HW Bonus 02: Decision Trees
See the associated Dropbox for due date.
Thomas Kinsman

This assignment is like Bonus 01, except that a greedy decision tree induction algorithm will fail to completely separate the data in the minimum number of moves. A greedy algorithm will not recognize that it should split on attribute 2 first.

Your training program will need to look-ahead a single step in order to find the answer. AND it will need to look ahead a single step for every possible attribute, and every possible threshold of every possible attribute in order to find the best solution.

You will need to call your classifier recursively, but with a parameter telling how deep the call stack is. So, it will need to know how it was called.

This is a minor modification to the last bonus, but it is not an easy program to write. Do not attempt this if it will damage your other grades. Be careful.

- - - - - - - - - -

Homework is to be programmed only in one of the following languages. No other languages will be accepted. Please limit yourself exclusively to: Java, Python, Matlab, or R. The last three had good native graphics and plotting support.

Assume that the grader has no knowledge of the language or API calls, but can read comments.
Use prolific block comments before each section of code, or complicated function call to explain what the code does, and why you are using it. Put your name and date in the comments at the heading of the program.

Hand in:

1. Your write-up,
2. Your well commented code,
3. Your results your classification results of the test file.

You are provided with a file of training data. This data has *only two* attributes to select from. The data is completely separable. No noise is added.

Using the training data, your goal is to write one decision tree that uses only three if statements to classify the results. This means you must use binary splits. Each if statement can only test one attribute at a time. (No feature generation allowed.)

The decision tree training program:
Name this program HWBNS_02_LastName_FirstName_Trainer…
You write a program that creates a decision tree, using the decision tree algorithms we discussed in class.

Use the GINI INDEX as a measure of purity, for consistency grading.

The output of your program goes into another program.

This program must be able to read in the *.csv file that is used to train the decision tree. So, the input to the

decision tree training program is a flat file.

If you alter the training file, by deleting the header line, then turn it back into the dropbox for training.

For simplicity's sake, you can write this program to make hard-coded assumptions. It might assume that the there are only numeric attributes (which is true) and that there are only two attributes.

So, don't feel like you need to write a generic decision tree training program.

The goal is to get the best accuracy on your *testing* data as possible. But, of course, you don't know the class of the testing data. (That's the point of classification.)

You want to submit second program: called **HW_BNS02_LastName_FirstName_Classifier…**

The classifier program
HW_BNS02_LastName_FirstName_Classifier…

This resulting decision tree classifier program looks somewhat like this. You can use at most three if statements, and they must have the following structure.

```
…. header and prolog …

Given one parameter --
the string containing the filename of the training file.

read in the training file
for each line in the test_data:


   if ( attribute-a >= threshold-a )
      if ( attribute-b >= threshold-b )
            class = [ 0 or 1], you choose;
      else
            class = [ 1 or 0], you choose;
   else
      if ( attribute-c >= threshold-c )
            class = [ 0 or 1], you choose;
      else
            class = [ 1 or 0], you choose;

   # OR YOU COULD WRITE:
   if ( attribute-a >= threshold-a )
      class = [ 0 or 1], you choose;
   else
      if ( attribute-b >= threshold-b )
            class = [ 0 or 1], you choose;
      else
            class = [ 1 or 0], you choose;
      etc…

print out the class value for each line of the test data file.
```

The goal of your decision tree training program is to select attribute-a and threshold-a, then attribute-b and threshold-b, and then attribute-c and threshold-c.

You will use the data in the *training_data* file to write this program.
Then you will run this program on the *test_data* file, and guess the classification of each entry in the *test_data*.

Then you will run the classifier, have it print out one classification per line, so that the grader can quickly go down the list and compare to his classifications.

1. **Files to Submit:**
   Turn in the following code items, and support files IN ONE ZIP FILE:
   a. **HW_BNS_02_LastName_FirstName_Trainer…**
   b. **HW_BNS_02_LastName_FirstName_Classifier…**
   c. **HW_BNS_02_LastName_FirstName_MyClassifications.txt ← YOUR RESULTS!!!**
   d. **HW_BNS_02_LastName_FirstName_Writeup.pdf ← PDF DOCUMENT or *.DOCX.**

2. **Write-UP:**
   a. Show a 2D Scatter plot of your data. Your program should generate this.
      How did you design your program?
   b. Did your program need to break any ties?
   c. What stopping criteria did you use? How did you make it terminate?
      Did you hard code it?
   d. What did your final classifier look like? What was the if-else tree you got?
      The grader might not want to look at any code. Repeat the final result here.
   e. What was the accuracy of your resulting classifier, on the training data?
   f. Did your program actually create the classifier program, or did it just generate the attribute list and thresholds?