CS 484 Data Mining Homework 2

Theodore Church – G01127117

Miner Name: who

Rank: 96

Accuracy: .77

Approach:

I started super late, I’ve had some life issues the past few weeks which has resulted in focus issues, I’ve been playing “catch up.” Losing spring break didn’t help me at all. This is important because I did the entire homework in a single day, that means I had 5 attempts for submission. I split my 5 submissions into 1 Knn, 1 Bayes, and 1 Tree, then I was going to look at which preformed the best without further analysis and see if I could improve that. I didn’t do much preprocessing of numbers. All I really did was take the last column off the training set and turn it into the set answers. Some of the columns are binary variables. If I had more attempts I’d run tests on the binary variables alone to see how important they are, maybe excluding one or two to see how that changed accuracy. For all of this I used pandas to read the csv file and format and each function calls scalar to normalize the matrix. **SPECIAL NOTE**: the tester needs to change the line of code for their file directory, it’s currently using my file directory.

My initial accuracies for the models used:

Knn: .77

Tree: .72

Bayes: .44

With these numbers I decided to analyze KNN further to see if I could push myself into the 80s. Before I talk about that I’ll mention some methodologies for Decision trees and Bayes. I used the Gaussian Byes and figured that with an accuracy as low as sub 50 it wasn’t worth pursing. I think Bayes would work better with text and I’m considering trying it on homework 1. I noticed upon closer inspection that I made a mistake with my byes implementation. I ran the unprocessed code(pre scalar) for the algorithm. The mistake has been commented and the correct implementation has been added. My Decision tree I wrote with 7 leaves. Writing it with less causes the tree to not have all 7 classifications. I did not try writing it with more. I picked entropy, gini didn’t seem to have a wide variance of answers when I briefly checked the file. I’m going to come back to decision tree later.

For KNN I initially started with a K value of 5. No reason, just a starting point to be improved upon. Later I implemented the code on the slides to check the optimal K value and observe the differences in the k values. The average accuracy I was getting was about .75. 0.7460426703372333 to be exact. Looking at the graph the lowest error rate was at K = 2 so I switched my KNN value, ran it again and submitted my results to get: .76. Lower than what I had initially started with. I tried again for a repeated .76. I’m unsure if this is just an unlucky partition of data and my actual accuracy is higher (lower?) or if I looked at my data wrong. I had misinterpreted my data, [0.22918100481761872, 0.24982794218857537, 0.23227804542326222, 0.24122505161734342, 0.23916035788024775, 0.2477632484514797, 0.24501032346868548, 0.251892635925671, 0.2539573296627667, 0.25843083275980727] are the first 10 entires, K = 1 should be the lowest error rate so I made a mistake when I looked over my data. Further more looking at the leaderboard some students have managed an accuracy in the 80s. Thinking about this and the error rate for my KNN I’m led to believe 2 things: First is that I’m either near the cap of accuracy for KNN or I need to process the data more. Secondly: If Byes is so low, and KNN seems to be peaking in the low 80s, I wonder if a decision tree would be a more optimal approach to accuracy? If I had more time/attempts I’d be looking into changing my implementation for decision tree to see if I can optimize it’s accuracy.

Code Efficiency:

My KNN takes less than a minute, Byes and Decision tree both finish almost instantly. The KNN cross validation(listed as accuracy) also takes less than a minute. Some students have mentioned that their code takes 5-10 minutes to run if not more and mine finishes in a minute or two. I was satisfied with this efficiency, so I didn’t feel the need to try and optimize run time. That being said, cross validation is the slowest of the bunch and if runtime was an issue I’d either run cross validation alone, or look into using a smaller portion of the training / test data to cross validate. This might effect accuracy though so I’d have to play around with it more.