# CS7637: Project 2: Assess Learners

Omkar Vaidya
ovaidya3@gatech.edu

***Abstract*—In this report, the behavior of various machine learning algorithms will be explored to predict the returns of a particular financial instrument, using the returns of other financial instruments. The machine learning algorithms used will be decision trees, random trees, and ensemble learners.**

## 1 INTRODUCTION

This report aims to use various machine learning algorithms to predict the returns of a particular financial instrument. For this analysis, the time series component will not be considered, and the returns of eight other financial instruments will act as predictors or features. To this end, the date column will be ignored. The three algorithms to be explored are the decision tree, random tree, and bag learner (type of ensemble learner). To analyze these algorithms, three experiments will be performed. The first will analyze the effect of overfitting on the decision tree, as the samples per leaf decreases. The second will analyze the effect of overfitting on the BagLearner algorithm. Lastly, the two tree algorithms (decision tree and random tree) will be compared in terms of two different performance statistics – time to train and mean squared error (MAE).

For the first experiment, I expect the performance to initially improve as the samples per leaf increase. This would be due to the algorithm capturing a greater number of real patterns with a more complex model (with more branches and leaves). However, at a certain point, at low values of leaf size, I expect the model to start overfitting, as the model captures more and more of the random patterns inherent in the data.

For the second experiment, I would expect the BagLearner to be less likely to overfit. This is because each learner would fit to a different data set, and the averaging process afterward would reduce the effect of learners fitting on random patterns.

For the last experiment, I expect the decision tree to perform better than the random tree in terms of accuracy or MAE. This is because the decision tree actively

selects the best feature to perform a split, whereas the random tree selects features at random, which may include features that aren't as useful. The difference between the accuracy of both models would depend on how predictive each feature is compared to each other. For the training time, I would expect the random tree to be faster, as it does not need to compute the best feature for each split.

## 2 METHODS

### 2.1 Decision Tree and Random Tree Implementation

For the first experiment, the decision tree algorithm uses the JR Quinlan approach (Quinlan, 1986). The algorithm is implemented recursively, and the tree is stored in a NumPy array. First, the best feature to split on is found. This is done by selecting the feature with the highest correlation with the response variable. The feature is then split in half using the median. Next, the left and right trees are built recursively. There are two base cases to be handled, one where the number input data points are below the specified leaf size, and one where all the response values are the same. For each recursion, the node number, split value, and both child node numbers are added to the array. For each leaf, the node number is replaced with a string. Keep in mind that NumPy arrays have a single data type, so when querying the tree, remember to convert the node number to an integer. There is also an additional error to handle when the leaf size is very low, and there are only two distinct values in the selected feature. For this, use the average mean of the feature to be the split value. To query the tree, recursion can be used again, to traverse the tree until a leaf is reached. The right child node number will always be one greater than the current node number, and the left child node number will always be the sum of left tree depth and the current node number plus one. Again, remember to convert the node number to an integer when slicing. The random tree is implemented in the same way, except that the best feature to split on is selected at random.

### 2.2 BagLearner Implementation

The first step in implementing the BagLearner is to create subsets of the training data, known as 'bags'. These subsets are the same size as the train set, but they consist of points which are selected at random, with replacement. If implemented correctly, each bag should be unique. On each bag, a learner is trained. When the BagLearner is queried, each individual learner trained on each bag will

also be queried, and the average mean of the outputs of each learner will be the BagLearner's prediction.

### 2.3 Experiment Set Up

The first experiment examines whether the Decision Tree algorithm overfits. To this end, the in sample and out of sample RMSE values are plotted against the leaf size. In this report, the x-axis will be inverted so that the leaf size will go towards zero. The point at which the out of sample RMSE exceeds the in sample RMSE is when overfitting occurs.

In the second experiment, the effect of overfitting on the BagLearner is examined. The BagLearner will have a fixed number of bags – 20 bags, and the input learner will be the Decision Tree. The same plot will be created for the BagLearner, and will be compared to the plot in the previous experiment.

In the last experiment, the performance of the Decision Tree and the Random Tree will be compared based on two metrics – training time and mean absolute error (MAE). Both metrics will be plotted against leaf size for both algorithms. For the MAE metric, the out of sample test set will be used to compare accuracy.

## 3 DISCUSSION

### 3.1 Experiment 1

Below is the plot for experiment 1, with the Decision Tree RMSE plotted against the leaf size. It appears that overfitting occurs at a leaf size of around 10, where the out of sample RMSE begins to increase significantly, and the in sample RMSE decreases to zero as each leaf corresponds to each point in the train set. The best accuracy occurs with leaf size 10-20, where the out of sample error is minimized remains constant at an RMSE of around 0.0057. This verifies my hypothesis that as the leaf size decreases, the algorithm will fit more and more to random patterns, and cause overfitting. To mitigate overfitting, it is prudent to select the optimal leaf size when making queries. In this case, a leaf size of 20 is optimal as that is where the out of sample RMSE stops decreasing.

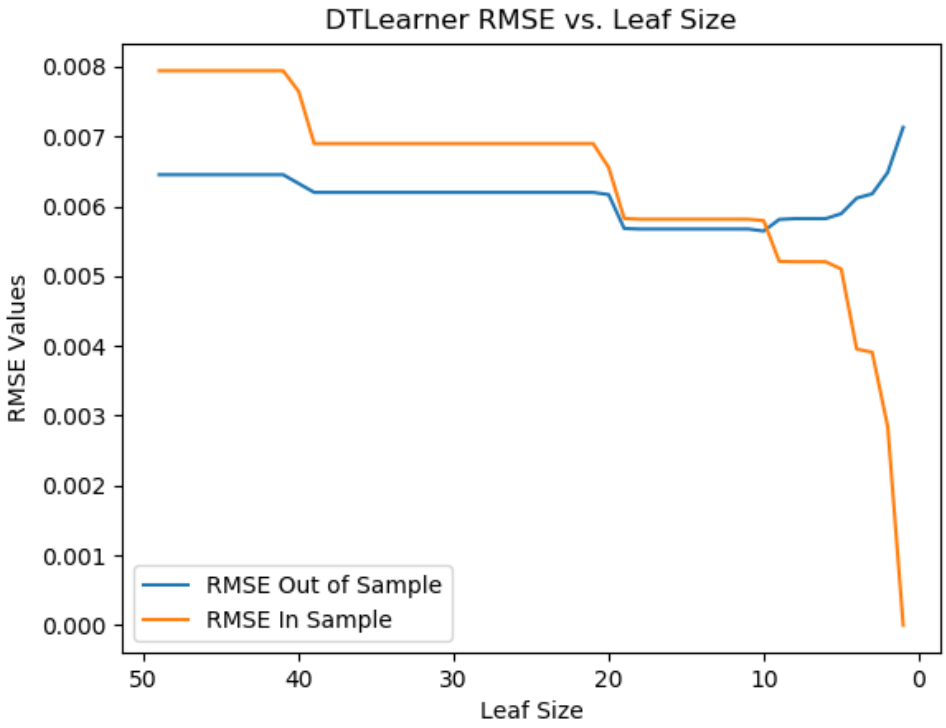


***Figure 1*** — Experiment 1 DT Learner Overfitting with Leaf Size

### 3.2 Experiment 2

In experiment 2, the plot differs from the plot in experiment 1, where the out of sample RMSE increases significantly as the leaf size tends to zero. Here, the impact of overfitting is significantly mitigated. The out of sample RMSE only increases slightly at leaf sizes below 10. This again verifies my hypothesis that the averaging process among unique bags results in less overfitting. The RMSE value is also lower compared to the plot in experiment 1, with a value below 0.0050. The BagLearner algorithm reaches its peak performance between leaf size 10 to 20, but the change in performance varies little among leaf sizes. The bag size used was 20, but perhaps if the bag size was increased further, overfitting could be negligible at lower leaf sizes. This also means that larger bag sizes could be used instead of smaller leaf sizes to improve performance.

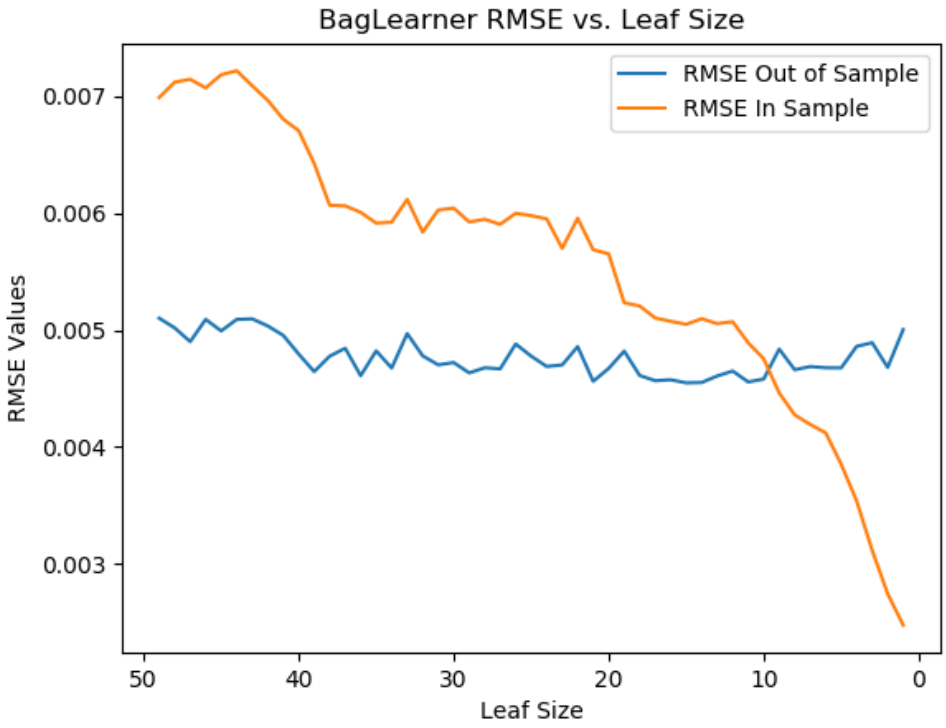


***Figure 2*** — Experiment 2: BagLearner Overfitting Vs. Leaf Size

### 3.3 Experiment 3

The plot for the Decision Tree and Random Tree train times is shown below. My hypothesis for the training time is verified, as the decision tree consistently had a higher train time compared to the random tree algorithm. It is roughly twice the time for the decision tree compared to the random tree algorithm. The time to find the best feature to split on appears to be the same as the time to create the rest of the tree. For the MAE plot, my hypothesis appears to be rejected. There is little difference between the two algorithms, and any difference is likely due to randomness in the data. The decision trees created for similar leaf sizes are likely to be the same tree, while the random trees are different each time they are created, which explains the difference in variation between the two trees.

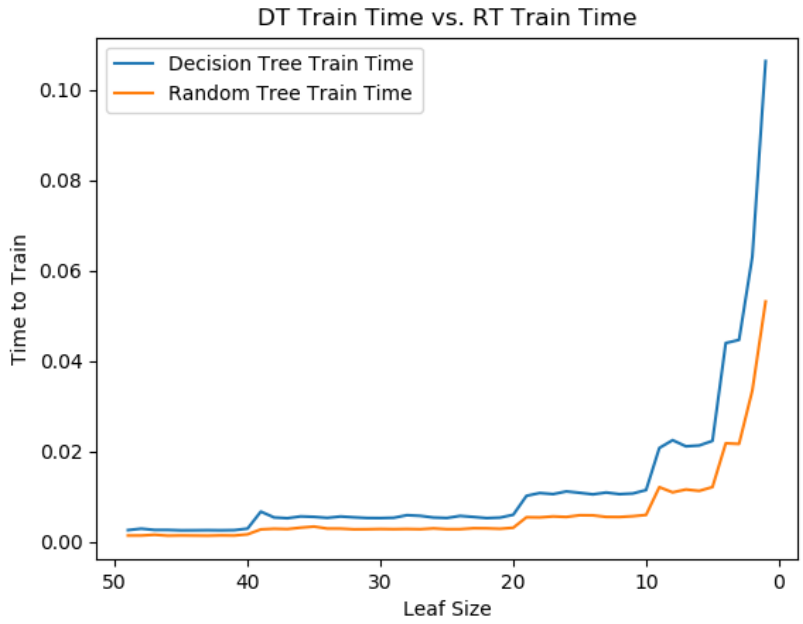


*Figure 3* — Experiment 3.1: Decision & Random Tree Train time

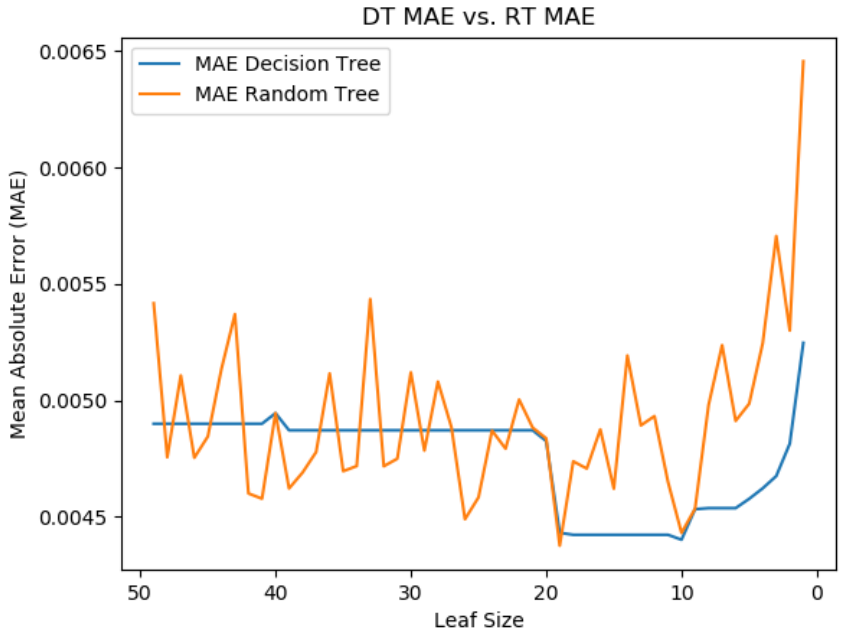


*Figure 4* — Experiment 3.2: Decision & Random Tree MAE

One possibility is that the correlation for each feature is very similar, so that it doesn't matter which feature is split. Perhaps if the dataset included features with varying absolute correlations to the response variable, there would be a greater difference in MAE between the two models. This is confirmed when I checked the correlation coefficients of each feature, which are as follows: 0.58889117, 0.69343965, 0.52252876, 0.65612867, 0.67349517, 0.5557239,

0.69473707, 0.71186044. The correlations only vary between 0.52 and 0.71. When choosing between the random tree and the decision tree algorithm, it is best to first construct a correlation matrix between the features and the response variable, to determine whether a decision tree is necessary. Once it is established that there is significant variation in the correlations, then a tradeoff can be made to decide whether the additional training time is worth the gain in accuracy.

## 4 SUMMARY

In summary, as leaf size decreases, overfitting is likely to occur in a basic decision tree. When bagging is used, overfitting decreases significantly across all leaf sizes, with an increase in accuracy compared to using a single model. Based on the empirical results, it is unclear whether a decision tree model outperforms the random tree model. Perhaps one area of future research would be to test the decision tree model on data with features that have varying correlations to the response variable, then compare it to the random tree model.

## 5 REFERENCES

1. Quinlan, J. R. (1986). Induction of decision trees. Machine learning, 1, 81-106.Joyner, D. A. (2017). Scaling Expert Feedback: Two Case Studies. In *Proceedings of the Fourth Annual ACM Conference on Learning at Scale*. Cambridge, Massachusetts.