Kyle, Sean, Tristan

Prof. Fitzsimmons

AI

9 December 2022

Project 3

The decision tree is made up of nodes that have parameters of value, parent, children, and attributes. The value of each node is the name of either the attribute or branch. In our case, for the restaurant example, the first node's value would be 'Patrons', and for the house-votes example the first node's value is "physician-fee-freeze". The Node class also includes children nodes stored in the form of a dictionary, where each key for the dictionary is the name of a branch, and each value of the key is the next node in the tree to be recursed on. The attribute of each Node is a list of the possible choices/branches that the Node can go to. For example, our first node of 'Patrons' from the restaurant data set would have an attribute list of [Some, Full, None] as the possible branches from itself.

The implementation first started with reading in the .txt and .csv files correctly. We wanted to read each data set into a 2D array that we could manipulate as we work with it. Next, we focused on the implementation of the importance function and the functions within that to calculate which attribute from the data set is the most relevant to branch off of. Our importance function was split into three separate functions to help it calculate the importance value of a certain attribute. First, we had importance(labels, and rowList), where labels are the attribute labels from row 0 in the data set, and rowList is the rest of the data set in 2D array form. importance() then splits rowList into columns for entropy to parse for our importance value. For entropy(responses, column, labels), responses are the list of responses(either "Will Wait" or

"Response") for a certain attribute, column is one column from the importance() function, and labels are the same as before. For b(q), we calculate the entropy given a q-value, using the definition from the slides. After running the functions, we should have the most important attribute returned as maxNode. Our decision_tree_builder function follows the pseudo-code of the decision tree learning algorithm that we were given in class and calls the importance function in a similar spot, where we start building our tree. After our function stops recursing, we should have an accurate depiction of what our tree should look like by viewing the various importance values and selected attributes. After the tree builds, we print it out from main() using printTree(), which recursively prints our decision tree given the head node.

Following the printing of our initial tree, we are able to get stats regarding the tree, and we can proceed to prune the tree using the chi2 method. We could not get chi2 or printTree() (for the house-votes tree) to work as intended.

Confusion matrices for congressional voting records dataset, both the training and test sets, for your unpruned tree and your tree pruned with a significance level of 5% or 1%.

Restaurant Un-pruned Tree:

	Predicted Negative	Predicted Positive
Actual Negative	TBD	TBD
Actual Positive	TBD	TBD

Restaurant Pruned Tree (α = 0.05):

	Predicted Negative	Predicted Positive
Actual Negative	TBD	TBD
Actual Positive	TBD	TBD

Restaurant Pruned Tree (α = 0.01):

	Predicted Negative	Predicted Positive
Actual Negative	TBD	TBD
Actual Positive	TBD	TBD

House Votes Un-pruned Tree:

	Predicted Negative	Predicted Positive
Actual Negative	TBD	TBD
Actual Positive	TBD	TBD

House Votes Pruned Tree (α = 0.05):

	Predicted Negative	Predicted Positive
Actual Negative	TBD	TBD
Actual Positive	TBD	TBD

House Votes Pruned Tree (α = 0.01):

	Predicted Negative	Predicted Positive
Actual Negative	TBD	TBD
Actual Positive	TBD	TBD

We did not quite understand how to fill out the confusion matrices given that our chi2 function did not work. Assuming that our results were found, we saw in class that the restaurant dataset prunes down to a small tree. And, based on the size of the datasets for the house-votes, we can assume that the tree's size shrinks significantly, considering its initial size. Unfortunately, we did not get to actually see the trees that you wanted us to print. We had trouble implementing the code; for the most part, the concepts and calculations we were able to do by hand, or at least visualize, but we just found it hard coding it the way it should be.

Thanks for a great semester! We learned a lot and it was super interesting to learn more about this field!