COMP349 HW1

Chen Si
Dong Shu
Ding Zhang

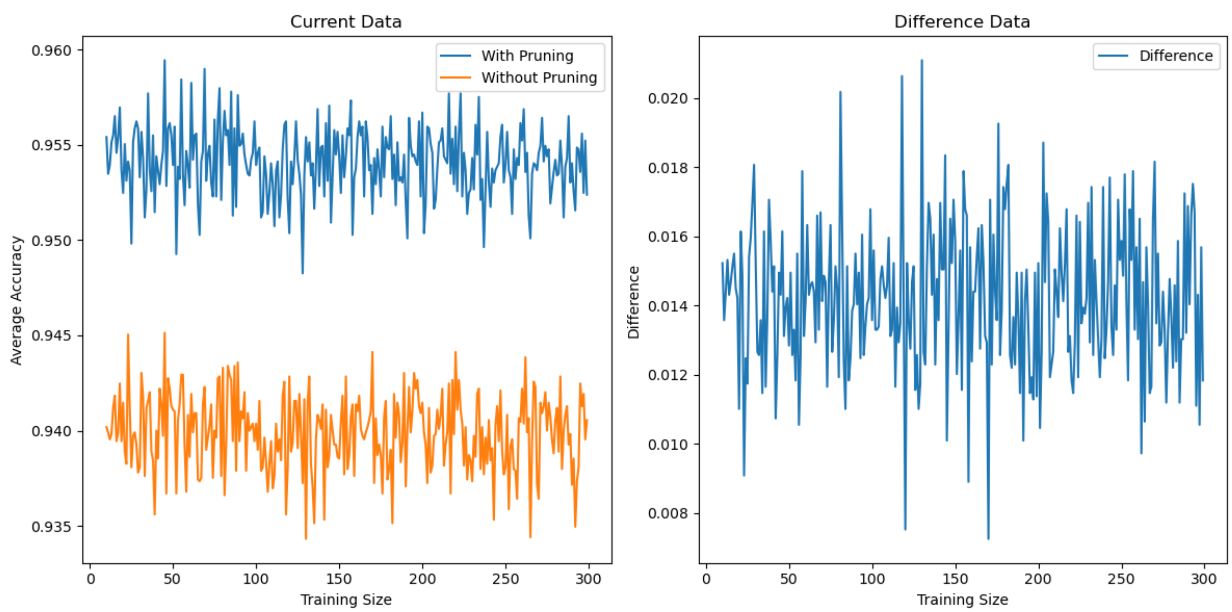
1. (2.0 points) Did you alter the Node data structure? If so, how and why?
   - Yes, we did alter the Node data structure.
   - Besides the fields labels and children, we also add two fields: attributes and entropy for each node. The field "attribute" of each Node inherits the parent's attribute, which is an indication that the current node is splitted using that particular attribute. The field "label" is the unique value of that attribute, which serves as edges connecting the nodes if we would draw out the decision tree on paper. If the node is a leaf node, the attribute field would be None, and the value of the label is the predicted class label. We have also added several functions: change_label, change_attribute, add_child and change_entropy. These methods allow us to change or add values to each of the node values.

2. (2.0 points) How did you handle missing attributes, and why did you choose this strategy?
   - We replace missing attributes with the most common label in the data rows (the mode of that column). In our opinion, if the data set is large enough, missing values are trivial in a sense that they won't affect the overall prediction a lot. Thus, we choose to replace missing values with the mode of the column.

3. (2.0 points) How did you perform pruning, and why did you choose this strategy?
   1. Pre-Pruning. We perform pruning by calculating the absolute value between the parent's entropy and the child's entropy. We prune the node if the entropy difference is greater than some threshold e (in our case we set e = 0.1).
   2. Post-Pruning. We mainly utilize the reduced error pruning technique. For a given tree and validation set, we first perform the ID3 algorithm on the validation set and record the current precision value. We then find the deepest node of the entire tree and retrieve its parent node. We remove the child nodes from the parent node and recalculate the precision value. If the precision value after pruning is greater than before, we preserve the pruned tree and recursively perform pruning on the next deepest node. If not, we do not perform pruning.
      a. Two approaches: start performing pruning from the root to the leaf node(work the way down)
      b. Start from the leaves of the tree and work your way up towards the root.
         i. We perform a pruning function for each node in the tree.
         ii. Temporarily remove the subtree from the validation dataset and calculate the accuracy.

iii. Compare the accuracy with the accuracy of the original tree (i.e before removing the subtree). If the accuracy remains the same or improves, then we prune the subtree; otherwise, we keep it.
iv. Repeat this process for all nodes in the tree.

4. (4.0 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. What is the general trend of both lines as training set size increases, and why does this make sense?

- Based on our generated learning curves shown above, we have noticed that the deviation of the pruning curve tends to decrease significantly as the number of training sizes increases. The learning curve for our non-pruned tree, on the other hand, has an increasing (or at least non-decreasing) deviation. Even though it is not significantly clear in this graph, the tree with pruning should have a trend of increasing accuracy as training size increases, and a trend of decreasing accuracy for non-pruned trees.

- This makes sense because the key idea for pruning is trying to reduce overfitting. There is a much larger probability that the tree will overfit on a large dataset, which makes the algorithm harder to generalize to test sets. Pruning will reduce the impact of overfitting and have higher average precision for larger datasets.

b. How does the advantage of pruning change as the data set size increases? Does this make sense, and why or why not?

- Similar to the idea above, as the data set size increases, the advantage of pruning will become more significant. A decision tree is very susceptible to overfitting, especially when the size of the dataset increases. This makes it harder for the original tree to generalize to data sets other than the training dataset, as it fits the training set too well. Pruning will remove the observed overfitting and can be better generalized to other testsets. This advantage becomes significantly noticeable as dataset size increases.

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.

5. (optional 2.0 points) Use your ID3 code to construct a Random Forest classifier using the candy.data dataset. You can construct any number of random trees using methods of your choosing. Justify your design choices and compare results to a single decision tree constructed using the ID3 algorithm.

- Using a single decision tree, we achieved an average accuracy of 0.7209090909090909 (with pruning) and 0.6927272727272724 (without pruning) after running the model 100 times on the candy dataset. In contrast, when employing a Random Forest classifier with 100 trees, our test accuracy significantly improved to 0.8636363636363636. Note that this number may vary between different runs, but in general, our Random Forest Predictor had significant improvement on accuracy.
- We opted for a Random Forest size of 100 trees because the testing time for the single decision tree was set to 100 runs. This decision ensures a fair comparison between the two models. It is evident that when comparing a single decision tree to a Random Forest classifier, the accuracy of the single decision tree, even with pruning, falls behind that of the Random Forest classifier.