1. (0.5 points) Which other student, if any, is in your group? (either names or netIDs is fine)

   Brian Suchy and Tristan Litre

2. (0.5 points) Did you alter the Node data structure? If so, how and why? (2 sentences)

   Yes the Node data structure was changed to be as follows:

   class Node:

   def __init__(self):

   self.label = None – This was originally in the code, but stored the attribute that was split on to reach the node.

   self.children = {} – This dictionary held all the children of a node. The key was the value that caused the traversal to the child.

   self.decision_attribute = "" – This held what attribute to branch on. (Based on Information Gain)

   self.previous_decisions = [] – This holds previous attributes split on by the nodes parents. Helped prevent infinite recursion

   self.examples = [] – This held all applicable examples to the node in the tree. It is a waste of space, but helped during development.

   self.depth = 0 – This held what depth level the node was at. Not used.

   self.decisionMade = "" – This told if the node was the last node; therefore, returned the classification.

   self.value = Value – This holds the nodes guess at a classification. It will hold the Mode of examples as default.

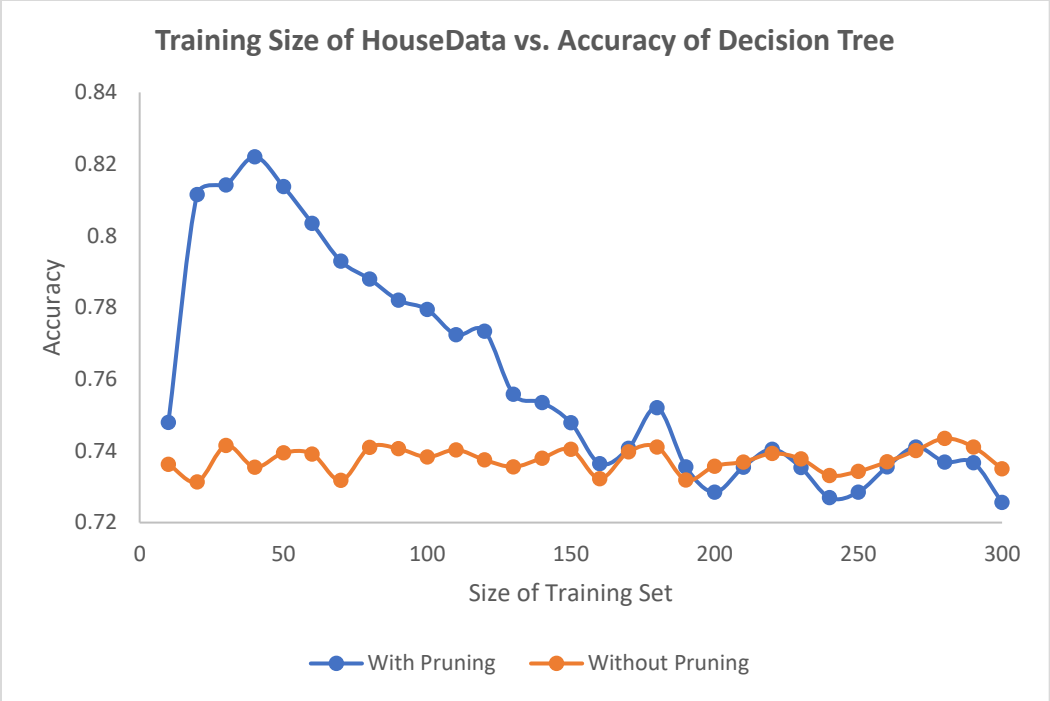
3. (1 point) How did you handle missing attributes, and why did you choose this strategy? (2 sentences)

We handled missing attributes by making them a decision themselves. We chose this because abstaining a response and/or lacking data in a category is actually valuable information itself. It also simplifies the process as opposed to assigning it a value and potentially creating inaccuracy.

4. (1 point) How did you perform pruning, and why did you choose this strategy? (4 sentences)

   The method we used to prune was to attempt to remove children recursively starting with the root. The pruning would see if simply selecting the mode of a node would result in higher accuracy rather than continue navigating the tree to the children. If the accuracy was higher. Then the children were killed (oof) and the self.decisionMade variable was set to "Y" for the node. If the accuracy was not improved or the same, then the pruning would be called on each of the children of the current node. Rinse and repeat.

5. (2 points) Now you will try your learner on the `house_votes_84.data`, and plot learning curves. Specifically, you should experiment under two settings: with pruning, and without pruning. Use training set sizes ranging between 10 and 300 examples. For each training size you choose, perform 100 random runs, for each run testing on all examples not used for training (see `testPruningOnHouseData` from `unit_tests.py` for one example of this). Plot the average accuracy of the 100 runs as one point on a learning curve (x-axis = number of training examples, y-axis = accuracy on test data). Connect the points to show one line representing accuracy *with* pruning, the other *without*. Include your plot in your pdf, and answer two questions:



a. In about a sentence, what is the general trend of both lines as training set size increases, and why does this make sense?

   As the training set size increases, the accuracy of the tree seems to converge to an accuracy around ~75%. This makes sense because the decision tree has become mostly filled out and there isn't much guessing to be made with the test examples since the decision tree is now so rich in information.

b. In about two sentences, how does the advantage of pruning change as the data set size increases? Does this make sense, and why or why not?

   Pruning becomes much more effective as the validation data set size is larger; however, as the validation set size decreases, because the training set starts taking more and more from the validation set, the accuracy of the tree begins to sink back down to the level of not pruning at all. This makes sense because a smaller validation set may not be as representative of the entire set as opposed to a larger validation set.

Note: depending on your particular approach, pruning may not improve accuracy consistently or may decrease it (especially for small data set sizes). You can still receive full credit for this as long as your approach is reasonable and correctly implemented.