Artificial Intelligence Lab C - Decision Trees

This lab has been adapted from David Musicant's work.

Introduction

This Lab is worth 60 points. Please follow the Notes for All Labs carefully. The lab should be done in **Python 3**.

This is a pair-optional assignment. If you are working with a partner, you and your partner should work together for all portions of the assignment, unless otherwise indicated. You should not split it up and each do half. This lab is a significant amount of work, and may overlap with reading assignments and short discussion assignments.

Deliverables

Your **deliverables** are one or more code files, a README, a design plan, and a .pdf report file. All submitted work must be in the indicated file, or it will not count. If you wish to write a script that tests your code, you can also submit that. Indicate how to use it in the README.

Deadlines

This lab has **two** deadlines.

You should submit your design plan no later than **March 27th**. This *will* be graded at this point. On the same day, **March 27th**, you should submit an initial attempt at an implementation. This will *not* be graded at this time, but should reflect significant work.

Finally, you will submit a finished version on **April 3rd**, which can include revisions to your design plan based on things you learned along the way. Please let me know if you have any questions

The reason for these intermediate deadlines is to allow class discussion to address tricky spots in the lab. The expectation is that, after these deadlines, you will have explored the problem set in enough detail for class discussions of the lab to be useful to you.

Background

Before beginning the lab, please carefully read **Chapter 18.1-3 of Russell and Norvig**. This lab will be exceptionally difficult if you do not understand what it is you are building, so be sure to ask me any questions you might have.

In this lab, you will write a program in Python to implement a decision tree algorithm - specifically the DECISION-TREE-LEARNING algorithm in chapter 18 of your text.

Design Plan

The first thing you will produce is a **design plan**. This may be in any format you like; however, it should be clear enough and detailed enough that somebody could reasonably implement the code primarily by just using your design document. It can and should include charts of how the different components of your system will connect to each other and comments about things you want to make sure you remember during implementation. The design plan should be submitted as a typewritten .pdf, a scanned .pdf, or some combination of those things. It should include photographs only if that is the only reasonable way to capture what you've written. (For example, whiteboard notes.)

Data sets

Your code should read in a tab-delimited dataset, and output to the screen your decision tree and the training set accuracy in some readable format.

There are three sample datasets you can try: tennis.txt, titanic2.txt, and pets.txt. They are located on Moodle in the "Lab C Data Files" folder. They are also located in ~barbeda/cs365-share/lab-c/ on the shared course server.

(Sources: titanic2.txt is from Vanderbilt University Department of Biostatistics)

The first line of the file will contain the name of the fields. The last column is the classification attribute, and will always contain the values **yes** or **no**. All files are **tab delimited**.

Running Your Program

When you run your program, it should take a command-line parameter that contains the name of the file containing the data. For example:

```
python3 decisiontree.py tennis.txt
```

Program Behavior

For output, you can choose how to draw the tree so long as it is clear what the tree is. You can use text if you are not familiar with any other ways to draw your result. You might find it easier if you turn the decision tree on its side, and use indentation to show levels of the tree as it grows from the left. For example:

```
outlook = sunny
|  humidity = high: no
|  humidity = normal: yes
outlook = overcast: yes
outlook = rainy
|  windy = TRUE: no
|  windy = FALSE: yes
```

You don't need to make your tree output look exactly like above: feel free to print out something similarly readable if you think it is easier to code, but it should be clean and unambiguous.

If your tree cannot distinguish some of your examples (because they have the same attributes but different classifications), it should return the plurality classification (the winner of the vote). If it's a tie, it should return 'no'.

For full points, your tree should not do any branching that does not reduce entropy.

Accuracy Testing

To test our accuracy, we will use leave-one-out cross-validation. To do this, we generate a tree using the entire training set except for one of the examples, then see if the tree classifies that one correctly. We repeat this a total of n times, where n is the number of examples in the training set, leaving out a different example each time. We then report the accuracy of the ones we tested on, as described in the Report section.

One challenge you may find when using leave-one-out cross-validation is that there may be a particular value in the dataset that only exists in a single point. This value then is seemingly not in the training set when you go to build your decision tree when that point is left out for cross-validation purposes. You'll need to handle this appropriately; when building the decision tree, your program will need to handle all potential values in the dataset without breaking, even if they don't appear in the training set for a given fold.

FAQ

How do I do logs in Python?

See here. In short, you will need to import math, and then do something like log(.5, 2), where 2 is the base, and .5 is the value you wish to take the log of.

My mentor has used this assignment in the past. Here are some FAQs that he has gotten regarding this assignment:

Is it possible that some value, like "normal," could appear in more than one column?
Yes.
Could "yes" and "no" appear as possible values in columns other than the classification column?
Yes.

Report

In your **report**, include a copy of the trees you generated from each training set I provided, along with the **training set accuracy** and the **number of nodes** in the tree. The training set accuracy is computed by building a tree using the entire training set and then checking what percent of the training set is classified correctly by this tree. For a set where all of the examples are unique in their features (like tennis.txt), this should be 100%.

Run your with leave-one-out cross-validation, and measure the **test set accuracy** of all three data sets. Give these values in the **report**.

Going Further: The ID3 and C4.5 algorithms are more sophisticated ways of building decision trees. If you are interested in going deeper into this topic, you may try implementing those. (C4.5 is a direct evolution of ID3, so start with ID3.) C4.5 is the dominant decision tree algorithm in use today. If you choose to do this, please still submit a version that uses the basic algorithm, and make it clear in the readme how to run each. This is not worth any extra points or anything.