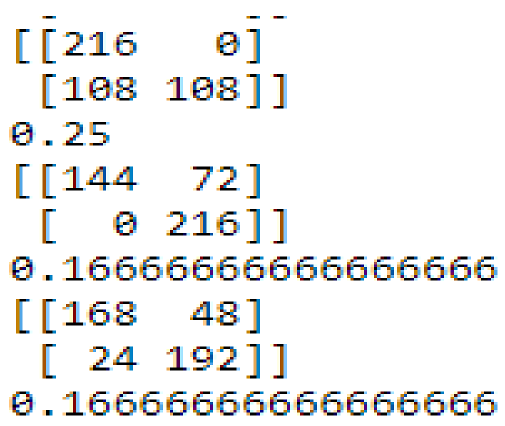


Figure 1: Confusion matrix for binary Classification Problem

d. (scikit-learn, 20 points) For monks-1, use scikit-learn's DecisionTreeClassifier [3] to learn a decision tree using criterion='entropy' for depth = 1, 3, 5. You may use scikit-learn's confusion matrix() function [2].

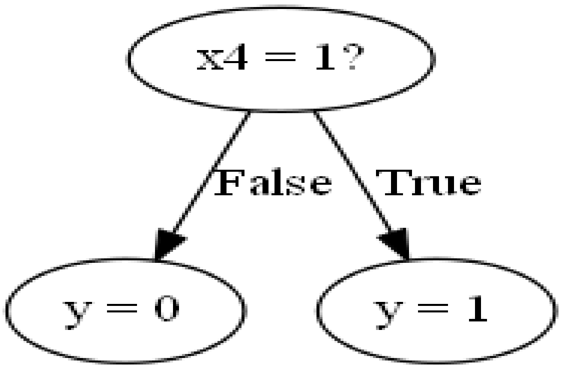
Answer:

For Depth=1,3,5



```
[[216   0]
 [108 108]]
0.25
[[144  72]
 [  0 216]]
0.16666666666666666
[[168  48]
 [ 24 192]]
0.16666666666666666
```

For Depth=1



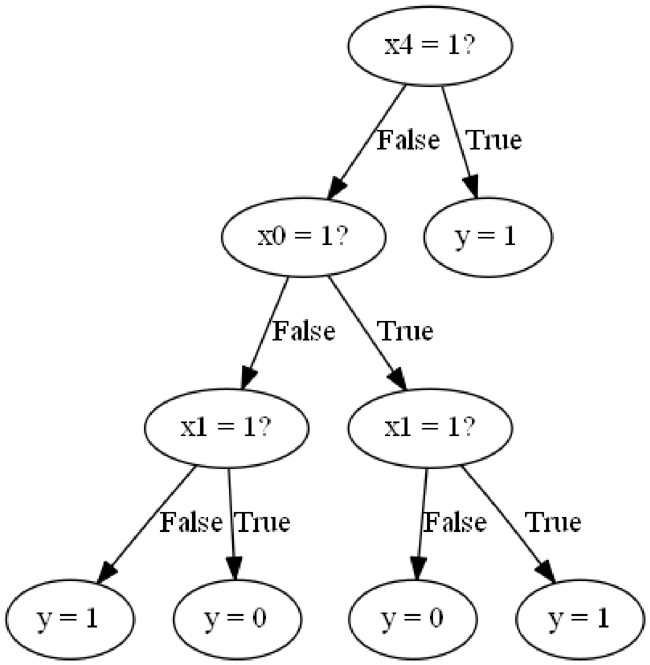
Confusion Matrix:

[[216   0]

 [108 108]]

Testing Error=25%

For Depth=3



Confusion Matrix:

```
[[144  72]
 [  0 216]]
```

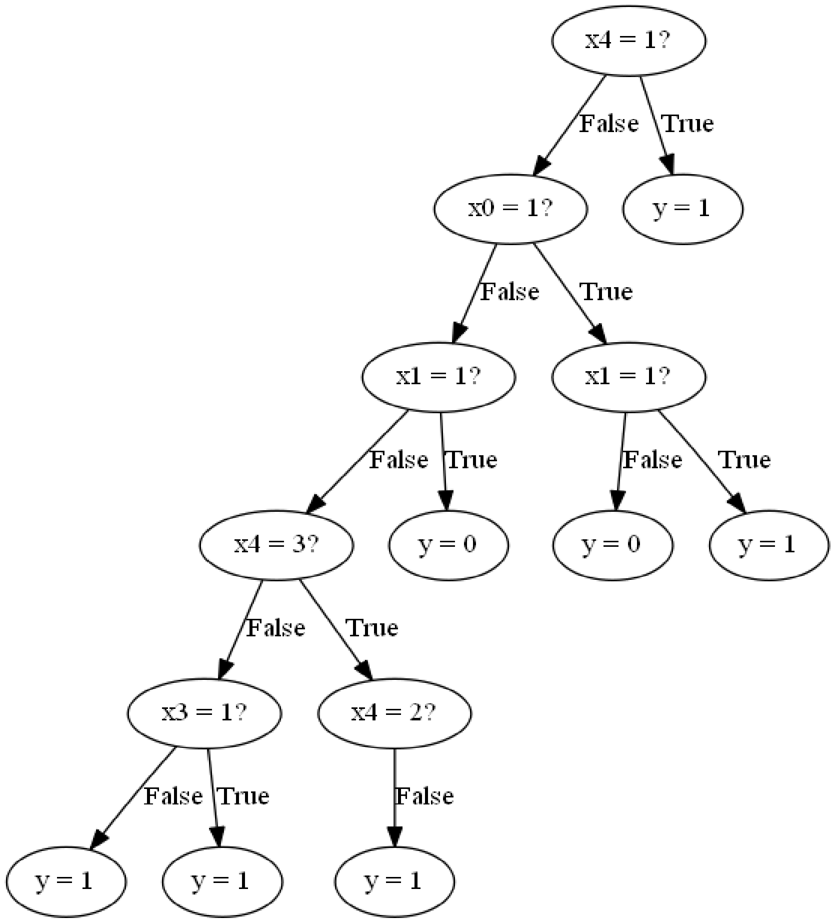
Testing Error= 16.66%

For Depth=5

Confusion Matrix:

```
[[168  48]
 [ 24 192]]
```

Testing Error= 16.66%



Observation:

Here we have created decision Trees for Specific Depth 1, 3, and 5 for Dataset Monk-1. We have created these decision trees with use of scikit-learn's DecisionTreeClassifier function.

And for these depths we also have created a Confusion Matrix using scikit-learns's confusion matrix().

Also, As we increase Depth the Error drops for the decision tree.

e. (Other Data Sets, 20 points) Repeat steps (c) and (d) with your "own" data set and report the confusion matrices. You can use other data sets in the UCI repository. If you encounter continuous features, consider a simple discretization strategy to pre-process them into binary features using the mean. For example, a continuous feature x can be discretized using its mean μ as

$$x_{\text{binary}} = \begin{cases} 0, & \text{if } x \le \mu, \\ 1, & \text{if } x > \mu. \end{cases}$$

Answer:

Here I have used a dataset from UCI repository. The Dataset is Balance Scale Dataset.

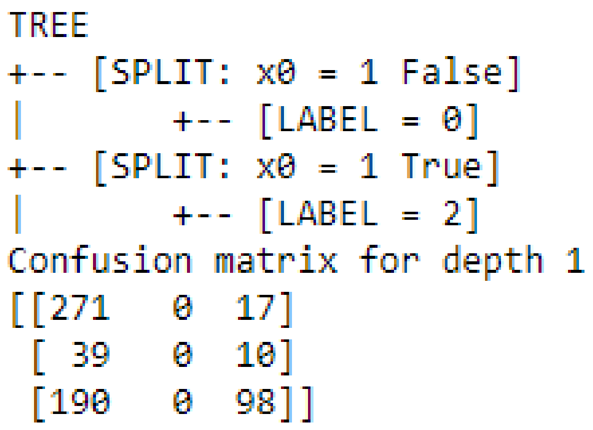
Data Set Information:

This data set was generated to model psychological experimental results. Each example is classified as having the balance scale tip to the right, tip to the left, or be balanced. The attributes are the left weight, the left distance, the right weight, and the right distance. The correct way to find the class is the greater of (left-distance * left-weight) and (right-distance * right-weight). If they are equal, it is balanced.

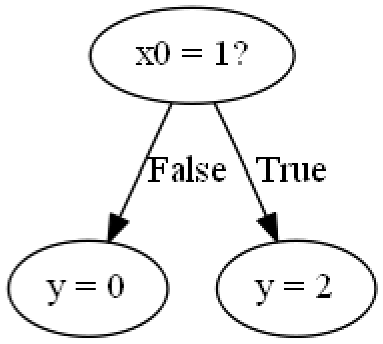
Attribute Information:

1. Class Name: 3 (0, 1, 2)
2. Left-Weight: 5 (1, 2, 3, 4, 5)
3. Left-Distance: 5 (1, 2, 3, 4, 5)
4. Right-Weight: 5 (1, 2, 3, 4, 5)
5. Right-Distance: 5 (1, 2, 3, 4, 5)

For Depth=1



```
TREE
+-- [SPLIT: x0 = 1 False]
|       +-- [LABEL = 0]
+-- [SPLIT: x0 = 1 True]
|       +-- [LABEL = 2]
Confusion matrix for depth 1
[[271   0  17]
 [ 39   0  10]
 [190   0  98]]
```



Confusion Matrix:

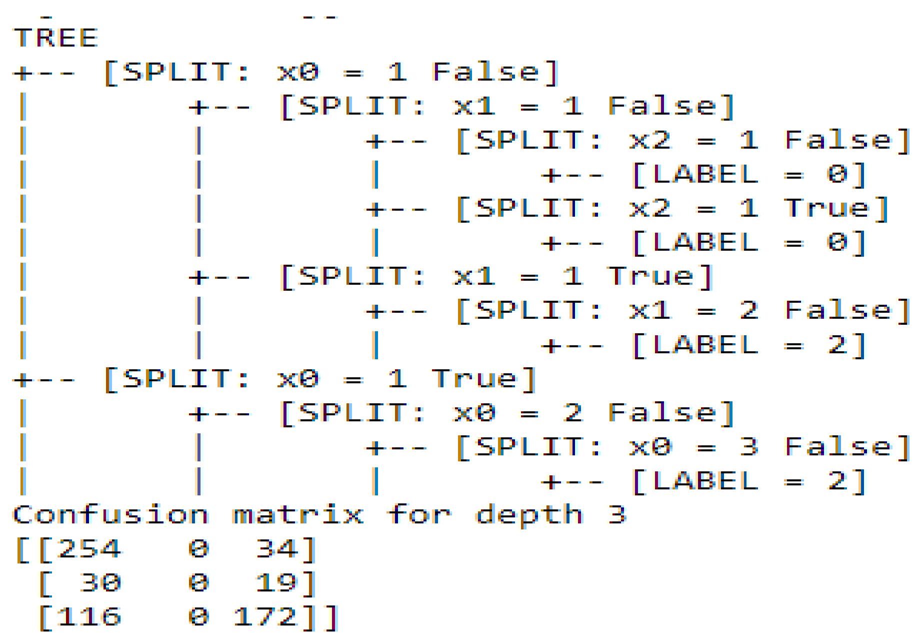
```
[[271   0  17]
 [ 39   0  10]
 [190   0  98]]
```

Confusion Matrix Using Decision Tree Classifier:

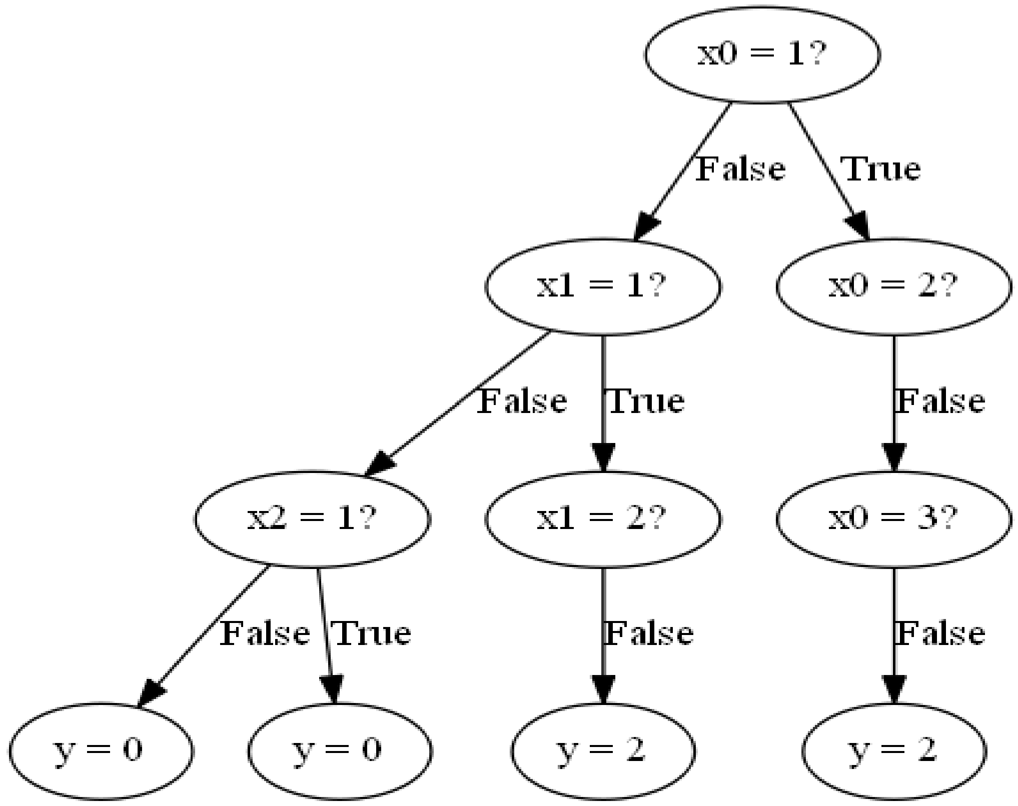
```
[[228   0  60]
 [ 28   0  21]
 [119   0 169]]
```

Testing Error = 36.48%

For Depth=3



```
TREE
+-- [SPLIT: x0 = 1 False]
|       +-- [SPLIT: x1 = 1 False]
|       |       +-- [SPLIT: x2 = 1 False]
|       |       |       +-- [LABEL = 0]
|       |       +-- [SPLIT: x2 = 1 True]
|       |       |       +-- [LABEL = 0]
|       +-- [SPLIT: x1 = 1 True]
|       |       +-- [SPLIT: x1 = 2 False]
|       |       |       +-- [LABEL = 2]
+-- [SPLIT: x0 = 1 True]
|       +-- [SPLIT: x0 = 2 False]
|       |       +-- [SPLIT: x0 = 3 False]
|       |       |       +-- [LABEL = 2]
Confusion matrix for depth 3
[[254   0  34]
 [ 30   0  19]
 [116   0 172]]
```



Confusion Matrix:

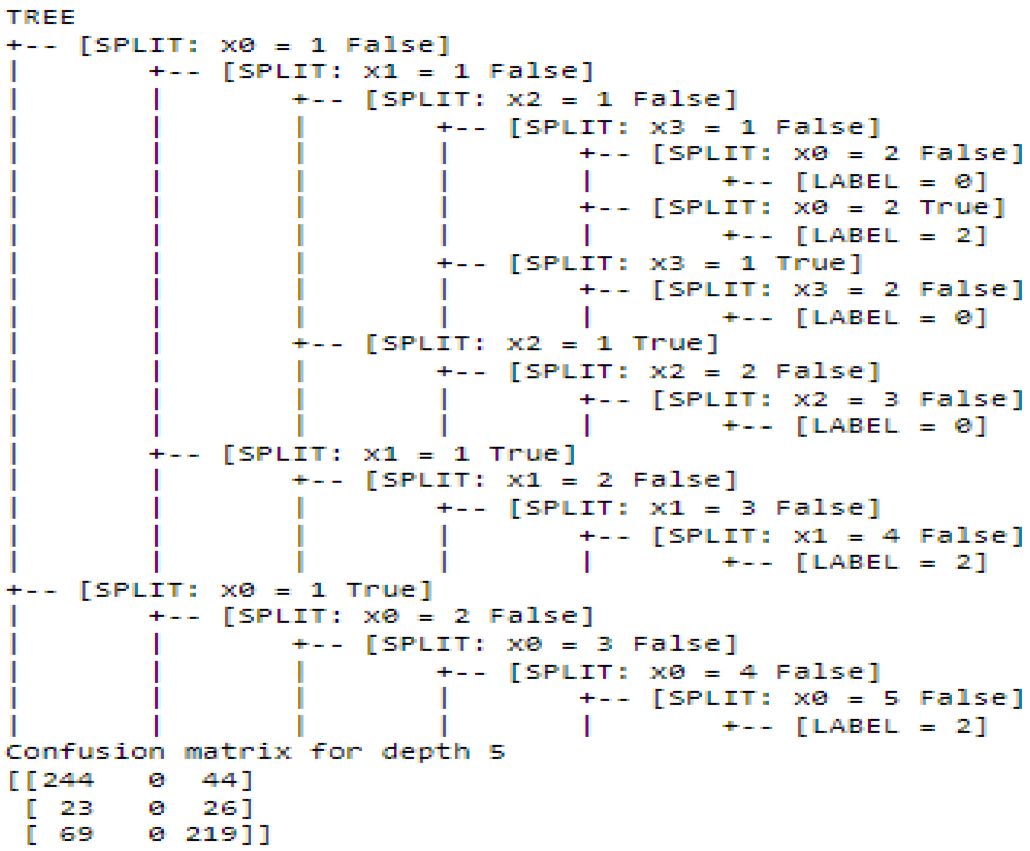
```
[[254   0  34]
 [ 30   0  19]
 [116   0 172]]
```

Confusion Matrix Using Decision Tree Classifier:

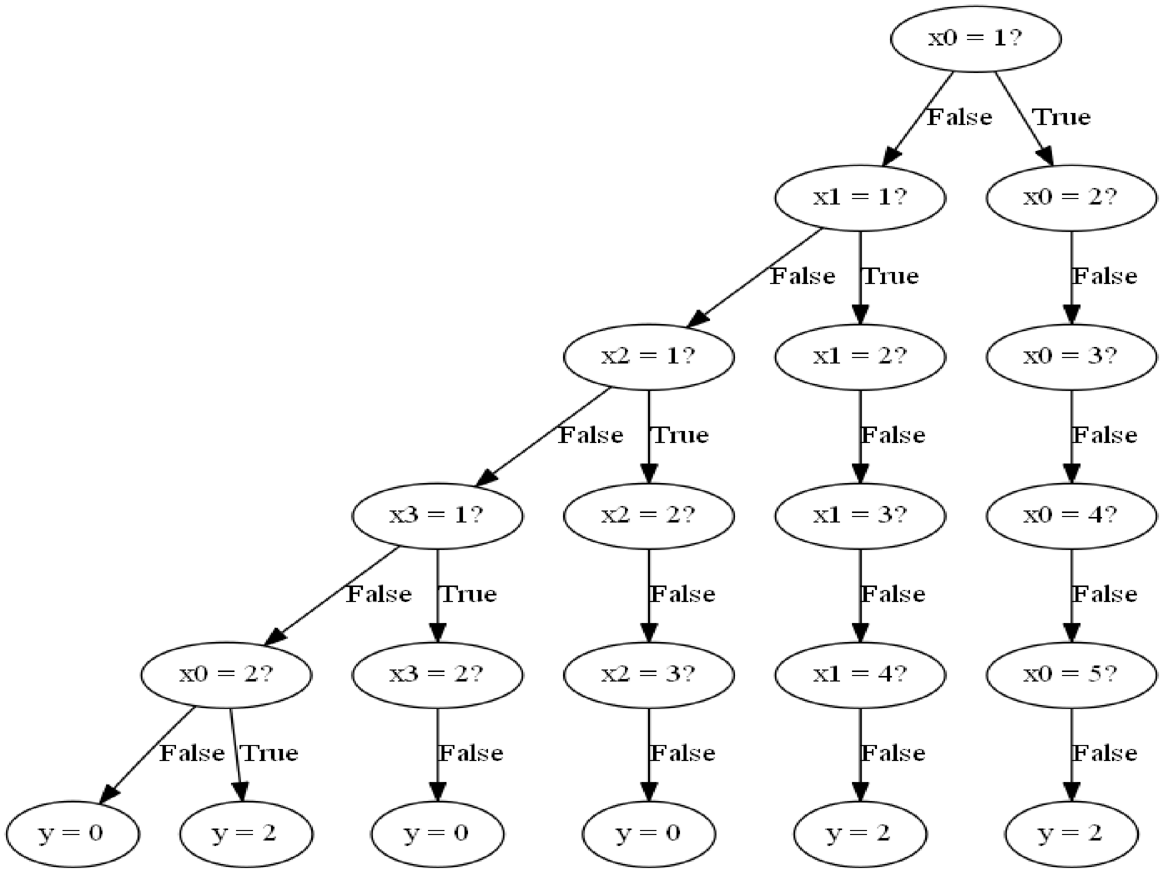
```
[[239   0  49]
 [ 29   0  20]
 [ 57   0 231]]
```

Testing Error = 24.8%

For Depth=5



```
TREE
+-- [SPLIT: x0 = 1 False]
|       +-- [SPLIT: x1 = 1 False]
|       |       +-- [SPLIT: x2 = 1 False]
|       |       |       +-- [SPLIT: x3 = 1 False]
|       |       |       |       +-- [SPLIT: x0 = 2 False]
|       |       |       |       |       +-- [LABEL = 0]
|       |       |       |       +-- [SPLIT: x0 = 2 True]
|       |       |       |       |       +-- [LABEL = 2]
|       |       |       +-- [SPLIT: x3 = 1 True]
|       |       |       |       +-- [SPLIT: x3 = 2 False]
|       |       |       |       |       +-- [LABEL = 0]
|       |       +-- [SPLIT: x2 = 1 True]
|       |       |       +-- [SPLIT: x2 = 2 False]
|       |       |       |       +-- [SPLIT: x2 = 3 False]
|       |       |       |       |       +-- [LABEL = 0]
|       +-- [SPLIT: x1 = 1 True]
|       |       +-- [SPLIT: x1 = 2 False]
|       |       |       +-- [SPLIT: x1 = 3 False]
|       |       |       |       +-- [SPLIT: x1 = 4 False]
|       |       |       |       |       +-- [LABEL = 2]
+-- [SPLIT: x0 = 1 True]
|       +-- [SPLIT: x0 = 2 False]
|       |       +-- [SPLIT: x0 = 3 False]
|       |       |       +-- [SPLIT: x0 = 4 False]
|       |       |       |       +-- [SPLIT: x0 = 5 False]
|       |       |       |       |       +-- [LABEL = 2]
Confusion matrix for depth 5
[[244   0  44]
 [ 23   0  26]
 [ 69   0 219]]
```



Confusion Matrix:

```
[[244   0  44]
 [ 23   0  26]
 [ 69   0 219]]
```

Confusion Matrix Using Decision Tree Classifier:

```
[[267   0  21]
 [ 27   0  22]
 [ 32   0 256]]
```

Testing Error = 16.32%

Observation:

Here we have created decision trees for Balance Scale dataset for depth=1,3,5.

And for these depths we also have created a Confusion Matrix using scikit-learns's confusion matrix().

Also, we have created decision Trees for Specific Depth 1, 3, and 5 for Balance Scale dataset. We have created these decision trees with use of scikit-learn's DecisionTreeClassifier function.

And for these depths we also have created a Confusion Matrix using scikit-learns's confusion matrix() with DecisionTreeClassifier function.

Also, As we increase Depth the Error drops for the decision tree.