Programming Assignment #2*

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 data sets (in the folder ./data/) are obtained from the UCI Repository and are collectively the MONK's Problem. These 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 in the raw files), and the class labels (column 1). There are 2 classes. Refer to the file ./data/monks.names for more details.

a. (**Learning Curves**, 40 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.

b. (**Weak Learners**, 30 points) 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. For binary classification problems, it will be:

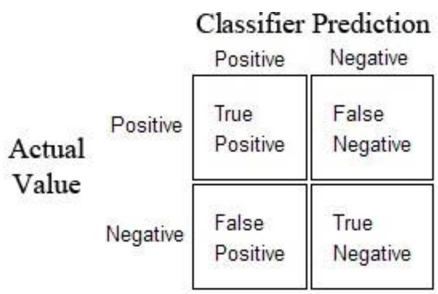

Figure 1: Confusion matrix for a binary classification problem.

c. (scikit-learn, 15 points) For monks-1, use scikit-learns's default decision tree algorithm[2] to learn a decision tree. Visualize the learned decision tree using graphviz[3]. Report the **visualized decision tree** and the **confusion matrix** on the test set. **Do not change the default parameters**.

d. (**Other Data Sets**, 15 points) Repeat steps 2 and 3 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

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

*Adapted from Gautam Kunapuli's programming assignment.
[1] https://archive.ics.uci.edu/ml/datasets/MONK's+Problems
[2] http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
[3] see http://scikit-learn.org/stable/modules/tree.html#classification for an example.