Description

We'll again use the writing portfolio data for this assignment, so that you can compare performance that you get with Naive Bayes to what you got with k-nearest neighbor. Who will win?

Again using the Carleton writing portfolio data in `/Accounts/courses/cs324/writingportfolio.xlsx`, write code to predict whether or not a student would pass or fail, and estimate your accuracy. One critical thing to do is to make sure that when you are testing an individual student, you should be leaving that student out of the estimates that you do for the probability calculations. This is just like the k-nearest neighbor example; in that case, you made sure that an individual student wasn't eligible to be their own nearest neighbor. Similarly, when doing probability calculations where you're pretending that you don't know the actual classification, you shouldn't be using that student in the probability estimates.

This would seem to mean that you'll have to recalculate probabilities for each student, rather than getting to preprocess them up front. Actually, you can be smart about this. Think about how to preprocess what you need to calculate the probabilities (or related values), yet adjust them appropriately for each individual student.

You'll also need to figure out how to handle numeric data. Should you treat it as discrete? Should you use bins? Or should you use a Gaussian distribution to estimate probabilities? This is up to you, but do something intelligent. Justify your choices in the writeup. You might try experimenting with different options to see what gets you the best accuracy.

In addition to submitting your code, submit a written "lab report." Explain the choices that you've made on each of the numeric fields, and how you handle them. Also describe the accuracy you ultimately obtain, and how it compares with what you did with k-nearest neighbor.

Grading

I'm giving you significant flexibility in how to set up your classifier. Hopefully, that's interesting and fun. It does make grading more challenging, because we can't just run it and see if you get the "right answer." Therefore, I'm going to be instructing the graders to look at your code in order to verify that it appears you're doing the calculations correctly. You must make it as clear in your code as you possibly can how you are doing the calculations. Write your code in a way so as to be as clear as possible, and supplement with comments to explain what you are doing. Your job is to make a convincing argument with your code that you are doing this correctly. I'll be instructing the graders that they can take points off if they cannot determine to their satisfaction that the calculations are correct; this could be because the calculations aren't right or because the code is hard to understand (or both!).

Something else the graders will be looking for is verifying that you are smart about generating the probabilities; both with regards to not including an item in its own probability calculations, as well as not rebuilding all probabilities from scratch each time. Again, make it as clear as you can what you're doing with well-written code and with program comments.

Finally, the graders will also be looking at the quality and clarity of your code in general (apart from the Naive Bayes piece), as well as the lab report.