Alex Childers

PA #4 Reflection Essay

2019/04/08

If I were to rate my work on this assignment, I’d give it a 7.2 out of 10. While some of my higher-level design choices were questionable, I *was* able to create a fully-functioning program with some extra, real-world test cases, as well as taking the time to add some extra functions that made the program a little bit better. For instance, I wrote a function to randomly select rows from the whole data set, and I was able to give the user feedback on what percentage of predictions from the decision tree were actually correct.

One of the most obvious struggles I had with this assignment was deciding how to implement the tasks that involved taking user input and responding to it, serving as a sort of interaction between the user and the underlying decision tree/files. In the last few assignments, I’ve been creating a separate Tasks.h file specifically to contain code that I write for the purpose of making the program usable. However, I wasn’t sure if it made sense to just dump all my own functions into a catch-all header file, so I put all my user-interfacing functions in main.cpp for this assignment. While it was nice to have all my objects available for use in the same file, rather than bouncing control back and forth among main.cpp and the various header files, I think this made main.cpp far less readable, with many more lines of code than I would care to read in such a central file. I suppose this just means that I’m still learning how to abstract out my code appropriately.

Additionally, as ever, algorithmic efficiency was a struggle for me. I was particularly concerned when all my attempts to run the program on the 50,000 row data set of loan information resulted in a stack overflow. I spent a few hours Googling solutions for stack overflow, learning exactly what that term meant, and various ways I could solve it. I was relieved when I learned that others in class were having the same issue in C++. However, I am grateful for the time I spent trying to fix my code around such an issue, because I am now more familiar with the concept of a call stack itself and how recursion affects the memory usage of a program.

I noticed that, in this assignment, all the work of implementing the ID3 decision tree algorithm itself was given to me in class, while all the user interaction was left for me to program. I’ve been thinking about whether this helped me learn better or not, and reflecting about how well (or poorly?) I performed on the midterm section about decision trees. I really enjoyed this assignment in particular, and I’m wondering if that’s because I found the algorithm more interesting, if it was more fun because I could spend my time making the user interface look nice and detailed to my ideal standards, or if it was just easier for me to write trivial code that interfaces with the user, rather than having to figure out an ID3 implementation by myself. Would I have internalized the algorithm more thoroughly if I were forced to program it from scratch, or would I have just taken longer to get to the same understanding I’m at now? In the end, I think I do appreciate having the core algorithm programmed out for me so that I have a clean, working example to study at my own pace. Having the core algorithm already pre-programmed for me also gave me time to delve into the knowledge surrounding machine learning—I had a lot of fun trying to find large data sets online for testing my program, and stumbling across entire repositories of data sets that were specifically helpful for machine learning. Maybe I didn’t learn as much about the decision tree algorithm itself as I could have, but I’ve learned a really interesting assortment of other things throughout this assignment.