Jericho McLeod
CSI-777
Chapter 5 Review

In terms of validation methods, k-fold validation, n-fold validation, and bootstrapping are things I have covered at length in prior courses. Conversely, I have never been formally told how to find hyperparameters, and am a bit disappointed that my default method is the one suggested by the book. That is, I previously ran experiments at length whenever I needed to find hyperparameters using various methods of validation to compare results, but found it to be very time consuming, especially with larger datasets. I think I was secretly hoping for some succinct rule of thumb for subsetting data for hyperparameter tuning.

Confusion matrices are also a review, and lift charts are something I have used first hand professionally. As I mentioned in class, we used an estimate of lift to determine whom to contact regarding the value of retirement assets given specific actions by using a set of risk factors and prior response rates. We had two data sets; one of which accounts were likely at risk, and another of which accounts responded to prior mailings of offers and surveys. Combining the two, we limited our mailings from our entire customer pool of ~150,000 individuals to only 1,000, but we saw loans and distributions drop by a much larger percentage.

The MDL principle section was unusual in that it explained MDL, but did not define it. The only times the phrase ‘minimum description length’ is mentioned is in the prior chapter, the abstract and keywords list of this chapter, and in the remarks section of the ‘Predicting Probabilities’ section of this chapter. I did not remember the acronym that was only defined once in the prior chapter and once 15 pages earlier in this chapter.

That said, it is interesting that I had made it this far into studying various aspects of ‘data’ as well as numerous mathematics and statistics courses without hearing this term, especially given the general emphasis on reduction and simplification wherever possible. While the book relates the concept to clustering, to me it is more relevant in terms of decision trees and neural networks at the moment. Decision trees naturally build to a maximum for precision, and then pruning can be done to mitigate overfitting, which is relatively straightforward but is something I have been considering lately as I’m building a library for decision trees. For neural networks it gets a bit more complex and relates back to information loss: how many bits can classifications be stored in? I had initially set up 10 different output nodes for the ANN I am building for CSI-873, which needs to classify things into one of ten categories. After some consideration I converted my outputs to binary, reducing the output to 4 nodes, primarily because I was tired of troubleshooting the results from 10 every time I had an error, thus put forth the effort to reduce the model. This is, perhaps, more evidence that lazy developers are ideal because we seek the easiest ways to solve problems:
https://cacm.acm.org/blogs/blog-cacm/238155-lazy-developers-are-the-best-developers/fulltext