Nahom Ogbazghi
March 6, 2017
C 4.5 and Naïve Bayesian Algorithms

C4.5

For this algorithm, I created a node that contains, the splitting column number, the value of that splitting column, list of data after the split, the parent node, and a list of children. As the data is being split nodes are created to hold that split, and only at the very end where the leaf node will be found, the node for the leaf will only contain the attribute 0 and either a p or e for the value of the leaf. Using the mushroom data set, I did not attain a 100% accuracy, most of the leaf nodes that would contain p did not print. I suspect that my error must do with my fringe when testing the test data. The values that did print appear to be accurate in comparison to the J48 algorithm in Weka.

Naïve Bayesian Algorithm

For this algorithm, I generated a P(X|Ci) for both the classifying labels P and E within a hash table. So, for keys p and e, I generated a list of dictionaries per column relative to it's classifier. After testing I concluded that I needed to implement Laplacian Correction, which I did by iterating through both p and e lists of hash and filling in the inconsistencies between the list of hashes. Finally, in testing, I calculated the final P and E values and compared them to determine the class for each row. I could attain a 99.7% accuracy in classifying the test data set. Using the Naïve Bayesian implementation in Weka yielded a 95% accuracy.

Weka

For Weka, I used the SMO and SGD algorithms in Weka on the mushroom dataset. Per Weka, I used a SVM and these algorithms attained a 100% accuracy when classifying test data, but they did take longer to run than both C4.5 and Naïve.