Courtney Wood
Artificial Intelligence
Coursework 2

Artificial Intelligence Coursework 2 Report

Train:
My program trains recursively. The train() method initializes the decisionTree variable with null and -1, because these values are set later. It then calls the trainRecursive() method, passing in the decisionTree and the training data.

The parameters for trainRecursive() are the current node, the current subset of the training data, and a list of already used columns. I chose to keep track of already used columns so that it does not try to divide the subset based on the same attribute more than once. For example, if it divides first on Outlook, then divides all Windy examples by Temperature, it will not divide any subsequent subsets by Outlook or Temperature again.

The base case for trainRecursive() is the case where all items in the class have the same classifier. If this is the case, it creates a leaf node and the recursion stops. If this is not the case, it finds the next attribute to divide the remaining data by calling findNextQuestion(). findNextQuestion calls split() for every attribute. The HashMap returned from split(), along with the value returned from calculateHS() (used to find the H(S) value for the remaining subset) is used to calculate the gain for each column of the data. The attribute with the highest gain is returned, and this is the question that divides the remaining data into subsets. If at least one subset is empty, the majority class is found instead of dividing the data. If not, trainRecursive() is called on each subset of the data, as long as no subsets are empty. If any are, the majority class is found.

Classify:
Classify also uses recursion. For each example in the test data, classify() calls classifyHelper(). classifyHelper() is a recursive function that traverses the tree according to the data in the example. The base case for this function is if the node has no children, which means it is a leaf node. It returns the location of the classifier in strings, which classify() prints to the output file. If the node is not a leaf node, classifyHelper() finds the child node that is correct for the current example, and calls classifyHelper() on this node.

Although I generally implemented the ID3 algorithm, I made some unique design choices. For example, I chose to have split() return a HashMap mapping Strings to Hashmaps because I figured this would be the fastest and most efficient way to access this data later in the program. Hashmaps are very fast for lookup and retrieving. I also chose to have a parameter in trainRecursive() that is an ArrayList of already used columns so that the program could not choose a question it has already used. However, different branches are allowed to ask the same question, so I had to make deep copies of the ArrayList so an addition in one branch would not add it to the ArrayList in another branch and mark it as already used. I chose to find the majority when a subset is empty because it is impossible to further divide an empty subset.

I tested the program by using multiple different training and test files that were provided, that I found online, or that I created myself. I also modified the training files to test how the program handled duplicates, or duplicates with different classes. There are no known bugs.