1. Written assignment:

(a) 2.4

(b) 2.7

(c) 3.4

(d) Consider two attributes Outlook (sunny, rainy, cloudy) and Humidity (high) and outcome PlayTennis (yes, no) for the instance space \( X \).

   i. Consider an unbiased hypothesis space \( H_1 \), enumerate all possible hypotheses \( (h_1, h_2, \ldots) \) in terms of subsets of instances. What is the number of possible unique hypotheses in \( H_1 \)?

   ii. For each hypothesis in \( H_1 \), represent it as a boolean expression. What is the number of unique hypotheses semantically?

   iii. Consider a biased hypothesis space \( H_2 \) where each attribute can only have a value, ?, or \( \emptyset \). What is the number of unique hypotheses semantically in the biased hypothesis space \( H_2 \)?

   iv. Identify hypotheses in the unbiased hypothesis space \( H_1 \) that are not in the biased hypothesis space \( H_2 \).

(e) With the programming assignment: Discuss and compare accuracy of no pruning versus rule post-pruning in testIris and testIrisNoisy.

2. Programming assignment: Decision Tree

(a) Allow more than two outcomes/classes

(b) Allow continuous-valued attributes

(c) Allow printing the tree

(d) Allow the option of rule post-pruning and printing the rules

(e) Two data sets: Tennis and Iris on the course web site.

(f) The same program should be able to handle the two data sets.

(g) For each of the following experiments, provide a script/program/function to run the experiment:

   i. testTennis: print the tree, tree accuracy on the training and test sets, the rules, rule accuracy on the training and test sets (no pruning, the dataset is too small)

   ii. testIris: print the tree, tree accuracy on the training and test sets, the rules after post-pruning, rule accuracy on the training and test sets

(h) Implementation:

   i. Use C (GNU gcc), C++ (GNU g++), Java (Oracle Java), LISP (CLISP), or Python. If you don’t have a preference, use Java since it’s more portable.

   ii. Your program preferably runs on code01.fit.edu (linux).

   iii. You might have these modules:

      A. Learner: input training examples/instances, output a tree (or rule set)

      B. Classifier/predictor: input a tree (or rule set) and labeled instances, output the classifications/predictions and how accurate the tree is with respect to the correct labels (% of correct classifications).

      C. Tree printer (pre-order traversal, deeper nodes are indented more, leaves have class distribution), for example:

         ```
         height = tall
         | size>2 = T
         | | color = black
         | | | weight = heavy : Yes (1,0)
         | | | weight = light : No (0,1)
         | | color = white
         | | weight = heavy : Yes (2,0)
         | | weight = light : No (0,1)
         | size>2 = F
         | | weight = heavy : Yes (4,0)
         | | weight = light : No (0,2)
         height = short : No (0,8)
         ```

      D. Rule set printer, for example:

         ```
         height = tall \& size>2 = T \Rightarrow Yes (1,0)
         height = tall \& size>2 = F \Rightarrow No (0,1)
         ```

   iv. Submission:

      A. README.txt: what are the files and how to compile and run your program on code01.fit.edu

      B. source code