# Programming Assignment #1

Submitted by:
Kapil Gautam - KXG180032
Vishwashri Sairam Venkadathiriappasamy - VXV180043

**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.

**Data Sets**: The data sets are obtained from the UCI Repository and are collectively the MONK's Problem. 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 in the raw files), and the class labels (column 1). There are 2 classes.

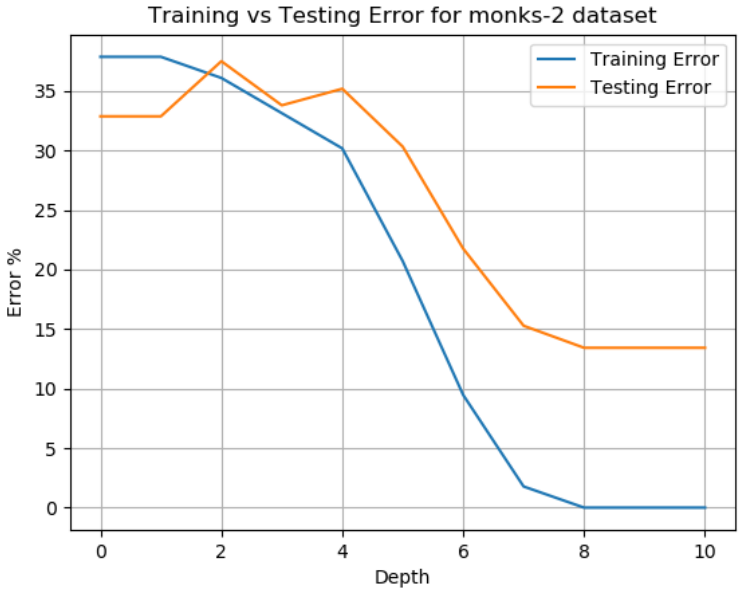
a. Learning Curves: For depth = 1.....10, learn decision trees and compute the average training and test errors on each of the three MONK's problems. Make three plots, one for each of the MONK's problem sets, plotting training and testing error curves together for each problem, with tree depth on the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

Computing testing and training error for
**monks-1 dataset**

| Depth | Train Error | Test Error |
|---|---|---|
| 1 | 26.61% | 25.00%. |
| 2 | 26.61% | 25.00%. |
| 3 | 8.87% | 16.67%. |
| 4 | 8.87% | 16.67%. |
| 5 | 8.06% | 16.67%. |
| 6 | 4.84% | 11.11%. |
| 7 | 3.23% | 9.26%. |
| 8 | 2.42% | 9.26%. |
| 9 | 1.61% | 8.33%. |
| 10 | 0.00% | 7.41%. |

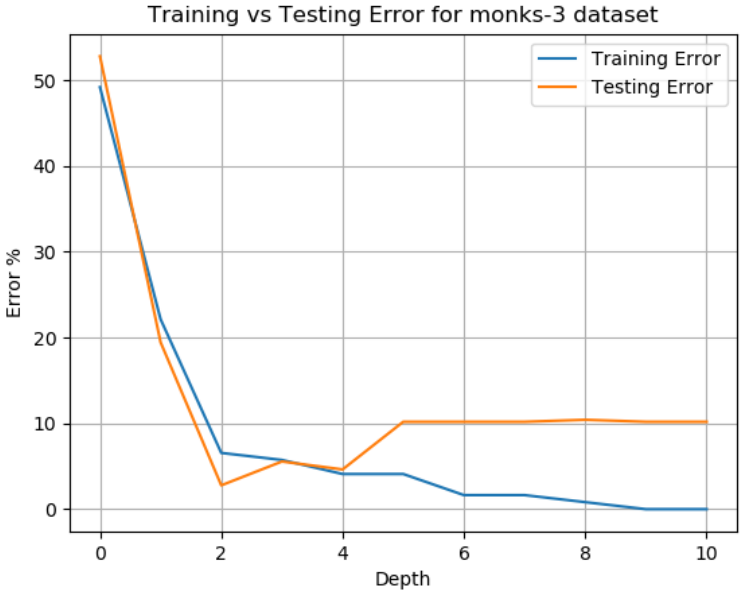
Computing testing and training error for **monks-2 dataset**

| Depth | Train Error | Test Error |
|---|---|---|
| 1 | 37.87% | 32.87%. |
| 2 | 36.09% | 37.50%. |
| 3 | 33.14% | 33.80%. |
| 4 | 30.18% | 35.19%. |
| 5 | 20.71% | 30.32%. |
| 6 | 9.47% | 21.76%. |
| 7 | 1.78% | 15.28%. |
| 8 | 0.00% | 13.43%. |
| 9 | 0.00% | 13.43%. |
| 10 | 0.00% | 13.43%. |




Computing testing and training error for **monks-3 dataset**

| Depth | Train Error | Test Error |
|---|---|---|
| 1 | 22.13% | 19.44%. |
| 2 | 6.56% | 2.78%. |
| 3 | 5.74% | 5.56%. |
| 4 | 4.10% | 4.63%. |
| 5 | 4.10% | 10.19%. |
| 6 | 1.64% | 10.19%. |
| 7 | 1.64% | 10.19%. |
| 8 | 0.82% | 10.42%. |
| 9 | 0.00% | 10.19%. |
| 10 | 0.00% | 10.19%. |



b. Weak Learners: For monks-1, report the learned decision tree and the confusion matrix on the test set for depth = 1 and depth = 2. A confusion matrix is a table that is used to describe the performance of a classifier on a data set.

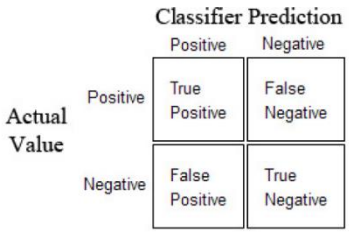



Figure 1: Confusion matrix for a binary classification problem.

**Depth = 1**

```
TREE
+-- [SPLIT: x4 = 1]
|     +-- [LABEL = 1]
+-- [SPLIT: x4 = 1]
|     +-- [LABEL = 0]
```

Train Error = 26.61%.
Test Error = 25.00%.

Confusion Matrix:

| | | Classifier Prediction | |
|---|---|---|---|
| | | Positive | Negative |
| Actual Value | Positive | 108 | 108 |
| | Negative | 0 | 216 |

**Depth = 2**

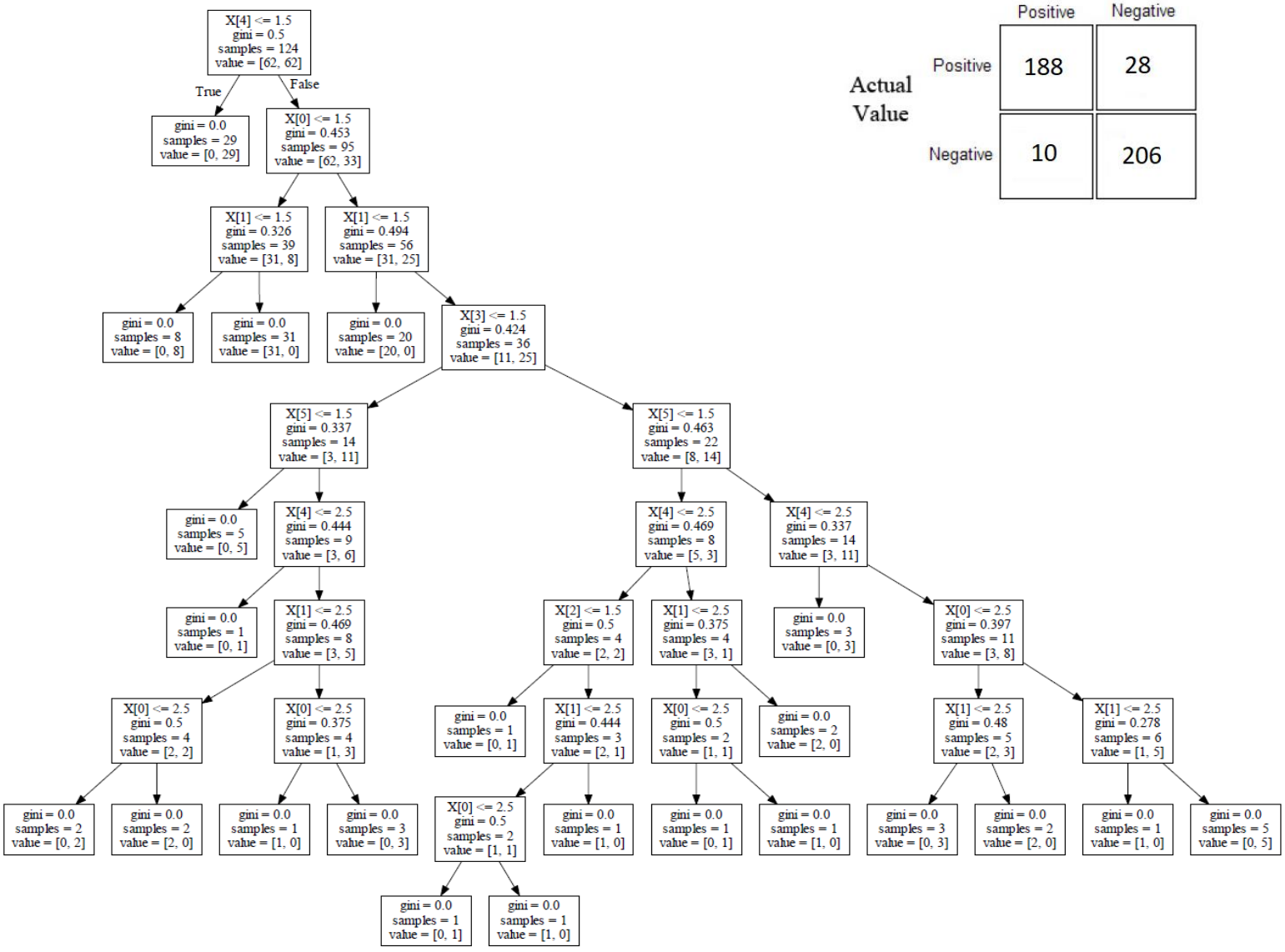
```
TREE
+-- [SPLIT: x4 = 1]
|     +-- [LABEL = 1]
+-- [SPLIT: x4 = 1]
|     +-- [SPLIT: x0 = 1]
|     |     +-- [LABEL = 0]
|     +-- [SPLIT: x0 = 1]
|     |     +-- [LABEL = 0]
```

Train Error = 26.61%.
Test Error = 25.00%.

Confusion Matrix:

| | | Classifier Prediction | |
|---|---|---|---|
| | | Positive | Negative |
| Actual Value | Positive | 108 | 108 |
| | Negative | 0 | 216 |

c. scikit-learn: For monks-1, use scikit-learns's default decision tree algorithm to learn a decision tree. Visualize the learned decision tree using graphviz. Report the visualized decision tree and the confusion matrix on the test set. Do not change the default parameters.
(In binary classification in sklearn, the count of true negatives is C[0][0] , false negatives is C[1][0], true positives is C[1][1] and false positives is C[0][1].)




Classifier Prediction

| Actual Value | Positive | Negative |
|---|---|---|
| Positive | 188 | 28 |
| Negative | 10 | 206 |

d. Other Data Sets : Repeat steps 2 and 3 with your “own" data set and report the confusion matrices.

**1.) Mushroom dataset**

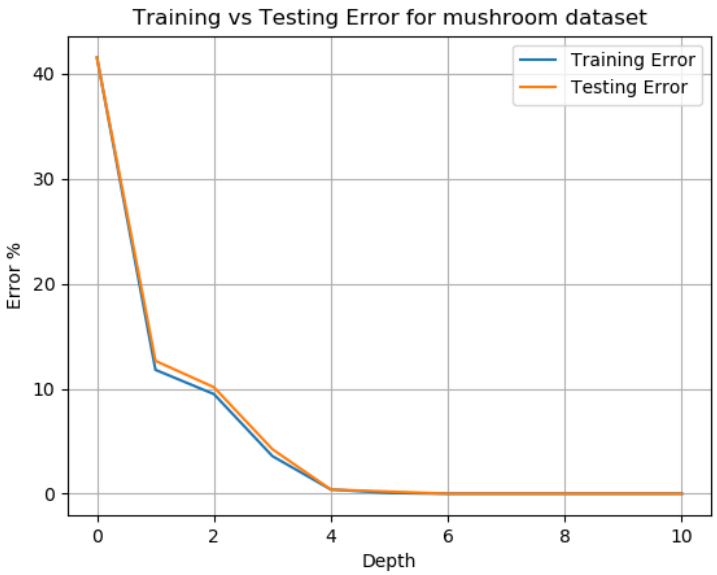
**Depth = 1**

TREE
+-- [SPLIT: x17 = 4]
|  +-- [LABEL = 1]
+-- [SPLIT: x17 = 4]
|  +-- [LABEL = 0]

Train Error = 11.80%.
Test Error = 12.65%.

Confusion Matrix:

| | | Classifier Prediction | |
|---|---|---|---|
| | | Positive | Negative |
| Actual Value | Positive | 793 | 51 |
| | Negative | 206 | 981 |




**Depth: 2**

TREE
+-- [SPLIT: x17 = 4]
|  +-- [SPLIT: x9 = 0]
|  |  +-- [LABEL = 1]
|  +-- [SPLIT: x9 = 0]
|  |  +-- [LABEL = 1]
+-- [SPLIT: x17 = 4]
|  +-- [SPLIT: x16 = 2]
|  |  +-- [LABEL = 1]
|  +-- [SPLIT: x16 = 2]
|  |  +-- [LABEL = 0]

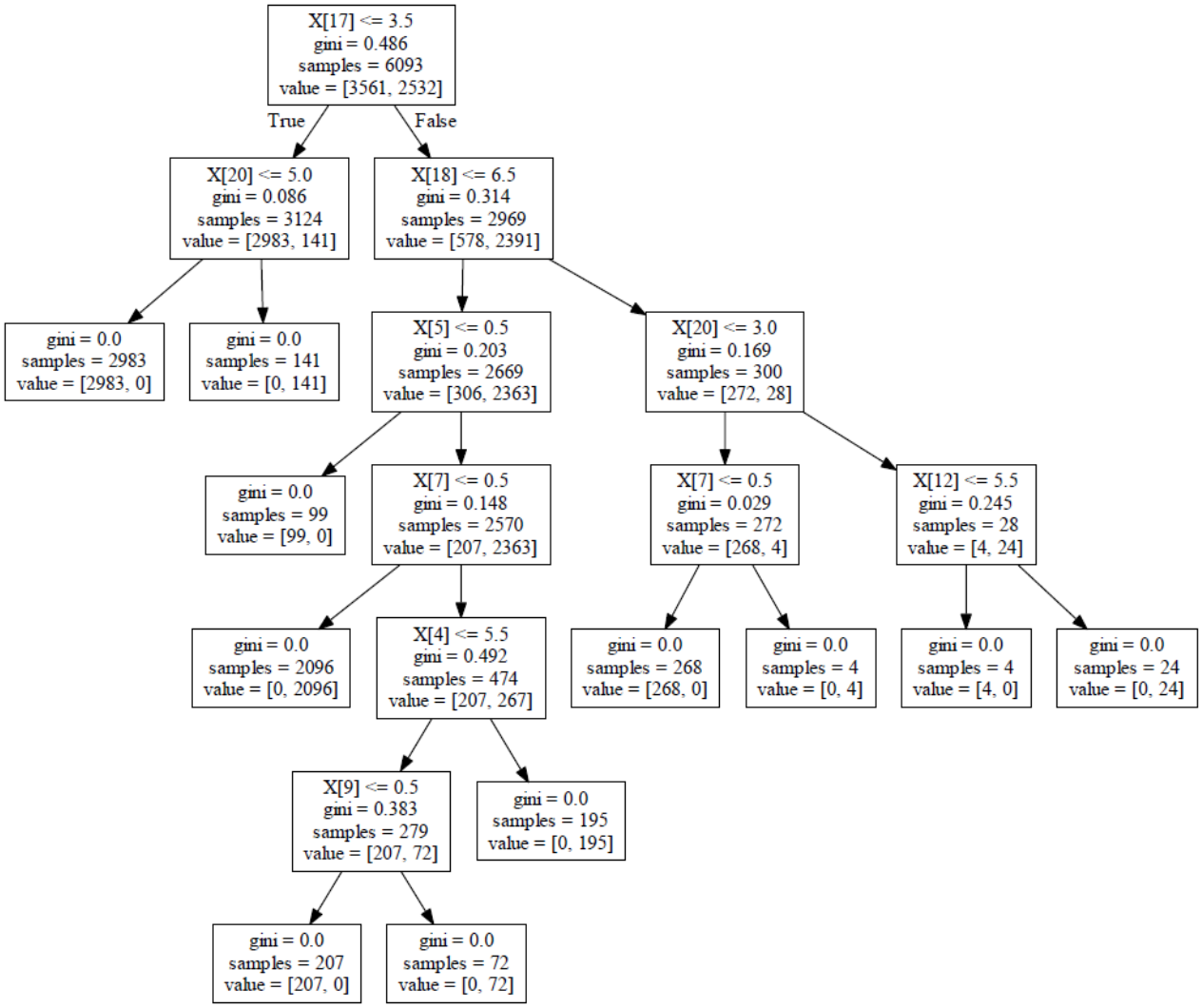
Train Error = 9.49%.
Test Error = 10.14%.

Confusion Matrix:

| | | Classifier Prediction | |
|---|---|---|---|
| | | Positive | Negative |
| Actual Value | Positive | 844 | 0 |
| | Negative | 206 | 981 |

Using sklearn default parameters TreeClassifier on mushroom dataset:

| | | Classifier Prediction: Positive | Classifier Prediction: Negative |
|---|---|---|---|
| Actual Value | Positive | 844 | 0 |
| | Negative | 0 | 1187 |



## 2.) Tic Tac Toe dataset

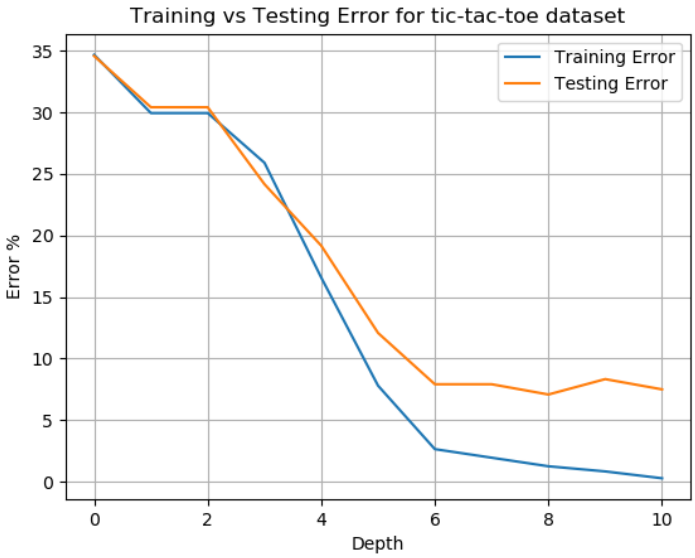
**Depth: 1**

TREE
+-- [SPLIT: x4 = 1]
|  +-- [LABEL = 0]
+-- [SPLIT: x4 = 1]
|  +-- [LABEL = 1]

Train Error = 29.94%.
Test Error = 30.42%.

Confusion Matrix:

| | | Classifier Prediction | |
|---|---|---|---|
| | | Positive | Negative |
| Actual Value | Positive | 120 | 37 |
| | Negative | 36 | 47 |




**Depth: 2**

TREE
+-- [SPLIT: x4 = 1]
|  +-- [SPLIT: x0 = 1]
|  |  +-- [LABEL = 0]
|  +-- [SPLIT: x0 = 1]
|  |  +-- [LABEL = 0]
+-- [SPLIT: x4 = 1]
|  +-- [SPLIT: x6 = 1]
|  |  +-- [LABEL = 1]
|  +-- [SPLIT: x6 = 1]
|  |  +-- [LABEL = 1]

Train Error = 29.94%.
Test Error = 30.42%.

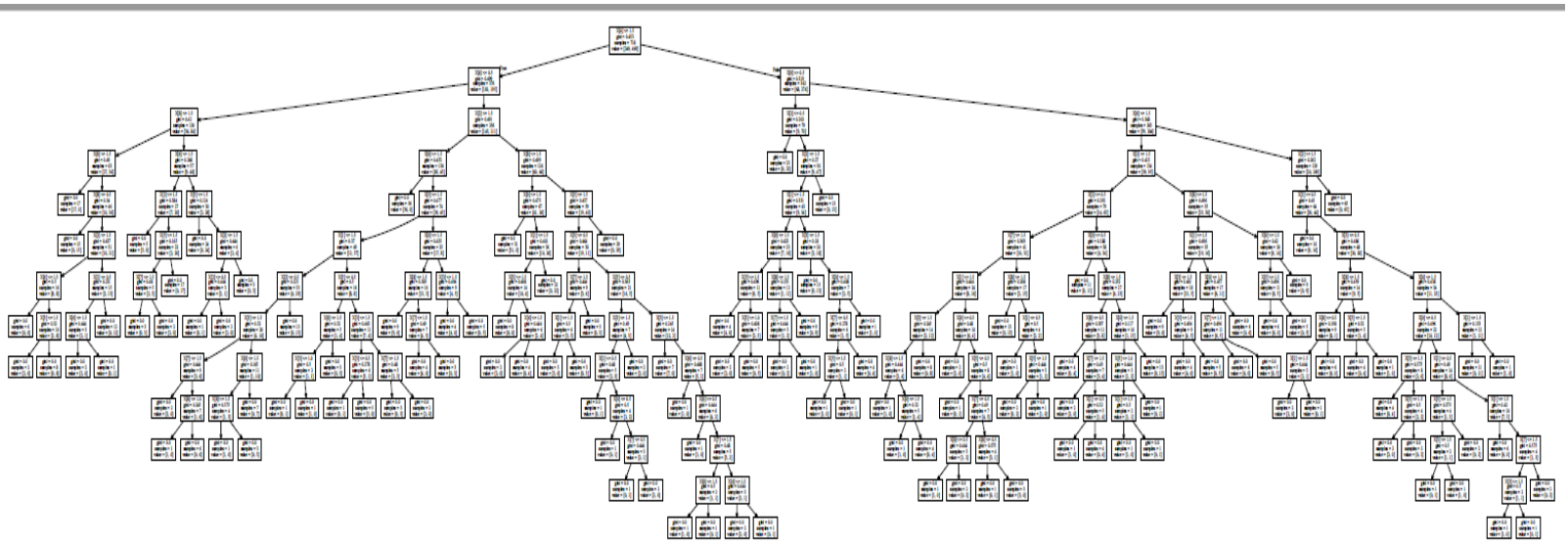
Confusion Matrix:

| | | Classifier Prediction | |
|---|---|---|---|
| | | Positive | Negative |
| Actual Value | Positive | 120 | 37 |
| | Negative | 36 | 47 |

Using sklearn default parameter TreeClassifier on tic-tac-toe dataset:

(This is the full size sklearn tree with default parameter, but it is difficult to comprehend it here.)

| | | Classifier Prediction: Positive | Classifier Prediction: Negative |
|---|---|---|---|
| Actual Value | Positive | 147 | 10 |
| | Negative | 20 | 63 |



Just for visualization purposes, modifying default parameter, max_depth of sklearn.TreeClassifier to 3