ID3 mini project

The purpose of this project is to implement an ID3, a decision tree learning algorithm in Java to classify objects with discrete values (String). The algorithm needs to train and build a decision tree based on labeled examples and then classify the unlabeled examples.

The ID3 is implemented within constrains provided, as required, none of the predifined methods or data structures have been modified, the algorithm is able to classify all specified legal input conditions (one, three or more classes, or data which can not be perfectly classified).

My train method is calling setTraining() method which I separated in order to create and work with two new arrays and manipulate “unused” data(rows and columns). It starts with checking if there is data left in the training set and if there is no more question to be asked it will return the TreeNode with most frequent label. As required for ID3, setTraining() is a recursive method and it builds on findSplitAttribute() method. findSplitAttribute() method will calculate the entropy and gain for each attribute and return the one with highest information gain. The setTraining() will take care of creating a branch for each string of the chosen attribute once the best attribute to split on is found. Then, it will call itself recursively, taking care to add the split attribute to the set not to be used again, like this it will not look at the same attributes already used. In the end it will return a new TreeNode with subsets and the best attribute for the split.

In other words, on each iteration, the algorithm will go through every unused attribute of the set and calculate the information gain all over again of that attribute for the given set. Once it chose the one with highest information gain it will split the set, or partition the selected attribute to produce the data. The algorithm continues to recur on each subset, considering only attibutes never used or selected before. Recursion on the subset will stop if either a) every element in the subset belongs to the same class (isSame() – in which case the node will be turned into a child (leaf) node and labelled with the class of the examples), b) there are no more attributes to be selected, even if the examples don’t belong to the same class (findMostFrequent() -in which case, the node is made into child node and labelled with the most common class of the examples in the subset), or c) there are no examples left in the subset (for example when there is no example in the parent set was found to match a specific value of the selected attribute).

The classify() method iterates through the branches of the tree in training set and finds a children node that is equal to the attributes in a given row. Each node has a list of leaf (children) nodes, and to navigate the tree, the method has to find the index of each value in the strings array rather than returning the value of the string.

Throughout the algorithm, the decision tree is constructed with each node which has at least one child representing the chosen attribute on which the data was split, and terminal nodes (leaf nodes) representing the class label of the final subset of this branch.