**ASSIGNMENT 2- DECISION TREE (dxc190009)**

1. Autograder - 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.

A close up of a piece of paper

Description automatically generated

A close up of a logo

Description automatically generated

1. 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

Monk 1 Plot

A close up of a map

Description automatically generated

Monk 2 Plot

A close up of a map

Description automatically generated

Monk 3 Plot

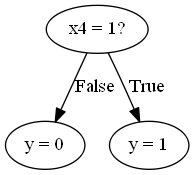
A close up of a map

Description automatically generated

1. 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

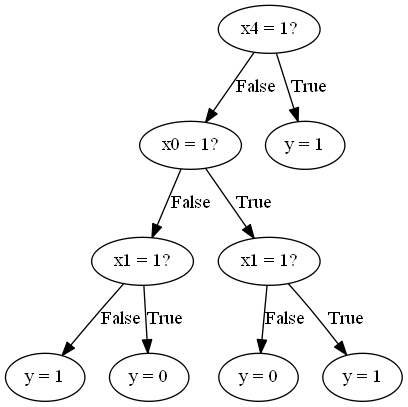
For Monk 1

Decision tree with depth 1:



A screenshot of a cell phone

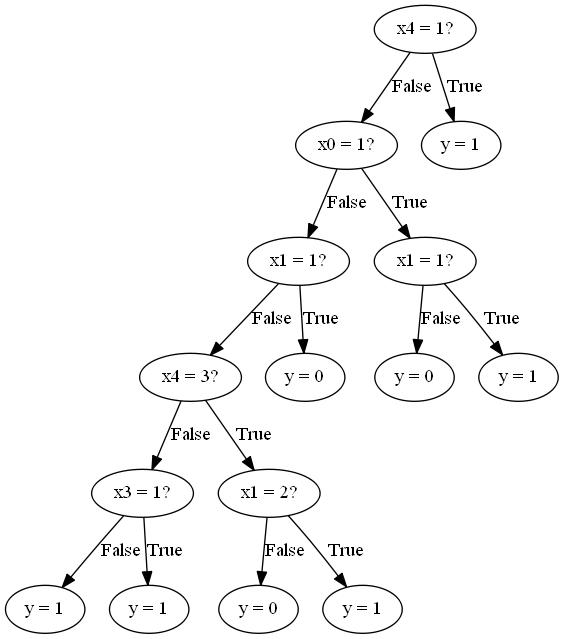
Description automatically generated

Decision tree with depth 3:

A screenshot of a cell phone

Description automatically generated

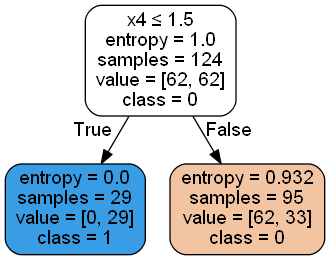
Decision Tree with depth 5:

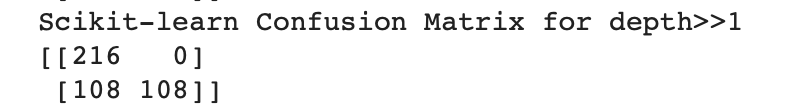


A close up of text on a white background

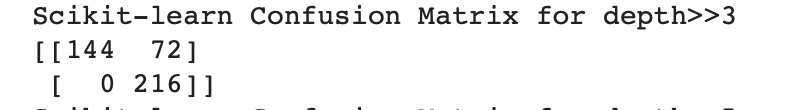
Description automatically generated

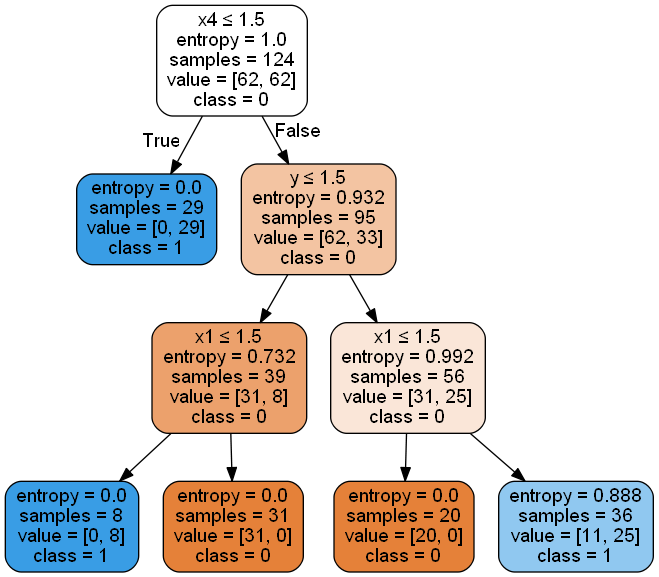
1. For monks-1, use scikit-learn’s DecisionTreeClassifier [3] to learn a decision tree using criterion=’entropy’ for depth = 1, 3, 5. You may use scikit-learn’s confusion matrix()

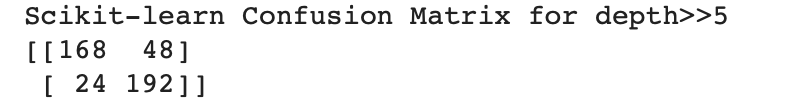
Decision Tree at depth1:

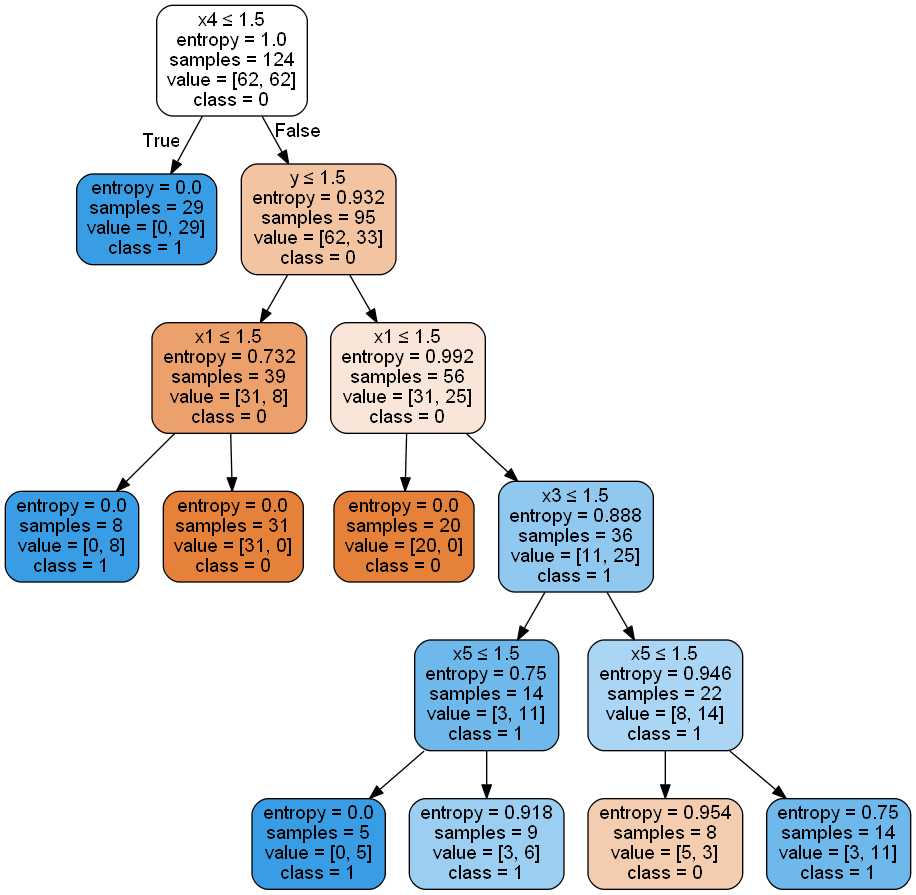


Decision Tree with depth 3:





Decision Tree with Depth 5:



1. 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

A drawing of a person

Description automatically generatedsampledataset (sample.train, sample.test)

Depth 1

A screenshot of a cell phone

Description automatically generated

Depth 3  
A close up of a logo

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Depth 5

A close up of a map

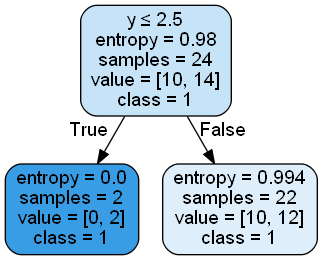
Description automatically generated

A close up of a piece of paper

Description automatically generated

A screenshot of a cell phone

Description automatically generated



A picture containing object

Description automatically generated

A screenshot of a cell phone

Description automatically generated