Ailin Chu

Weihao He

Xiao Qin

MSAI 349-Homework #1

17 October 2024

Decision Tree - Homework Report

1. Did you alter the Node data structure? If so, how and why?

Yes, we did modify the Node data structure by adding four member variables: leaf, attribute, classification, and father_certain_class.

Specifically, each new member variable serves the following purpose:

1. leaf: A boolean variable indicating whether the node is a leaf.
2. attribute: A string variable representing the attribute name on which the node branches. If the node is a leaf, the attribute is set to None.
3. classification: A string variable containing the classification result of the node. If the node is not a leaf, classification is set to None.
4. father_certain_class: An array variable storing the classification results of all child nodes of the parent node that are leaves. It is used to narrow down the selection range for replacement with a leaf during pruning.

2. How did you handle missing attributes, and why did you choose this strategy?

We handle missing attributes by assigning them with the most common value (mode).

This is a straightforward and fast strategy, assuming that the majority value is a good representative for missing entries.

3. How did you perform pruning, and why did you choose this strategy?

How we perform reduced error pruning:

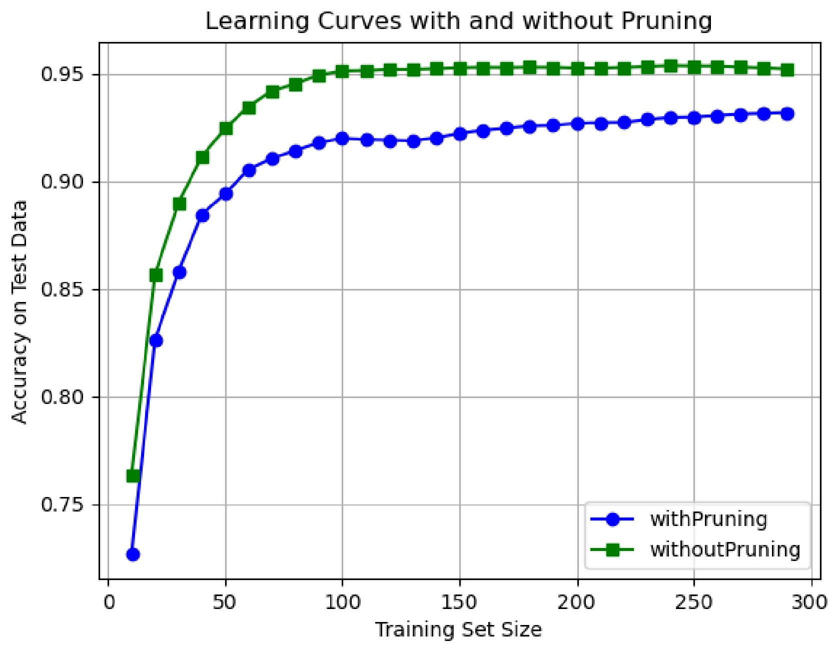
· Recursive pruning removes non-leaf nodes. If removing the leaf nodes does not increase accuracy, the pruned branches are restored.

· Additionally, we noticed that occasionally the pruned model performs significantly better than the original model. We believe this may be due to the imbalance caused by random samples. Therefore, if the accuracy improvement from pruning exceeds a threshold of 0.3, the original tree is restored.

We chose reduced error pruning for the following reasons:

Reduced error pruning focuses on improving the decision tree's performance on an independent validation or test dataset. This aligns with the goal of building models that generalize well to unseen data, avoiding overfitting that can occur if the tree becomes too complex. In contrast to other methods like "critical value pruning," reduced error pruning evaluates the impact of pruning based on performance, rather than relying on arbitrary thresholds for node importance. This ensures that only those splits that contribute meaningfully to performance are kept. And it's faster and more stable than error complexity pruning without need for hyperparameter tuning.

4. 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?

The accuracy of the pruned tree increases as the training set increases, and eventually it converges to around 92%. It is reasonable because when the size of the training set is low, the tree is prone to underfit, and the tree could suffer more when the pruning process cuts down its useful branches and leads to worse performance. As the size of the training set increases, the tree becomes more complex and deeper, performing better on accuracy. At this stage, pruning can be a good technique to prevent overfitting and enhance generalization by cutting off unnecessary branches.

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

As the data set size increases, the advantage of pruning may decrease. Larger data set size could lead to a more complex and deeper tree, and the tree naturally becomes more robust and generalizable because it is trained on a larger and more representative sample of the overall data distribution. Thus, while pruning is more beneficial when training on smaller datasets, its relative advantage diminishes as the dataset size increases because the model inherently starts to generalize better without needing much simplification. This trend makes sense, as pruning is primarily a regularization technique to prevent overfitting, and the risk of overfitting decreases with larger training sets.

5. Use your ID3 code to learn a decision tree on cars_train.data. Report accuracy on cars_train.data, cars_valid.data, and cars_test.data datasets. If your accuracy on cars_train.data is less than 100%, explain how this can happen. Prune the decision tree learned on cars_train.data using cars_valid.data. Run your pruned decision tree on cars_test.data, and explain the resulting accuracy on all three datasets.

Before Pruning: cars_train acc: 100% cars_validation acc: 71.43% cars_test acc: 65.71%

After Pruning: cars_test acc: 62.86%

Before pruning, the model overfits the training data, achieving 100% accuracy on train data but struggling to generalize to the validation and test sets for its low accuarcy.

After pruning, the model sacrifices some training accuracy which decreased to 70% to avoid overfitting. It does not memorize the training data as before, and this symbolizes better generalization, although the test accuracy reduces a little.

The fact that the test accuracy does not improve significantly after pruning could be due to other factors, such as the efficiency of our pruning technique, or the limitation of the dataset size, limiting the model's ability to further improve performance.

6. 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.

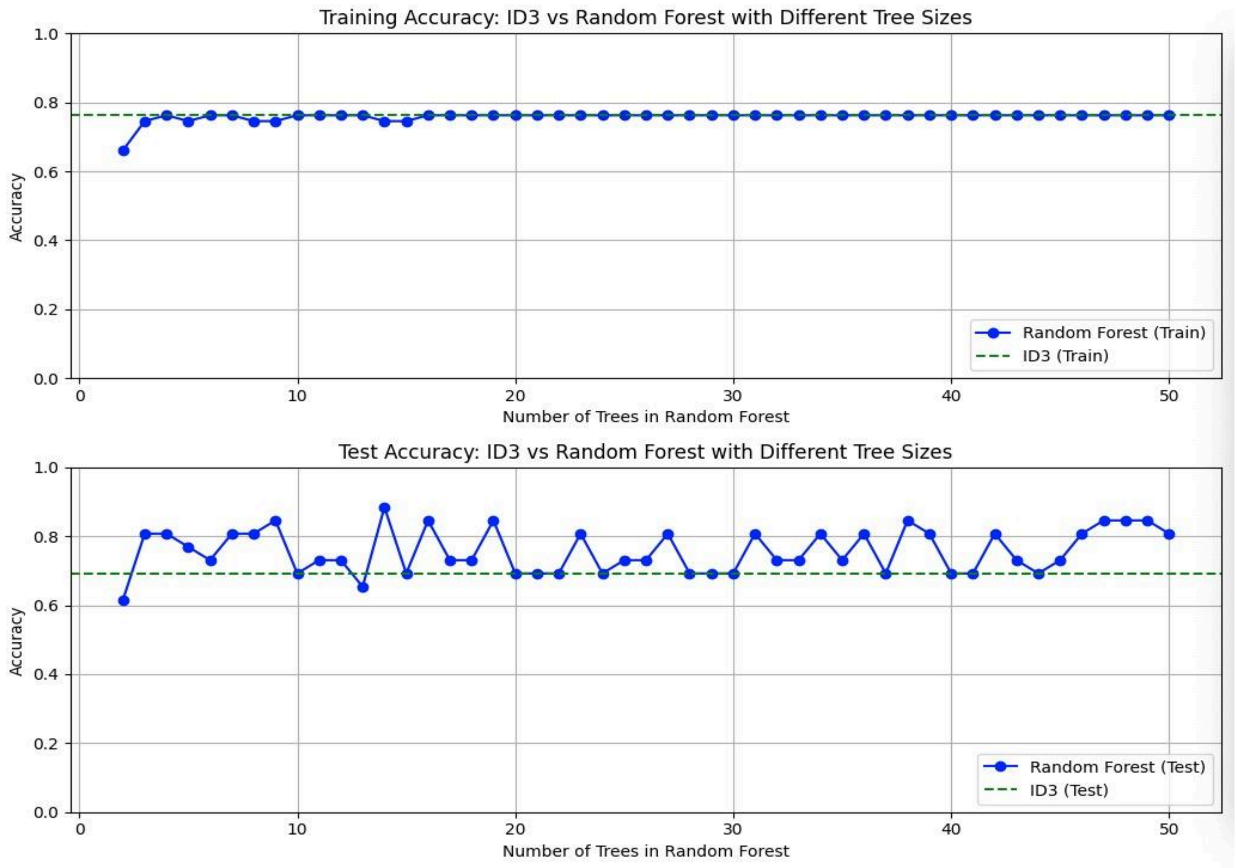
In this section, we constructed a Random Forest classifier based on three techniques besides ID3 decision tree: **Bootstrap Aggregating, Random Feature Selection, Voting:**

(1) **Bootstrap Aggregating** is used to create different training subsets for each tree. Each subset is created by randomly sampling from the original dataset with replacement. This would ensure that each decision tree is trained on different data, increasing diversity among all the trees in the Random Forest. And this technology prevents overfitting to some extent.

(2) **Random Feature Selection** is used to introduce more diversity in the process of making decisions. Specifically, instead of considering all features at each node of a decision tree, only a randomly selected subset of features was evaluated for the best split selections. This reduces the correlation between the trees and allows for a wider exploration of the feature space. It also helps prevent the model from over-relying on any single feature.

(3) **Voting** is used to aggregate the predictions from all decision trees. For classification task (like the prediction task on the candy.data), the final prediction is made based on majority voting, where the class receiving the most votes across the trees is selected to be the final prediction.

We conducted experiments on the **candy.data** utilizing the framework mentioned above to construct Random Forest classifier. And the results of the Random Forest with 2 to 50 trees were compared with the results of a single ID3 tree on both the training set and test set. In addition, the training and test sets were split in a 7:3 ratio. A comparison chart was plotted based on the experiment results.



According to the comparison chart, we can observe:

(1) In the Random Forest, the model's generalization ability on the training set is slightly lower that that of a single ID3 tree, indicating that to some extent, Random Forest can reduce the model's overfitting.

(2) Additionally, when comparing the results on the test set, it is obvious that Random Forest performs significantly better than the single ID3 tree. This further demonstrates that Random Forest can prevent the model from overfitting to the training data and improve its generalization ability.

However, we observed that the experiment results were unstable and Random Forest does not necessarily perform better than a single ID3 tree. We believe the reason lies in the fact that the candy dataset is too small, with very few feature dimensions. In this case, the effectiveness of building a Random Forest might be reduced, as the techniques employed by Random Forest are more suited to large datasets and high-dimensional feature spaces.