Project 3: Assess Learners

Julien HUANG
jhuang793@gatech.edu

***Abstract*—This third project analyzes overfitting in decision trees, the impact of bagging on overfitting, and compares DTLearner and RTLearner. Results show that bagging reduces overfitting, DTLearner offers stability, and RTLearner is faster but less consistent.**

1 INTRODUCTION

Decision trees are widely used in machine learning for regression and classification. Despite their interpretability and efficiency, they often suffer from overfitting, leading to poor generalization. This project examines overfitting in decision trees, the impact of bagging on mitigating this issue, and compare DTLearner and RTLearner under different conditions.

2 METHODS

A decision algorithm learns from a dataset, by using the best feature that has the highest correlation with the target variable, we split the training data into smaller groups.

We designed three experiments to analyze overfitting, the effect of bagging, and the performance of DTLearner vs RTLearner. The `Istanbul.csv` dataset was **shuffled** and split into 60% training and 40% testing. The dataset split was performed once and remained the same across experiments for consistency.

2.1 Experiment 1: Overfitting in DTLearner

We tested how overfitting affects DTLearner by measuring RMSE for training (in-sample) and testing (out-sample) data while varying the `leaf size` from 50 to 0.

2.2 Experiment 2: Impact of Bagging

This experiment repeats the first but applies **Bagging** using BagLearner with 20 bags. By training multiple models on different bootstrap samples of the dataset and taking the mean of their predictions. Bootstrap sampling involves randomly selecting training samples, ensuring diversity in the training process. The goal

is to see if this stabilizes RMSE and improves generalization.

2.3 Experiment 3: Comparing DTLearner and RTLearner

We compared them using three metrics:

- Maximum Error (ME)
- Mean Absolute Percentage Error (MAPE)
- Time

For each leaf size (50 to 0), we ran 20 trials and recorded the average, minimum, and maximum values to observe overall performance and variability.

We tested them in two scenarios:

- **Fixed training data:** Using the same dataset across all trials to assess raw performance for each leaf size.
- **Bootstrap sampling:** Randomly resampling the training data to see how well the models generalize in different training conditions.

3 EXPERIMENT 1: OVERFITTING ANALYSIS

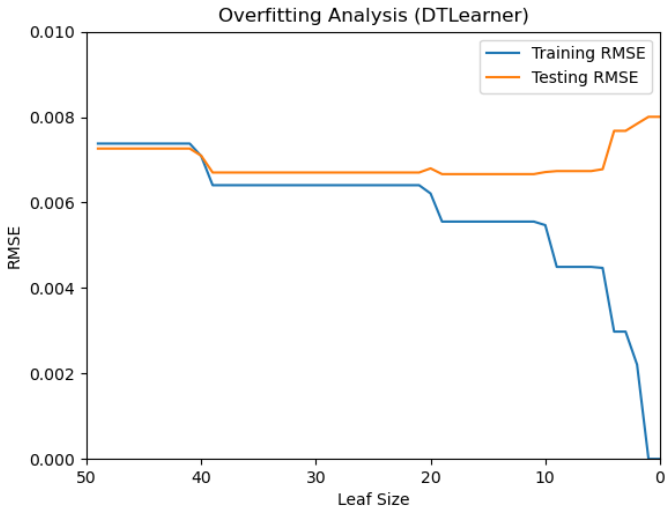


Figure 1—Overfitting Analysis (DTLearner)

3.1 Observations

Overfitting occurs when a model learns the training data too specifically, reducing its ability to generalize.

Figure 1 shows that:

- Below **leaf size = 5**, training RMSE **drops significantly** while testing RMSE increases, marking the onset of overfitting.
- Overfitting intensifies as leaf size approaches 0, with a growing gap between training and testing RMSE.

3.2 Mitigation

To reduce overfitting:

- Use an optimal **leaf size 5**. This is observed when the RMSE decreases for in-sample data but it increases for out-sample data.
- Explore ensemble methods like bagging (discussed in Experiment 2).

4 EXPERIMENT 2: EFFECT OF BAGGING ON OVERFITTING

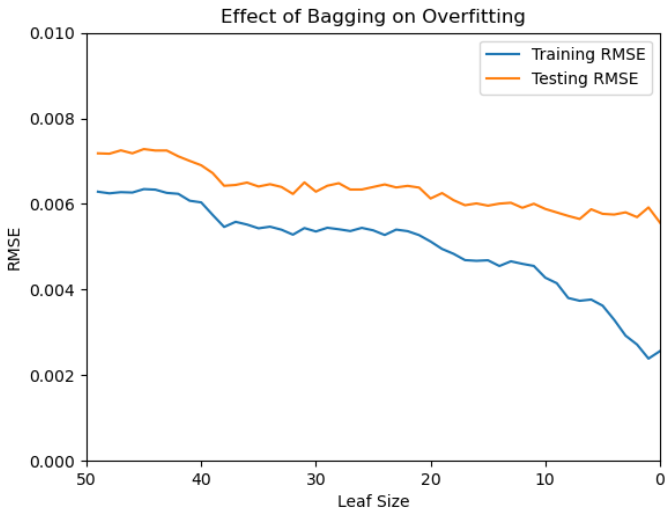


Figure 2—Effect of Bagging on Overfitting

This experiment tests if bagging mitigates overfitting using DTLearner with BagLearner (20 bags) and varying `leaf size` from 50 to 0, with RMSE as the metric.

4.1 Observations

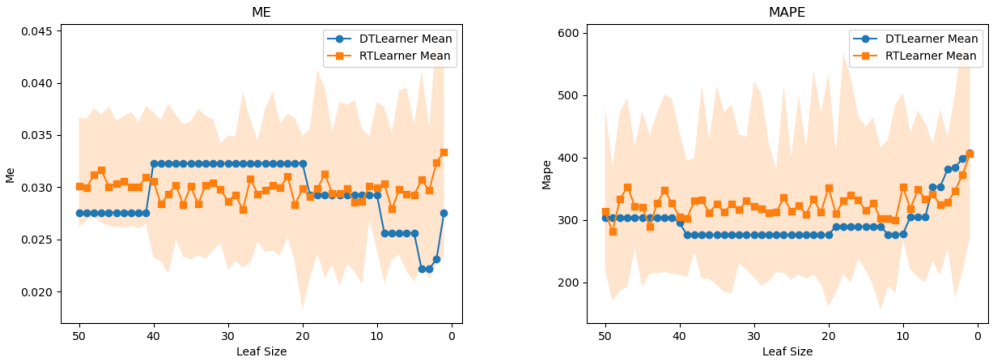
Figure 2 shows:

- RMSE curves are more stable, reducing variance.
- Overfitting starts at **leaf size = 12**, but the gap between training and testing RMSE is smaller than in Experiment 1.
- Testing RMSE stabilizes at low leaf sizes instead of increasing sharply, showing that bagging controls but does not eliminate overfitting.

4.2 Conclusion

Bagging reduces overfitting by preventing the model from learning noise but does not fully eliminate it.

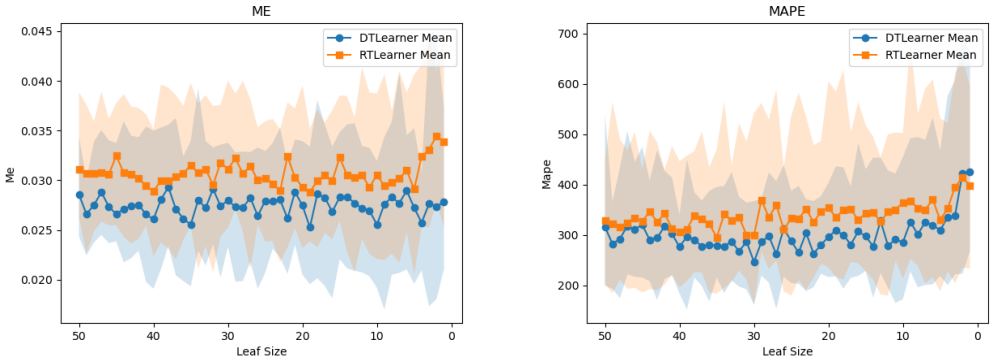
5 EXPERIMENT 3: DTLEARNER VS RTLEARNER



(a) Without Bootstrap sampling ME

(b) Without Bootstrap sampling MAPE

Figure 3—Comparison of ME and MAPE Without Bootstrap sampling(20 runs for each leaf size)



(a) With Bootstrap sampling ME

(b) With Bootstrap sampling MAPE

Figure 4—Comparison of ME and MAPE With Bootstrap sampling (20 runs for each leaf size)

This experiment quantitatively compares DTLearner and RTLearner using Maximum Error (ME), Mean Absolute Percentage Error (MAPE), and Training Time. The goal is to evaluate their performance under different conditions and determine which learner is better suited for generalization.

Why ME and MAPE ?

- **ME :** ME measures worst-case prediction error, highlighting model stability in extreme cases.
- **MAPE :** provides a relative error measure, making it useful for datasets with varying scales and assessing overall accuracy.

We tested both learners across 50 different tree configurations, from the simplest to the most complex, by varying the leaf size parameter.

5.1 Observations

5.1.1 *Fixed training data:*

Using the same dataset across all trials to assess raw performance for each leaf size.

- **ME:** DTLearner showed better stability compared to RTLearner, even though its mean ME was sometimes higher (e.g., between leaf sizes 40 to 20). This suggests that DTLearner produces more consistent predictions, whereas RTLearner, due to its random structure, can have higher worst-case errors in certain cases.

- **MAPE:** Similarly, DTLearner demonstrated greater stability, as expected, since the training set remained constant. While RTLearner occasionally outperformed DTLearner, its performance varied significantly between runs.

Since these results are highly dependent on a single dataset configuration (Istanbul.csv), they do not reflect real-world performance where training data distribution varies. To address this, we introduced bootstrap sampling to simulate a more general case.

5.1.2 *Bootstrap sampling:*

Randomly resampling the training data to see how well the models generalize in different training conditions.

- **ME and MAPE:** DTLearner remained slightly more performant overall, as seen from its lower mean values and smaller error intervals. The variance in RTLearner's error was still larger, reinforcing the idea that it is less stable but sometimes performs better in specific cases.

Training Time: DTLearner slows down significantly as leaf size decreases. RTLearner remains computationally efficient regardless of complexity.

Conclusion: DTLearner is generally more accurate and stable, while RTLearner is faster but more variable.

5.2 Analysis and Discussion

- **Which method is better?** DTLearner is generally more stable and reliable across different datasets, making it a better choice when consistency is required. However, RTLearner is faster and sometimes achieves lower errors, making it useful when training efficiency is a priority.
- **Why does one perform better?**
  DTLearner benefits from a more structured tree-building process, leading to lower variance in predictions. RTLearner, due to its random splits, can be more adaptive but also more inconsistent.
- **Is one method always superior?** No, because the best choice depends on the use case. DTLearner is preferable when stability and accuracy are critical. RTLearner is better when computational efficiency is more important.

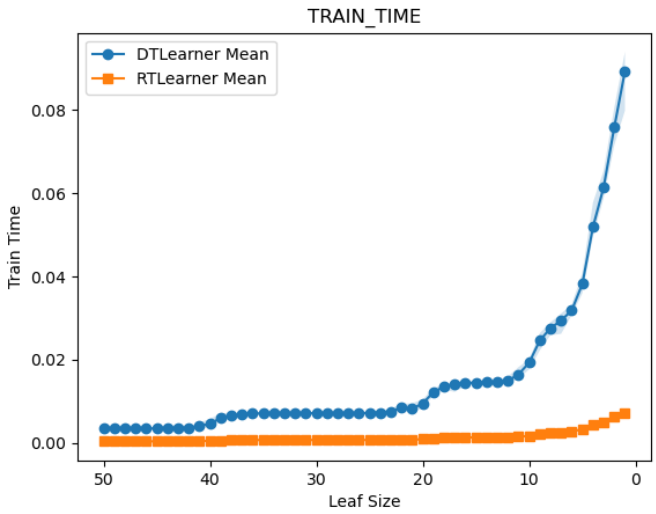


Figure 5—Time (20 runs for each leafsize)

Summary:

- Overfitting occurs when leaf size is too low (<5), reducing generalization.
- Bagging mitigates overfitting, but does not completely eliminate it.
- DTLearner is more stable and accurate. RTLearner is faster but more variable. Bootstrap sampling confirms DT's reliability. RT is preferable for large datasets where speed matters.