Project3: Assess Learners

Yicun Deng
ydeng335@gatech.edu

***Abstract*—This report explores the phenomenon of overfitting in Decision Tree Learning (DTLearner) and evaluates the efficacy of bootstrap aggregation (bagging) in mitigating this issue. It also provides a comparative analysis between classic Decision Trees and Random Trees (RTLearner) using novel quantitative measures. Experimental results provide insights into the performance of these methods and their applications in various scenarios.**

1 INTRODUCTION

Overfitting is a pervasive issue in machine learning where a model learns the training data too well, compromising its ability to generalize to unseen data. This investigation primarily focuses on understanding and mitigating overfitting in Decision Tree Learning (DTLearner)(Quinlan, 1986). Bootstrap aggregation, or bagging, is hypothesized to be a potential remedy to overfitting. Furthermore, a comparative analysis between DTLearner and Random Tree Learning (RTLearner) is conducted, offering insights into their relative strengths and weaknesses.

1.1 Method

The experiments were designed and performed in a Linux-based environment using Windows Subsystem for Linux (WSL2), with the machine learning environment of the ML4T software package. Python 3.6.1 was the main language for coding, and the packages utilized for data handling and calculations included NumPy and Pandas. For visualization purposes, Matplotlib was employed. The four learner classes implemented in the experiments were DTLearner, RTLearner, BagLearner, and InsaneLearner. Below is a brief description of each of these classes:

DTLearner: This is a decision tree learner that uses a divide and conquer strategy to recursively split the data based on the feature that reduces the standard deviation of the dependent variable the most. If the size of the remaining data is less than or equal to the given leaf size or if there is no feature that could split the data, the node becomes a leaf, and the mean of the dependent variable in the remaining data is assigned as its prediction.

RTLearner: The random tree learner is similar to the decision tree learner. However, instead of splitting the data based on the feature that reduces the standard deviation of the dependent variable the most, it randomly selects a feature to split the data.

BagLearner: This class uses bagging (Bootstrap AGGregatING) to reduce the variance of a base learner (which can be either DTLearner or RTLearner). The class trains a specified number of base learners, each on a different bootstrap sample of the data, and averages their predictions to make the final prediction.

InsaneLearner: This class is a combination of 20 bag learners, each with 20 DTLearners.

All of these classes follow a similar API. They each have an addEvidence method that takes in training data (Xtrain, Ytrain) and builds the model, and a query method that takes in test data (Xtest) and returns predicted values (Y).

The testlearner.py script is used to test the performance of the learners. It trains the learners on a part of the Istanbul.csv dataset, queries the learners on another part of the dataset, and calculates several performance metrics such as RMSE, correlation, and training time. The script generates plots of these metrics against leaf size for DTLearner and RTLearner, and against bag size for BagLearner. These plots provide visual insights into the overfitting behaviour of the learners and the effect of bagging on overfitting..

2 DISCUSSION

For this part of the paper, I aim to present the similarity and differences between different learner classes with different implementations. Particularly, three experiments were conducted in order to compare the algorithms closely.

2.1 Experiment1-Leave Size and DTLearner

In the first experiment, we studied the effect of different leaf sizes on the performance of the Decision Tree Learner (DTLearner). We used RMSE (Root Mean Square Error) and coefficient as a metric to evaluate overfitting. Showing in figure1, the DTLearner's training error and testing error is visualized in order for intuitively report.

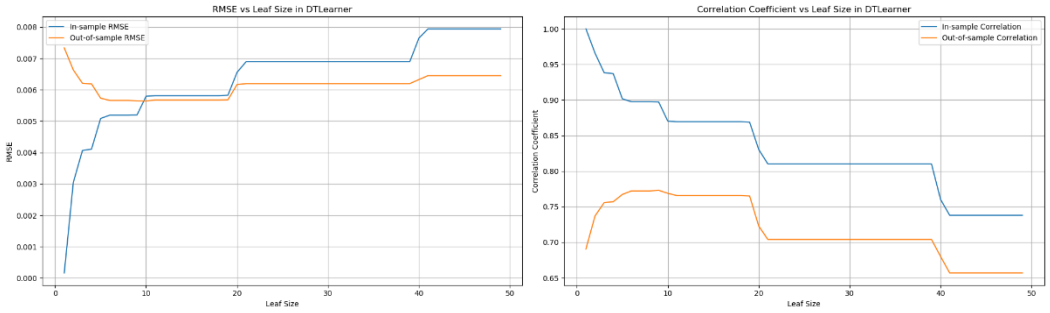

Figure 1 — RMSE and Coefficient of DTLearners regarding the leave size

From the chart displaying RMSE and coefficient versus leaf size for in-sample and out-of-sample data, we observe that as the leaf size increases, the RMSE for both in-sample and out-of-sample data initially decreases, reaching a minimum point, and then starts to increase. This indicates that there is a sweet spot for leaf size (8 to 10) where the model generalizes well to unseen data.

On the other hand, the smaller the leaf size, the more complex the model becomes, fitting the training data closely but failing to generalize well to unseen data. This phenomenon is known as overfitting. Overfitting appears to start when the leaf size is too small (around 1), where the RMSE for out-of-sample data begins to increase while the in-sample RMSE continues to decrease. This suggests that the model is starting to capture the noise in the training data, which harms its generalization ability.

2.2 Experiment2: Baglearner, to be or not to be

In the second experiment, we applied bagging to the DTLearner and studied its effect on overfitting. We kept the number of bags fixed and varied the leaf size

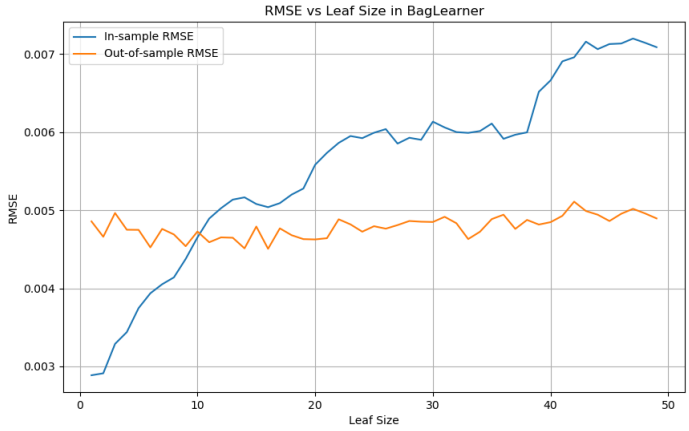

***Figure 2**- RMSE of Bagged DTLearner regarding leave size*

(Figure2). We used RMSE again as matrice to show bagged DTLearner's performance in accordance with leave sizes.

The results(Figure2) showed that bagging was effective in reducing overfitting (DTLearner is no longer overfitting as badly as in experiment 1 with small leave size). As seen from the RMSE vs leaf size plot for the bagged learner, the out-of-sample RMSE was generally lower than that of the non-bagged learner across different leaf sizes after size of 10 leaves. Moreover, the out-of-sample RMSE of the bagged learner did not increase as sharply as that of the non-bagged learner when the leaf size was small, suggesting that bagging helped prevent overfitting to some extent.

However, bagging could not completely eliminate overfitting. When the leaf size was too small, the bagged learner still suffered from overfitting, as indicated by the rising out-of-sample RMSE. This suggests that while bagging can make a model more robust against overfitting, it is not a silver bullet, and other regularization techniques might be needed in conjunction with bagging when the model is overly complex. In fact, while the performance is generally better than unbagged DTLearner, we discovered that the bagged DTLearner favors low leave size even more than the unbagged DTLearner.

2.3 Experiment3: RTLearner and DTLearner

In the third experiment, we compared the DTLearner and RTLearner using tree size, Mean Absolute Error (MAE), R-Squared (R2), and training time as metrics.

Our results (Figure3) revealed that both learners had their strengths and weaknesses. The RTLearner generally had almost identical tree sizes and faster training times than the DTLearner for the same leaf size, indicating that it was more efficient. However, the DTLearner had lower MAE and higher R2 scores, indicating that it was more accurate and explained the variance in the target variable better.

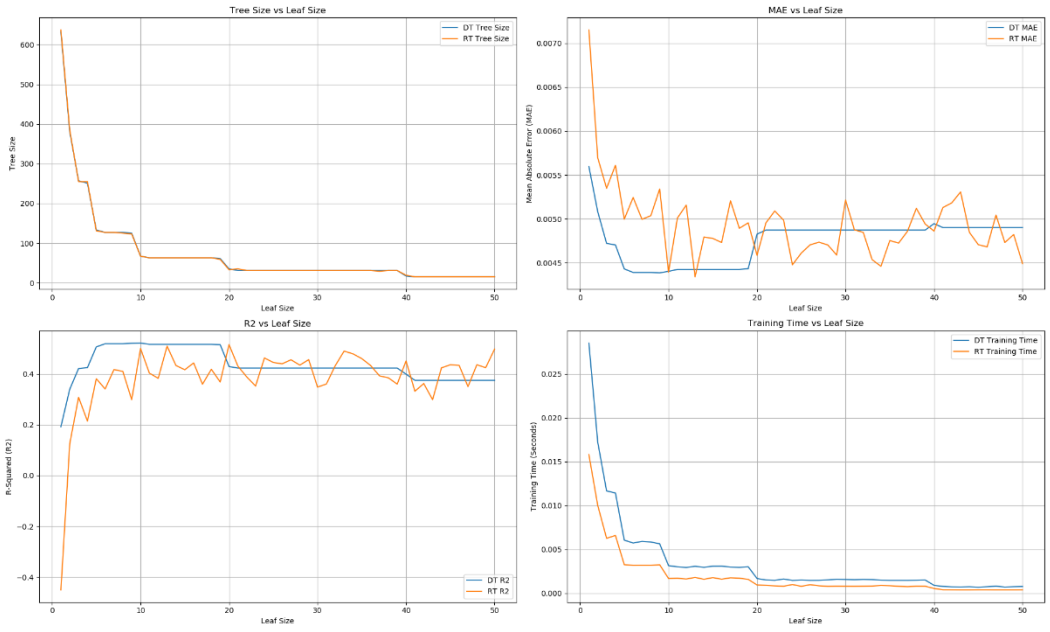

Figure 3 *- Comparision of MAE, R2, Tree Size, and Training Time between DTLearner and RTLearner*

While the DTLearner's performance was competitive with the RTLearner's, it is not always superior. The choice between these two learners depends on the specific problem and the trade-off between accuracy and efficiency. In some cases, it might be worthwhile to sacrifice some accuracy for faster training times and smaller model sizes, especially when dealing with large datasets or when computational resources are limited.

In conclusion, this experiment showed that different learners have different strengths and that there is no one-size-fits-all solution in machine learning. The choice of the learner should be tailored to the specific needs of the task at hand.

2.3.1 *Summary*

There is always trade-off between the algorithm's selection. Beside the embedded experiment, our several testing trials shows that both DTLearner and

RTLearner are performing generally worse than the Linear Regression algorithm provided by the instructor. While DTLearner seems to be a little bit more computationally expensive than RTLearner with slightly better performance. They both got beaten by the linear regression model. In addition, while that bagging of three types of learner provided better performance, bagged linear regression outperformed bagged DTLearner and RTLearner. While bagging is a good strategy for improving model performance, whether to use techniques like bagging depends on the actual dataset you are dealing with. Although techniques like bagging and boosting and emsemble helps with Machine Learning algorithms' performance, the base model typically determines your final outcomes' upper limit and lower limit. For example, bagged DTLearner might performing slightly worse than Linear Regression model without bagging. Thank you!

3 PROCEDURAL ELEMENTS

4 REFERENCES

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106. https://doi.org/10.1007/BF00116251

5 APPENDICES