CSE 4301/5290 Homework 5
Due: Dec 2, Wed, 5pm; Submit Server: class = ai , assignment = hw5

For programming problems (Lisp/Java/C/C++/Python):

- Submit:
  - all files that are needed to compile and run
  - README.txt with compilation and run instructions
- Your program should compile and run on `code.fit.edu` (Linux, remote access via ssh).

1. Q18.5, p764, 3Ed (Q18.10, p677, 2Ed)

2. Programming: Implement the decision-tree learning algorithm and evaluate the accuracy of the algorithm on the provided training and test sets. All data sets are available on the course web site.

   The Restaurant data set in Fig. 18.3 (3/2Ed) is the training set. No test set for this data set. Your implementation should reproduce the tree in Fig. 18.6 (3/2Ed).

   The functions (stated in LISP) include:

```
; given examples, attributes, and default class
; return a decision tree
(defun learn-decision-tree (examples attributes default-class)  ...)

; given a tree, print the tree using pre-order traversal
; with more indentation for nodes at deeper levels
(defun print-decision-tree (tree)  ...)

; given a tree and a data set, return the accuracy (%) of the tree on
; the dataset
(defun eval-decision-tree (tree dataset ...)  ...)

; given the file names for attributes, training set, and test set
; read in the attributes, training set, and test set
; learn a decision tree from the training data
; print the decision tree
; print the accuracy of decision tree on the training set
; print the accuracy of decision tree on the test set [could be nil]
(defun test-decision-tree (attr-fname train-fname test-fname) ...)

; test the restaurant data set
(defun test-restaurant ()
  (test-decision-tree 'restaurant-attr.txt 'restaurant-train.txt nil)
)
```

   TestRestaurant module for Java/C/C++/Python.

3. Programming: Choose one of the two problems:

   (a) The IDS data set contains records of network activities that are normal or part of a denial of service (DOS) attack(s) called Neptune (aka SYN-flood). Neptune tries to make many "half" connections to a server. Due to limited resources, a server usually has a maximum number of connections that it can handle. Many malicious "half" connections can prevent legitimate connections to be made. That is, the server might be filled with useless "half" connections, and cannot accept legitimate connections and provide the intended service (hence "denial of service"). (This data set is adapted from `http://kdd.ics.uci.edu/`; all values in the data set have been converted into discrete values.)

```
; test the ids data set
(defun test-ids ()
  (test-decision-tree 'ids-attr.txt 'ids-train.txt 'ids-test.txt)
)
```

   TestIDS module for Java/C/C++/Python.

   (b) The TTT problem is learning what winning means in the tic-tac-toe game—imagine a child trying to learn the game for the first time. The data set contains the end games. The attributes of each record describe the board configuration and the classes are: yes (x wins) and no (x does not win). This data set is adapted from `http://archive.ics.uci.edu/ml/`.

```
; test the ttt data set
(defun test-ttt ()
  (test-decision-tree 'ttt-attr.txt 'ttt-train.txt 'ttt-test.txt)
)
```

   TestTTT module for Java/C/C++/Python.

CSE 5290 only

4. Programming: We would like the program to learn how to make a move in the tic-tac-toe game. The nine (target) classes are: `top-left, top-middle`, ... The initial nine attributes are the nine squares with `x, o, b` as values [similar to Problem 3b].

   (a) Modify `learn-decision-tree` to handle more than two classes.

   (b) Write `gen-ttt-move-data` to generate data for training and test sets, using the initial nine attributes.

   (c) Devise additional attributes to help improve accuracy, incorporate them into `gen-ttt-move-data`.

   (d) Generate a training set with at least 200 records and a test set with at least 50 records:

      i. with the initial nine attributes

      ii. with the initial nine plus additional attributes.

   (e) Test your program with both data sets.

   (f) Submit your program and data files

   (g) Submit a report:

      i. Discuss changes in the implementation to handle more than two classes.

      ii. Describe the additional attributes.

      iii. Compare the two trees and their accuracy; analyze whether your additional attributes help.

The functions (stated in LISP) include:

```
; given the sizes and filenames of the training set and test sets
;   train2-fname and test2-fname have records with additional attributes
; generate a random legal board configuration, assuming x is next to move
; the user/teacher enters the "correct" move for x
; attributes and class of each record are written to the file
; each record should be unique
(defun gen-ttt-move-data (train-size train-fname train2-fname
                          test-size test-fname test2-fname) ...)

; test the ttt data set with initial attributes, and additional attributes
(defun test-ttt-move ()
  (test-decision-tree 'ttt-move-attr.txt 'ttt-move-train.txt
                      'ttt-move-test.txt)
  (test-decision-tree 'ttt-move-attr2.txt 'ttt-move-train2.txt
                      'ttt-move-test2.txt)
  'done
)
```

TestTTTMove module for Java/C/C++/Python.