Project 3 - Assess Learners (Report)

Vishnu Kamaraju
vkamaraju6@gatech.edu

***Abstract*—This study primarily investigates two learning algorithms, a classical Decision Tree Learner (DTLearner) and a Random Tree Learner (RTLearner), and how the leaf size affect their performance. Root Mean Squared Error (RMSE) and correlation between predicted and actual data were used to evaluate model accuracy. We also examine the effect of Bagging (Bootstrap Aggregating) on overfitting. Additionally, it compares the training time, and memory usage of DTLearner and RTLearner under varying leaf sizes, using the Istanbul dataset. The findings suggest an optimal leaf size for minimizing overfitting and improved generalizing through bagging.**

1 INTRODUCTION

This report explores the impact of leaf size on model performance and investigates the benefits of Bagging to reduce overfitting. Decision Tree Learners (DTLearners) are prone to overfitting when the model fits too closely to the training data. Bagging has been shown to address this issue by averaging predictions across multiple learners. We also compare the computational efficiency (in terms of time and memory) of DTLearner and RTLearner models. The hypothesis is that bagging will improve generalization and that RTLearner will offer faster training times due to random feature selection.

2 METHODS

The experiments were conducted using the Istanbul dataset. For Experiment 1, RMSE values were evaluated in both in-sample and out-of-sample predictions across varying leaf sizes to identify overfitting and underfitting. In Experiment 2, Bagging was applied to DTLearner, and its effect on overfitting was measured. Finally, Experiment 3 compared the time taken to train and memory usage of

DTLearner and RTLearner at different leaf sizes. The key metrics for the models were RMSE, time, and memory usage.

3 EXPERIMENTS

3.1 EXPERIMENT 1

The experiment discusses the Root Mean Squared Error (RMSE) for both in-sample and out-of-sample predictions for the Istanbul.csv dataset as we vary the *leaf size* of the *Decision Tree Learner (DTLearner)* model. This helps to identify the overfitting and underfitting in the model.

Figure-1 shows how the RMSE value changes as we vary the leaf size for the DTLearner for both in-sample and out-of-sample data.

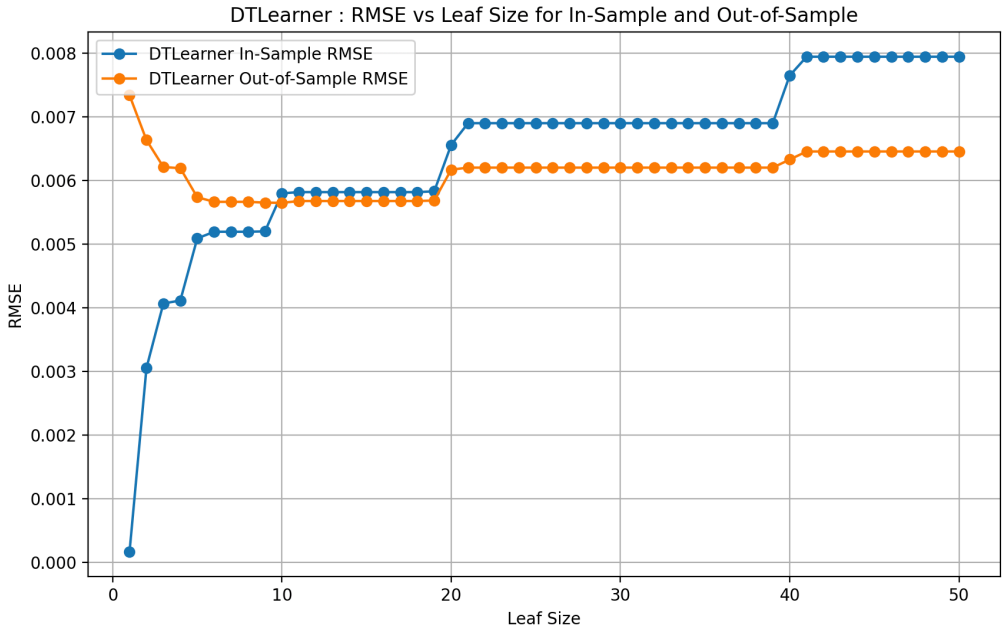

Figure 1— Captures the change in RMSE value as we vary the Leaf Size.

For **small leaf size (1 – 10)**, the in-sample RMSE is very low, which indicates that the model fits the training data very well. However, the out-of-sample RMSE is

quite high in comparison which indicates **overfitting.** This means that the model generalizes poorly to the unseen data.

For **medium leaf size (10 – 20),** the gap between in-sample and out-of-sample RMSE beings to **converge**. This indicates a better balance between fitting the training data and generalizing to the test set.

For **large leaf sizes (25+),** the in-sample RMSE **gradually increases**, reflecting that the model is becoming too simple and no longer capturing enough detail from the training data (underfitting). The out-of-sample RMSE remains relatively stable but stops improving. This suggests the model is overly biased and unable to capture more complex patterns.

In conclusion, the leaf size between **10 to 20** seems to the optimal region for this model as it strikes a good balance between underfitting and overfitting. Furthermore, the model may benefit by implementing multiple decision trees on different bootstrapped samples of data, and adding randomness to the feature selection – as this combination reduces variance and improves generalization.

3.2 EXPERIMENT 2

In this experiment we discuss how **Bootstrap Aggregating (Bagging)** affects overfitting when run on the Decision Tree Learner (DTLearner) model with varying leaf sizes on the *Istanbul.csv* dataset.

Overfitting is evident when the **in-sample RMSE** is much lower than the **out-of-sample RMSE.** In *Figure-2,* overfitting is most noticeable for **leaf sizes between 1 and 10**. The in-sample RMSE is very low in this range, but the out-of-sample RMSE is relatively higher, signaling overfitting

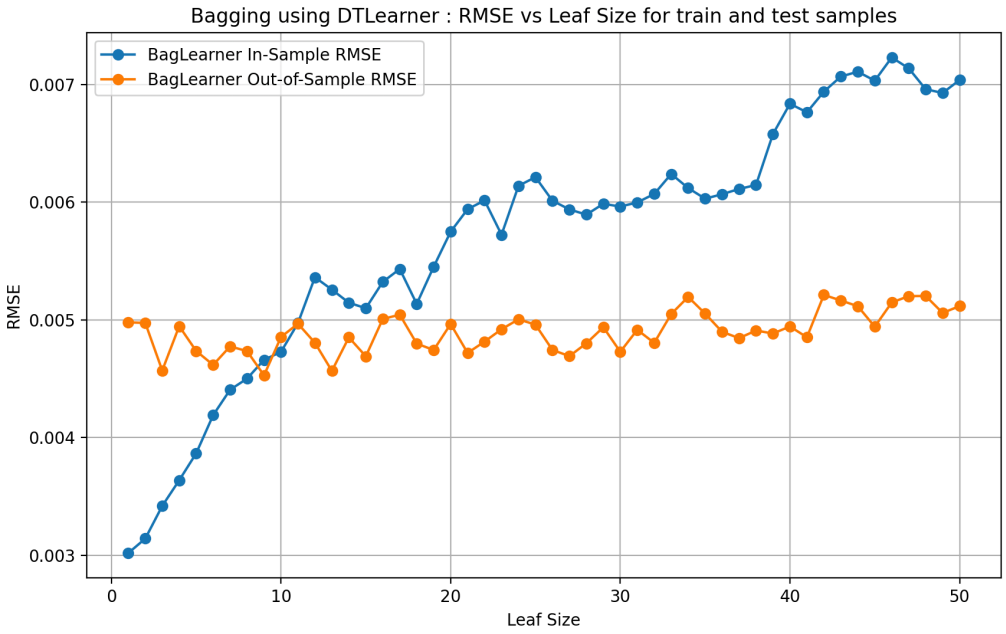

Figure 2 — Bagging approach demonstrates how it reduces overfitting

It is also evident from *Figure-2* that **bagging can reduce overfitting.** The **In-sample** RMSE increases with leaf size. As leaf size increases, we deal with fewer decision nodes that result in more generic splits. This makes the model simpler as the decision tree becomes less complex. However, **out-of-sample RMSE** stays stable, showing that **bagging generalizes the model well to the unseen data**, thus reducing overfitting.

It is also evident from *Figure-2* that while bagging can reduce overfitting, it **cannot eliminate it.** In this experiment, overfitting is visible for smaller leaf sizes (below 10 -15) where the training RMSE is low, but the testing RMSE is much higher.

In conclusion, bagging helps **smooth out overfitting by averaging predictions**, leading to better performance on unseen data. Although, bagging reduces overfitting, it cannot eliminate it.

3.3 EXPERIMENT 3

In this experiment, we use 2 quantitative metrics **time to train the model** and **memory used** to compare the classic decision tree learner (DTLearner) and the random forest learner (RTLearner). Both the learners are given as inputs varying

leaf sizes, and their quantitative metrics are recorded which is presented as evidence.

TIME TAKEN TO TRAIN THE MODEL

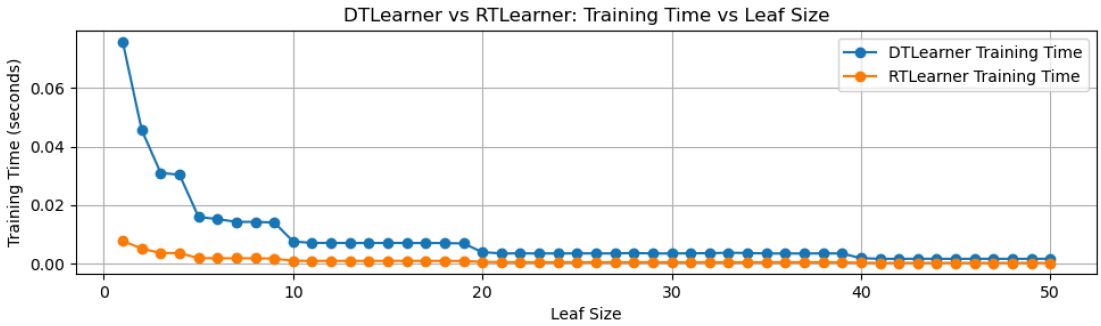

Figure 3 — Time taken to train DTLearner and RTLearner with varying leaf sizes

From *Figure 3,* it is evident that when training with smaller leaf sizes, the Decision Tree Learner **(DTLearner)** takes more time compared to the Random Tree Learner **(RTLearner).** This difference in time is attributed to:

- **Optimal split:** DTLearner needs to search for the best feature and threshold for splitting at each node. This process involves comparing multiple features and calculating the information gain, which can be computationally expensive.
- **Random features and splitting in RTLearner:** On the other hand, RTLearner randomly selects features to split. This variation results in skipping the optimization process – which involves finding the best feature based on the highest correlation between the features and the output value and finding its median value – that DTLearner algorithm utilizes.

As the leaf size increases, the training times for both DTLearner and RTLearner **converge.** This behavior is due to:

- **Simplified Trees:** Larger leaf sizes result in less complex trees for both the learners, leading to reduction in computational time. Fewer nodes created make the tree building process faster for both the learners.

- **Random and Deterministic splitting becomes negligible:** At smaller leaf sizes, DTLearner spends more time finding the optimal split, while RTLearner selects splits randomly. However, with larger leaf sizes, the impact of optimal vs. random splitting is reduced since fewer splits are needed, leading to similar processing times.

MEMORY USAGE

Both DTLearner and RTLearner stores similar tree structures in memory which consists of nodes, leaf nodes, and references to data points. This leads to similar memory usage by both trees in most cases. This behavior is further evident in *Figure-4* which plots the memory used by both the learners with varying leaf sizes.

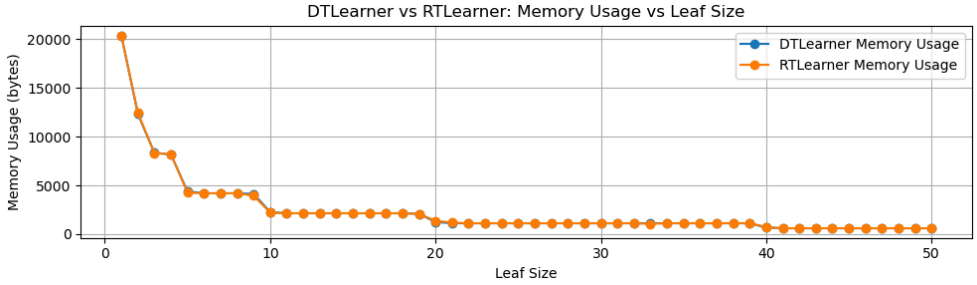

Figure 4— Memory used to store DTLearner and RTLearner tree structures

Since both DTLearner and RTLearner are building trees, they use memory similarly to store the tree, regardless of the time taken for training. Furthermore, as the leaf size increases, the depth of the tree reduces, which results in fewer nodes for both the trees, leading to usage of smaller memory space for both learners as seen in *Figure4*.

4 CONCLUSION

The experiments reveal that a leaf size between 10 and 20 strikes an optimal balance between overfitting and underfitting for DTLearner. Bagging reduces overfitting, though it cannot fully eliminate it. The comparison between DTLearner and RTLearner indicates that while RTLearner generally trains faster at small leaf sizes due to random feature selection, both models converge in

training time and memory usage as leaf size increases. These results highlight the trade-offs between model complexity and computational efficiency.