1. No. I work on it my own.

2. Yes. “values” as a dictionary type variable keeps track of names of children nodes and “pruneMe” as a boolean variable identifies if the node is a leaf node.

3. I assign those missing attributes with most common values of those data examples that have the same classifications because it is an easy approach to implement.

4. I dump the data into training data set and validation data set, and training data set takes the first two third of the data and validation data set takes the last one third of the data. Output a decision tree by training training data set and test it on validation data set and get a performance value. Prune each leaf node and replace its label with the most common class of the whole training set if it increases the performance.

5. a. Without pruning, we can observe from graph 1 the line gradually goes up and goes down after reach a turning point, due to the “overfitting” issue; With pruning, we can observe from graph 2 the line vibrates at the first, due to the small size of training set, and gradually approaches to a certain value but not reaching it, due to the stabilization of the performance as the training size increases.

   b. The advantage of pruning is, as the size of training data increases, the performance would not go down after a certain point known as “overfitting” issue. It makes sense because noisy or repeat data that would not do any good to our tree building process will be handled by our pruning strategy.