CS-349, Fall 2023
Homework #1: Decision Trees
Due Date: Tuesday, October 10th @ 11:59PM
Total Points: 20.0 (plus optional 2.0 bonus points)

In this assignment, you will work with your project group to implement decision trees. You may discuss the homework with other groups, but do not take any written record from the discussions. Also, do not copy any source code from the Web.

The algorithm you should implement is the same as that given in the decision tree lecture slides (slide 4-5, the "ID3" algorithm). We have written code to read in the data for you (`parse.py`). It represents each example as a dictionary, with attributes stored as key:value pairs. The target output is stored as an attribute with the key "Class".

Guidelines

- You should use Information Gain for choosing which attribute to split on.
- You must handle missing attributes, but exactly how is up to you.
- You must implement some kind of pruning, but exactly how is up to you.
- You must adhere to the signature in the ID3.py file. In particular, you must implement the four given methods as described in that file. You can and should define additional methods in ID3.py, and additional fields in the Node class.
- You do not need to handle numeric attributes, but your code should work for categorical attributes with an arbitrary number of attribute values, and an arbitrary number of output classes.
- Do not import any modules from outside the Python standard library. If you need other modules to produce e.g. graphs, write that code in a separate source file (which you should not turn in).
- If all available splits have zero information gain, you should prefer a split that is non-trivial (i.e. one where all examples do not have the same attribute value).

Steps to complete the homework

- (10.0 points) Complete the decision tree code. You should use Python 3.6 or later. Implement the four methods in `ID3.py`, adding new methods as necessary. You will also need to change `node.py`. We have included a few tests in `unit_tests.py` that you can run individually, to check your methods. Further, we have also included a file `mini_auto_grader.py` that gives you an idea of the kinds of tests we will run to grade your methods. *NOTE: that depending on how you implement pruning, it's possible (albeit unlikely) that the pruning test will not pass even if your implementation is acceptable.* We will test you code on an unseen dataset with random training, validation and test sets. The `tennis.data` dataset is also provided to help you debug your code on a much smaller and simpler dataset.

- Create a PDF document with the answers to the following questions (try to keep you responses to a few sentences or a single paragraph for each question):
    1. (2.0 points) Did you alter the Node data structure? If so, how and why?
    2. (2.0 points) How did you handle missing attributes, and why did you choose this strategy?
    3. (2.0 points) How did you perform pruning, and why did you choose this strategy?
    4. (4.0 points) Now you will try your learner on the `house_votes_84.data`, and plot learning curves. Specifically, you should experiment under two settings: with pruning, and without pruning. Use training set sizes ranging between 10 and 300 examples. For each training size you choose, perform 100 random runs, for each run testing on all examples not used for training (see `testPruningOnHouseData` from `unit_tests.py` for one example of this). Plot the average accuracy of the 100 runs as one point on a learning curve (x-axis = number of training examples, y-axis = accuracy on test data). Connect the points to show one line representing accuracy *with* pruning, the other *without*. Include your plot in your pdf, and answer two questions:
        a. What is the general trend of both lines as training set size increases, and why does this make sense?
        b. How does the advantage of pruning change as the data set size increases? Does this make sense, and why or why not?

        Note: depending on your particular approach, pruning may not improve accuracy consistently or may decrease it (especially for small data set sizes). You can still receive full credit for this as long as your approach is reasonable and correctly implemented.

    5. (optional 2.0 points) Use your ID3 code to construct a Random Forest classifier using the `candy.data` dataset. You can construct any number of random trees using methods of your choosing. Justify your design choices and compare results to a single decision tree constructed using the ID3 algorithm.

Suggestion: You may find it helpful work with Weka (https://www.cs.waikato.ac.nz/ml/weka). To work with ID3 in Weka, you may need to install the simpleEducationalLearningSchemes. A large number of Weka tutorials exist on the Web.

Submission Instructions

Turn in your homework as a single zip file, in Canvas. Specifically:

1. Create a single pdf file ps1.pdf with the answers to the questions above, and your graphs.
2. Create a single ZIP file containing:
    - `ps1.pdf`
    - All of your `.py` code files
3. Turn the zip file in under Homework #1 in Canvas.

Good luck, and have fun!