CS 540: Introduction to Artificial Intelligence
Homework Assignment #2

Assigned: 9/30
Due: 10/14 before class

Hand in your homework:

This homework includes a written portion and a programming portion. Please type the written portion and hand in a **printed** hard-copy in class. All pages should be stapled together, and the first sheet must include a header with: your name, Wisc username, class section, HW #, date — and, if handed in late, how many days late it is. The programming portion will be written in Java. All files needed to run the code (including any support files you've written) should be collected and compressed as one zip file named **<Wisc username>_HW2.zip**. This file should then be uploaded to the appropriate place on the course Moodle website.

Late Policy:

All assignments are due **at the beginning of class** on the due date. One (1) day late, defined as a 24-hour period from the deadline (weekday or weekend), will result in 10% of the total points for the assignment deducted. So, for example, if a 100-point assignment is due on a Wednesday 11 a.m., and it is handed in between Wednesday 11 a.m. and Thursday 11 a.m., 10 points will be deducted. Two (2) days late, 25% off; three (3) days late, 50% off. No homework can be turned in more than three (3) days late. Written questions and program submission have the same deadline. A total of two (2) free late days may be used throughout the semester without penalty.

Assignment grading questions must be raised with the instructor within one week after the assignment is returned.

Collaboration Policy:

You are to complete this assignment individually. However, you are encouraged to discuss the general algorithms and ideas with classmates, TAs, and instructor in order to help you answer the questions. You are also welcome to give each other examples that are not on the assignment in order to demonstrate how to solve problems. But we require you to:

- not explicitly tell each other the answers
- not to copy answers or code fragments from anyone or anywhere
- not to allow your answers to be copied
- not to get any code on the Web

In those cases where you work with one or more other people on the general discussion of the assignment and surrounding topics, we suggest that you specifically record on the assignment the names of the people you were in discussion with.

Question 1: Rules and Decision Trees [10]

Suppose you are trying to make a decision as to whether to begin your CS540 homework today or not, which we'll denote as [beginHomeworkToday] taking values Y for yes or N for no. Given the following logical statement, or rule, draw a corresponding partial decision tree, showing only the nodes represented. In each node, indicate the question asked and the question's value at that node.

IF (
([Time to do > 10 hrs]=N AND [haveAnotherHomework]=N)
OR
([Time to do > 10 hrs]=Y AND [haveParty]=N)
OR
([Time to do > 10 hrs]=Y AND [haveParty]=Y AND [wantToGetGradeA]=Y))
THEN
([beginHomeworkToday] = Y)

Question 2: Mutual Information [15]

You'd like to arrange your time for playing basketball according to the weather. Consider the following data which has three attributes: **Temperature** (with values Cool, Mild, Hot), **Humidity** (Normal, High) and **Wind** (Weak, Strong). The label is **PlayBasketball** (Yes, No).

Temperature	Humidity	Wind	PlayBasketball
Hot	High	Weak	No
Hot	Normal	Strong	No
Mild	High	Weak	Yes
Cool	Normal	Weak	Yes
Cool	High	Strong	No
Mild	Normal	Weak	Yes
Mild	Normal	Strong	Yes

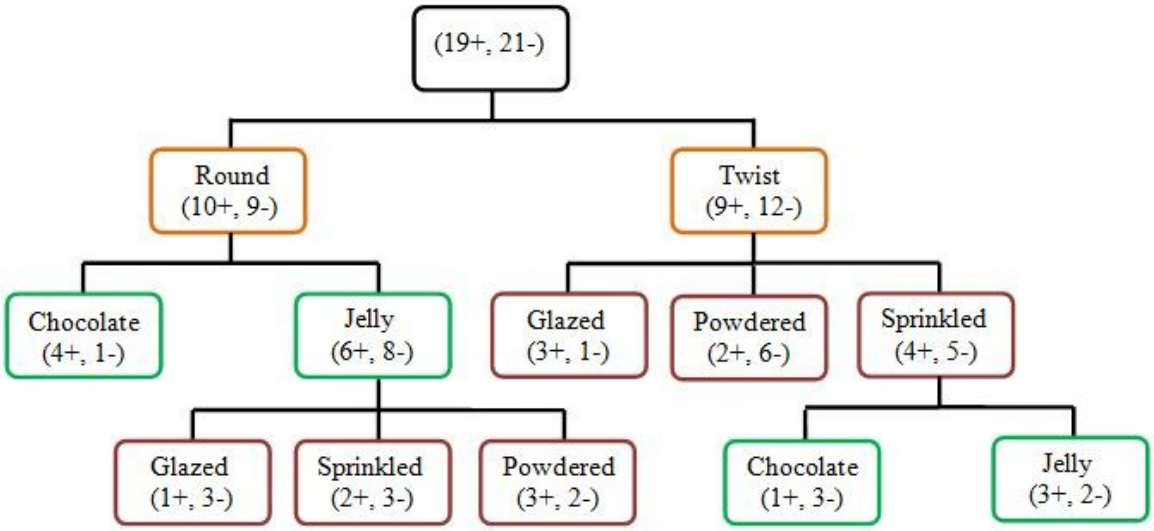
a) [3] What is the entropy of the label **PlayBasketball** given this dataset?
b) [9] Calculate the mutual information between (**Temperature** and **PlayBasketball**), (**Humidity** and **PlayBasketball**), and (**Wind** and **PlayBasketball**) respectively.
c) [3] Now we add a new attribute named **Day** to our existing dataset (keeping all else the same):

Temperature	Humidity	Wind	Day	PlayBasketball
Hot	High	Weak	Monday	No
Hot	Normal	Strong	Tuesday	No
Mild	High	Weak	Wednesday	Yes
Cool	Normal	Weak	Thursday	Yes
Cool	High	Strong	Friday	No
Mild	Normal	Weak	Saturday	Yes
Mild	Normal	Strong	Sunday	Yes

Calculate the mutual information between **Day** and **PlayBasketball**. Compare this value to the answers you got in part b). Which attribute should you choose as the split at the root?

Question 3: Pruning a Decision Tree [15]

You have been provided the following decision tree classifying the favorability (tasty=+, unliked=-) of the local selection of donuts. Donuts have been classified using the categorical attributes of Shape (Round, Twist), Filling (Chocolate, Jelly), and Topping (Glazed, Sprinkled, and Powdered). The numbers in parentheses within each node represent the number of instances of the training set that passed through that node and their labels. For example, in the training set there are 9 favorable (+) twist donuts and 12 unfavorable (-) twist donuts. Your decision tree returns a classification by taking a majority vote of the label value among the training data at the leaf reached.

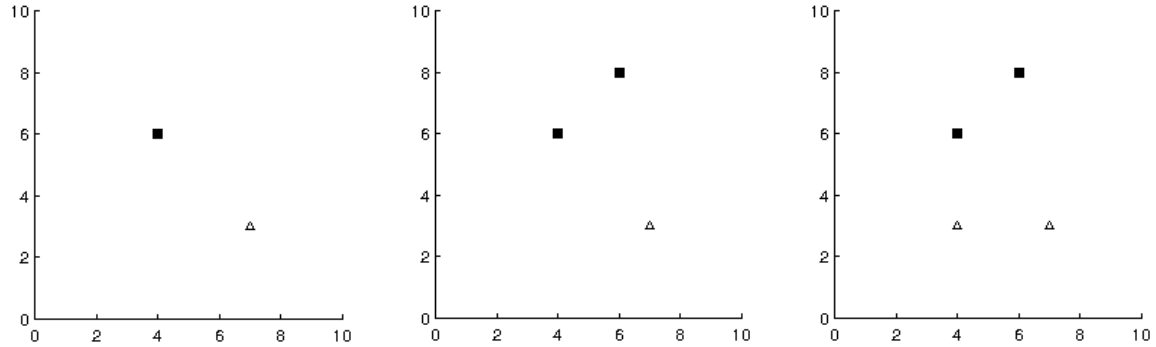


You have been provided the tuning set as follow:

Round	Glazed	Jelly	+
Round	Sprinkled	Chocolate	+
Twist	Glazed	Chocolate	-
Round	Powdered	Jelly	-
Twist	Powered	Jelly	+
Twist	Sprinkled	Jelly	-

a) [3] Explain how pruning potentially reduces overfitting on the training set.
b) [9] For each node available for pruning in the given tree, list the path to that tree, then calculate the tuning set accuracy using the tree with that node pruned.
c) [3] Which node should you prune first?

Question 4: SVM [20]



(i) (ii) (iii)

The plots above show two classes of points: solid squares (y = +1) and triangles (y = -1).

a) [10] Visually estimate an SVM classifier for each plot by drawing, **by hand,** the decision boundary with max-margin. Also circle the support vectors. Assume we use a linear kernel function,.

b) [10] Find the SVM solution {W, b} for the first plot (i) such that the margin is maximized. The solid square has values $X_1 = (4, 6)$, $y_1 = +1$, and the triangle $X_2 = (7, 3)$, $y_2 = -1$. Report the values of {W, b} that you find.

Question 5: [40] Building Decision Trees in Java

For this part of the assignment, you are required to implement in Java the decision tree construction algorithm given in Figure 18.5 on page 702 of the textbook.

The Dataset:

The dataset you will learn to classify is a set of credit information collected in Germany. It includes personal finance status for many individuals and the prediction as to whether or not their credit is considered good. Information which, for instance, a bank might want to determine if they should provide a loan or not to an individual.

There are a total of 300 records, or instances, which have been divided into: a training set of 200 instances, a tuning set of 50 instances, and a test set of 50 instances. Each record is composed of 7 attributes, each attribute related to the financial status of an individual. Some of the attributes are categorical while others are numerical. You will use some combination of these 7 attributes to build your decision tree. Please see the data description file (german.desc) for more information, including attribute names and values.

Source Files:

You have been given a set of files that handle input from the data files and provide output functionality. See the comments in each file for a description of the provided functionality. **The only file you are allowed to edit is DecisionTreeImpl.java**. In this file there are four methods whose functionality you must implement. Any changes to the others will be overwritten by the testing script. If you require any additional code, you can supply this in additional java class files. Just be sure to include these extra files in your hand-in package.

The 4 methods you must implement are:

1. DecisionTreeImpl(DataSet train) // Build a decision tree given only a training set.

2. DecisionTreeImpl(DataSet train, DataSet tune) // Build a decision tree, then prune it using a tuning set.

3. String[] classify(DataSet test) // Return the predicted classifications on the test set, using the current tree

4. void print() // Prints the tree in the specified format.

Deliverables:

Place all code necessary to run your program into a zip file (as specified on the first page of the homework) including your modifed DecisionTreeImpl.java and any additional java class files needed to run the program.

Your code should provide the functionality described in the questions below.

Your submitted code must compile on a university computer using javac. This can be done by placing all .java files in a single directory and running the command “javac *.java” in that directory. Please do NOT put your code in a package. Otherwise, it will likely fail to compile and that will be a mandatory deduction.

Running the code:

Your code will be tested using the wrapper class HW2 with the following calling convention:

```
java HW2 <modeFlag> <trainFilename> <tuneFilename> <testFilename>
```

`modeFlag`: takes values 1 or 2 (see question parts a) and b) for more details)

1 : print a learned decision tree, and then print the accuracy calculated on the test set

2 : print a learned decision tree after pruning, and then print the accuracy calculated on the test set

`trainFilename` : path to the file containing the training set

`tuneFilename` : path to the file containing the tuning set

`testFilename` : path to the file containing the test set

Notes:

- For each question, only output the requested lines. Comment out any debugging statements or additional extraneous print statements.

- To eliminate ambiguities in the algorithm defined in Figure 18.5 of the text book, in the case of a tie in majority vote (PLURALITY-VALUE), you must always choose the positive label.

- In the case of a tie of mutual information values when selecting the best attribute, you must choose the attribute with the higher attribute index (i.e. 7 to 1, as defined in german.desc).

- Child nodes should be printed in the same order of the attribute values listed in the training file.

- Once again, your submission should consist of a single folder with only uncompiled class files (.java) inside.

- The only file from the provided set you can modify is DecisionTreeImpl.java and you must modify it.

- You are allowed to add your own class files. When added to the provided set, everything must compile on a CSL Linux machine using the javac command. You can import useful tools from the Java standard library to facilitate your work. No third-party libraries may be used.

- Additions or modifications of this assignment will be posted to the class forum on Piazza. Please post questions about the code there.

Questions:

a) [20] When modeFlag = 1, use the training set to build and print out a decision tree (without pruning), splitting on all potential attributes. Report the accuracy on the test set.

- To handle the continuous attributes (the 5^{th} and 6^{th} attribute), you must discretize them. For this program, use the following method: First, calculate the midpoint of the i^{th} attribute's value using:

$$midpoint = 0.5*(max(i) + min(i))$$

where *max(i)* and *min(i)* are the maximum and minimum attribute values in the subset of examples at that node (not the entire dataset). Second, each example must then be given the discretized value A if the attribute value is less than or equal to the calculated midpoint, and B otherwise.

- To draw the tree, use simple indented ASCII text to indicate a node's level in the tree during a depth-first tree traversal. At each level, indent by 4 blank spaces. Here is an example what the tree should look like:

```
Root {Existing checking account?}
    A11 (2)
    A12 {Foreign worker?}
        A71 {Credit amount?}
            A (1)
            B (2)
        A72 (1)
    A13 (1)
    A14 (2)
```

- Note that internal nodes include both the attribute value at that node and the next question asked, in curly braces. Leaf nodes include the attribute value at that node and the label value, in parentheses.

b) [20] When modeFlag = 2, train a decision tree in part a), but this time perform pruning as described in the class slides. When tuning is complete, print out the decision tree as in a). Report the accuracy on the test set.

Important:

1. We have written code to read the training data, tuning and test data for different tasks according to the modeFlag. Your task is to implement the building of the tree, pruning the tree, printing the tree, and classifying the test instances.

2. When implementing the print statements described above, use the attribute names and values found in the data description file german.desc. For discrete attribute values, just print the attribute value as given, for example A10, A23. For continuous attributes, you should use A to indicate that the attribute value is less than or equal to the mid-point, and B otherwise, as described above.

3. Note that in the training and tuning data set, the class label is represented as a number 1 or 2. When printing the label of a leaf, print these label values (1 or 2).

4. In the `print()` method, you just need to implement the algorithm to print out the tree. The calculation and printing of accuracy is done for you (See `calcTestAccuracy()` in HW2.java). In order to run `calcTestAccuracy()` when calculating accuracy on the test set, please make sure that the return value String[] of `classify()` is the same size (has the same number of instances) as the test set. The first element of the String array should be at index 0.