Project Assess Learners Report : For CS7646 ML4T, Fall 2023

Aditya Lata
alata6@gatech.edu

***Abstract*—** This project report delves into the implementation and evaluation of four CART regression algorithms: Decision Tree learner (DTLearner), Random Tree learner (RTLearner), Bagging learner (BagLearner), and Insane Learner. Working mainly with the Istanbul data file, the overall objective of the project is to predict the return for the MSCI Emerging Markets (EM) index based on the other index returns.

1 INTRODUCTION

Classification and Regression Trees (CARTs) are a family of machine learning algorithms that can be used for both classification and regression tasks. CARTs work by recursively partitioning the data into smaller and smaller subsets until each subset contains only data points of the same class or with similar regression values.

Given the variability in behavior and performance of CARTs, this project conducts three experiments to assess these algorithms. The report outlines the results of these experiments, focusing on the implementation and evaluation of these four CART regression algorithms in Python.

2 METHODS

The assess_learners folder has five CART learners as regression learners: LinRegLearner(Linear Regression), DTLearner(Decision Tree), RTLearner(Random Tree), BagLearner(Bag Learner – an ensemble of multiple learners), and InsaneLearner(ensemble of multiple Decision Trees). Each of these learners adheres to an API specification[1] for training and then querying to estimate a set of test points given the model we built.

The training and testing data can be used from the Data folder, current implementation does not consider date-time columns, and randomly selects a 3:2 split for training and testing respectively. Each data file contains N+1 columns: X1,

X2, … XN, (collectively called the features), and Y (referred to as the target). testlearner.py contains the code necessary to run all the experiments. The hyperparameter varied in the experiments is leaf size. We use the numpy coercoeff[5], and argsort[6] functions to evaluate the Pearson coefficient of correlation, and also as a helper to evaluate R-squared, and to perform an indirect sort.

3 DISCUSSION

3.1 Decision Trees Learner - DTLearner

In DTLearner, we build a tree recursively, where at each step we identify the "best feature to split on" as the feature (Xi) that has the highest absolute value correlation with Y. When the tree is constructed recursively, if there are leaf_size or fewer elements at the time of the recursive call, the data is aggregated into a leaf. Thus, leaf size is a critical hyperparameter deciding the overall behavior and precision of the DTLearner model. To study the role of leaf size on the DTLearner, we varied leaf size and measured Root Mean Square Error (RMSE) [3] between the predicted and actual values of the target variable, and the observations can be seen in Figure 1.

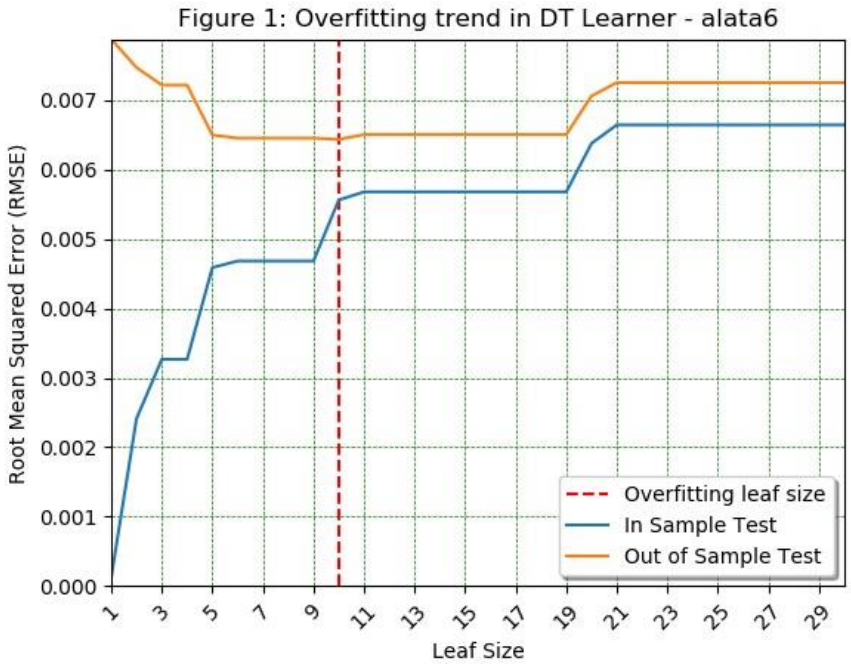

Figure 1 — RMSE vs Leaf Size observations for In Sample and Out of Sample testing of a Decision Tree Learner.

We observe that as leaf size decreases, the in-sample test RMSE continues a downward trend and tends to zero at a leaf size of 1. **However, the out-of-sample test RMSE drops until a minimum as leaf size decreases, but after a point (leaf size = 10), the RMSE increases as leaf size decreases.**

Overfitting is an undesirable machine learning behavior that occurs when the machine learning model gives accurate predictions for training data but not for new data.[2] Thus from the above observation, we can conclude that **overfitting occurs in DTLearner with respect to leaf_size, starting at leaf size = 10, and continues to overfit as leaf size decreases further.**

3.2 Bagging Learner - BagLearner

The Overfitting observed could be due to multiple factors like very little or very similar training data, or training the learner so much that it learns even the noise rather than just learning the general trends. To overcome these limitations, one solution is to combine predictions from several separate machine learning algorithms. Ensemble methods combine multiple learners to analyze sample data and pick the most accurate or mean of the outcomes. The two main ensemble methods are bagging and boosting. Boosting trains different machine learning models one after another, prioritizing the missed cases, while bagging trains them in parallel.

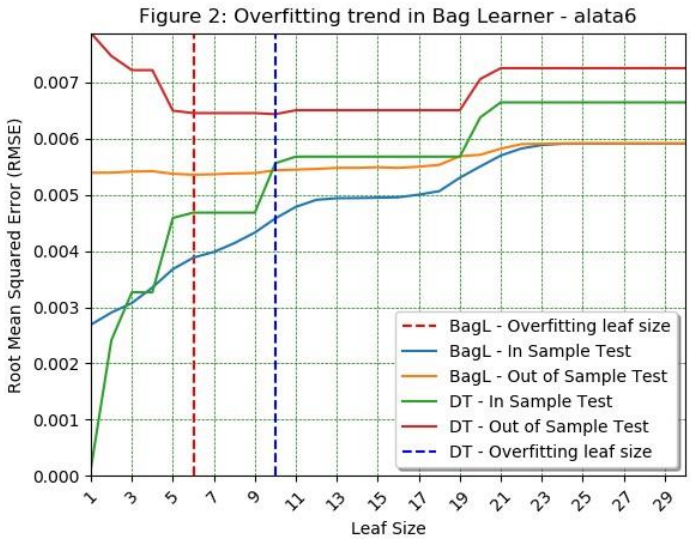

Figure 2 — RMSE vs Leaf Size observations for a Bag Learner in contrast with a Decision Tree Learner.

3.3 Random Tree Learner - RTLearner

R-squared (R2), a statistical measure quantifies the variance in the dependent variable explained by an independent variable in a regression model. Whereas correlation explains the strength of the relationship between an independent and a dependent variable, R-squared explains the extent to which the variance of one variable explains the variance of the second variable. Beta and R-squared are two related, but different, measures of correlation. Beta is a measure of relative riskiness. A mutual fund with a high R-squared correlates highly with a benchmark. If the beta is also high, it may produce higher returns than the benchmark, particularly in bull markets. R-squared only works as intended in a simple linear regression model with one explanatory variable. With a multiple regression made up of several independent variables, the R-squared must be adjusted. [4]

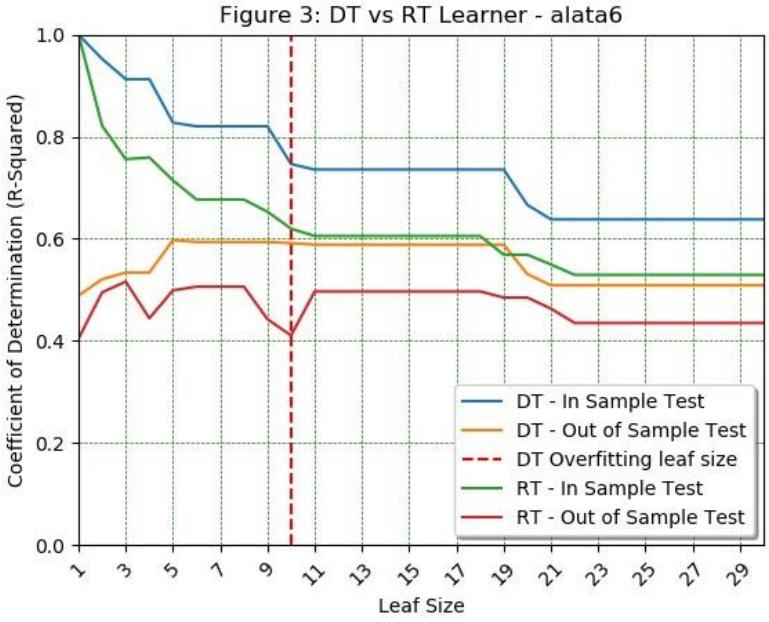

Figure 3 — METRIC 1: R-squared vs. Leaf Size observations for Decision Tree and Random Tree Learners.

The above hypothesis can be verified as seen in Figure 3. The R-square score is consistently higher for DTLearner than RTLearner, evidence of the fact that DTLearner can explain the variance higher than RTLearner, and DTLearner is

better at finding complex relationships in the data, even though the RMSE may appear to be close in Figure 5.

The RTLearner is exactly like the DTLearner except that the choice of feature to split on made randomly. Since in DTLearner, the split factor was chosen based on the highest correlation with target Y, there is an additional overhead of calculating the highest correlated column, which is no longer the case with RTLearner, thus the training time for the RTLearner is consistently lower than DTLearner. The difference in training times grows as the leaf size decreases since the RTLearner would have to generate more nodes with decreasing leaf sizes, thus more highest correlation calculations. This behavior can be verified in Figure 4.

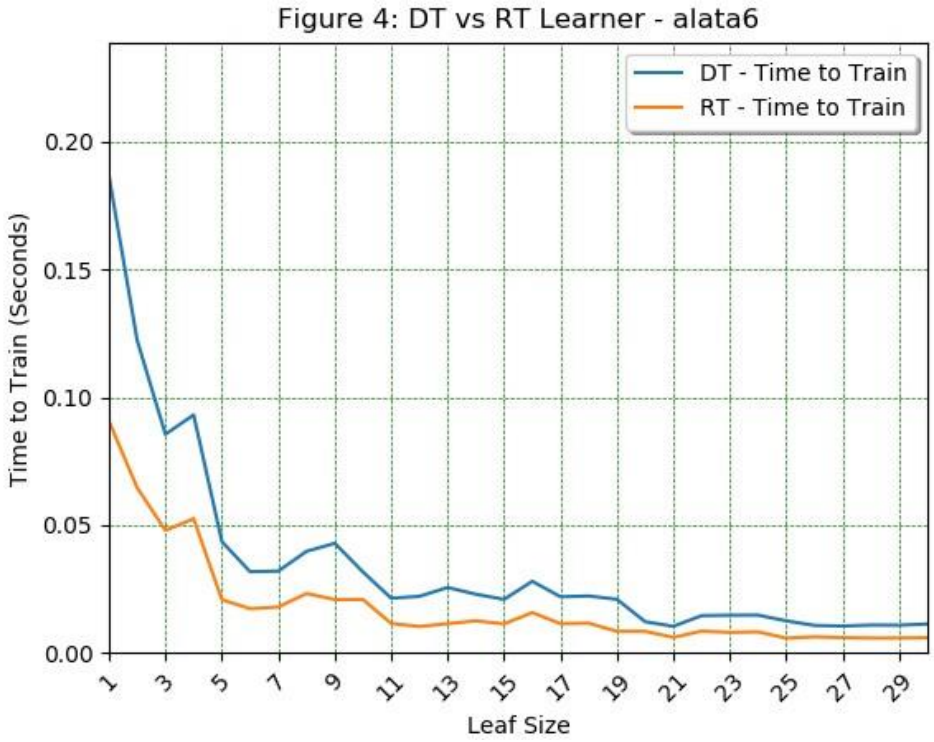

Figure 4 — METRIC 2: Time to Train vs. Leaf Size Observations for a Decision Tree and Random Tree Learner.

The difference in training times grows as the leaf size decreases since the DTLearner would have to generate more nodes with decreasing leaf sizes, thus more highest correlation calculations. This behavior can be verified in Figure 4.

From Figure 4, we can infer that the **RTLearner is better at the Time required to Train the Learner** since the additional overhead to calculate the highest correlated split factor goes out of the equation. This can become a critical factor in examples where calculating correlations becomes a performance overhead. **DTLearner can have its worst case when it generates a skewed tree** when the median is always the highest or lowest value. This was observed in the case of wine datasets.

From Figure 3, we can infer that **DTLearner is better at explaining the variance of the target variable** due to higher R-squared values. Figure 5 shows the RMSE comparison of DTLearner and RTLearner, although not a quantitative metric, we see that **DTLearner performed better than RTLearner** as the RMSE is consistently lower for the same leaf sizes and test samples. RTLearner also displays overfitting for lower leaf sizes, like DTLearner.

As both **DTLearner and RTLearner have their pros and cons, we cannot say that one learner will always be superior.**

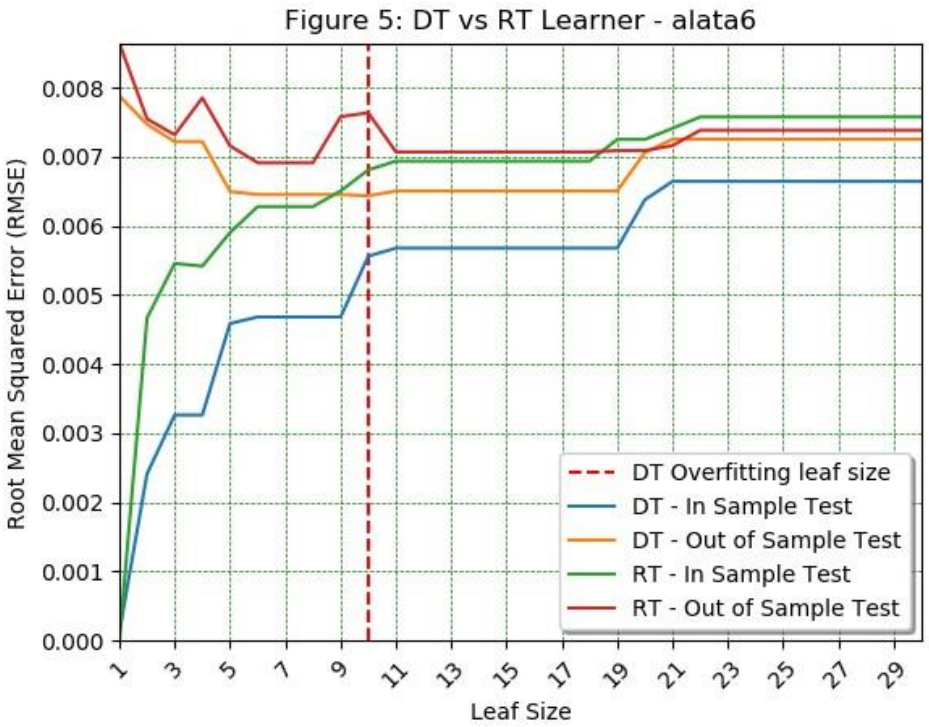

Figure 5 — RMSE vs Leaf Size observations for a Decision Tree and Random Tree Learner.

4 SUMMARY

The results of these experiments show that overfitting can occur with respect to leaf_size in the Decision Tree learner. A key finding of this project is that bagging can be an effective way to reduce overfitting in CART regression algorithms. However, it is important to note that bagging cannot eliminate overfitting completely. It is also important to choose a suitable number of bags to use, as too many or too few bags can lead to decreased performance.

Experimenting with other datasets, showed interesting trends. The wine datasets had high overfit start leaf size in the range of 27 for DTLearner, whereas RTLearner split each entry into leaves. Also, the Insane Learner had higher RMSE and lower correlation than BagLearner, even though InsaneLearner was an ensemble of multiple BagLearners, demonstrating the ill effects of overfitting.

In this experiment, we allowed the trees to grow without bounds, effectively allowing completely skewed trees as well. An interesting future investigation would be to evaluate the performance of the Learners when a max depth or max important features or pruning is introduced to limit overfitting.

Regularization methods try to eliminate those factors that don't impact the prediction outcomes by grading features based on importance. Using regularization to limit overfitting would also be an interesting future investigation.

5 REFERENCES

1. https://lucylabs.gatech.edu/ml4t/fall2023/project-3-documentation
2. https://aws.amazon.com/what-is/overfitting
3. https://en.wikipedia.org/wiki/Root_mean_square
4. https://www.investopedia.com/terms/r/r-squared.asp
5. https://numpy.org/doc/stable/reference/generated/numpy.corrcoef.html?highlight=corrcoef#numpy.corrcoef
6. https://numpy.org/doc/stable/reference/generated/numpy.argsort.html?highlight=argsort#numpy.argsort