Joseph Barbati
CS378

Classification – Assignment 3

Bayesian Classification

My Bayesian classifier implementation very simply finds the solution to the following formula: P(X|Ci) = P(x1|Ci) x P(x2|Ci) x … x P(xn|Ci). Where X is a random variable X = {x1, …, xn} and Ci is the class whose probability is currently being found. I apply this formula for all possible classes and then take the maximum probability (P(X|Ci)) found and the class associated with that probability (Ci) is returned as the guessed class.

As you can see in res.txt, the Bayesian classifier correctly classifies 95.72% of the transactions in the mushroom test dataset.

C4.5 Classification

The C4.5 Algorithm is a recursive algorithm to use a training dataset to create a decision tree to classify data.

First, let me describe my Node class. This class takes a value (possible entry into the column), a column number, a results dictionary, a true child, and a false child. The results dictionary has keys which are the possible classes and values which are the frequencies of those classes for a given path through the tree. The true and false children are children of the node where if a row at the node's column number has the same value as the node's value we enter the true child node and enter the false otherwise.

The build_tree method takes a set of rows.

The algorithm to build the tree is as follows:

1. Check if the set of rows is empty, if so return an empty Node (all variables set to None/null)
1. Calculate the info(D) of the set of rows.
2. For each column and for each value in that column:
    a. Partition the set of rows based on the value.
    b. Calculate gain ratio on the value
    c. Choose the greatest gain ratio and break out of both loops
3. Check to make sure the gain ratio is still greater than 0
    a. If so, call build_tree on the 'truth' set and then on the 'false' set.
    b. Then return a Node where the value is the value of the highest gain ratio calculated in 3 and column is the column with this value. None/null for the results and the 'true' and 'false' children.
4. If gain ratio is 0 or less (shouldn't ever be less), this is a condition to stop partitioning. Return a Node with arbitrary value and column, no 'true' or 'false' child as it is a leaf, but the results are in a dictionary as described above.

In order to classify the node, we need a transaction and the root node.

1. Check if the current node has results
    a. If so, get the 'decision' i.e. the row's value at the current node's column number. Check if the decision equals the current nodes value.
        i. If so, call classify recursively where the row is the same but pass the 'true' child node in for the node.
        ii. Else, call classify recursively where row is the same but pass the 'false' child as the node.
    b. If results exist in the node, return the value (class) with the highest frequency as majority rules. This is the guessed class by the decision tree.

As res.txt shows, the accuracy for this algorithm is, amazingly, 100%. I'm sure with a greater number of test transactions the accuracy would depreciate, but as it stands, the accuracy is 100%.

Potential Issues

My program doesn't handle .arff files, so it has no way of knowing all the possible values of an attribute. It solely relies on the training data, so if the test data shows a value not in the training set, it will throw a KeyError (regarding a key in a dictionary).

Weka

Using the J48 method (which is a Java implementation of C4.8), After building a model with the training dataset, the test data set was classified with an accuracy of 62.05%. My own model produced 100% accuracy, which is much better than the Weka model. Weka's performance, on the other hand, was better as Weka produced a model in around half the time my algorithm did.