# ASSESS LEARNERS

## Machine Learning for Trading

Amar, Alon
aamar32@gatech.edu

# Contents

Question 1 ..... 2

Question 2 ..... 3

Question 3 ..... 4

Metric 1 – Time ..... 4

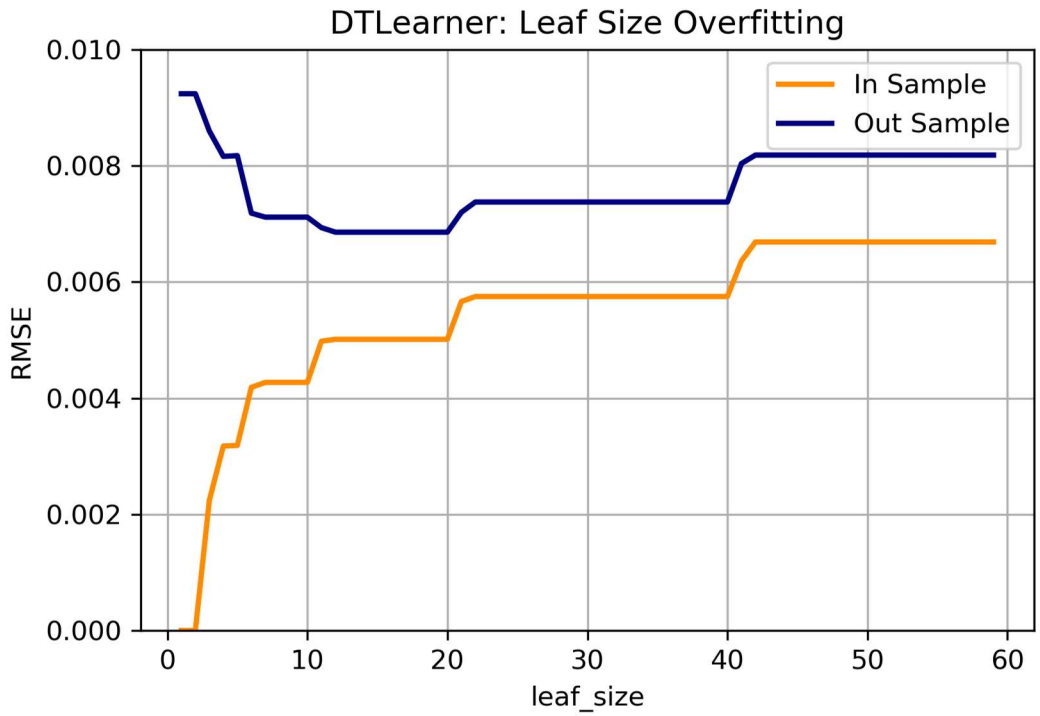
Metric 2 – Space ..... 5

## Question 1

To determine if overfitting occur with respect to the leaf size, I ran our DTLearner for 59 different values (1-59) of leaf size. For each value, I measured the RMSE of the in-sample and the out-sample. As stated in the lecture, we can define the area of overfitting where our in-sample error is decreasing, and the out-sample error is increasing.

I took the mean of 10 runs like that, each run had a different dataset (shuffled), so we can have a more general sense of the algorithm.

The results:




As you can see, the area of overfitting starts at around leaf_size = 11. All the area to the left of it is overfitted. That is something to be excepted: for leaf_size = 1, we basically have the exact answer for each sample in our training dataset. When leaf_size increase, we generalize our answer, hence, overfitting decrease.

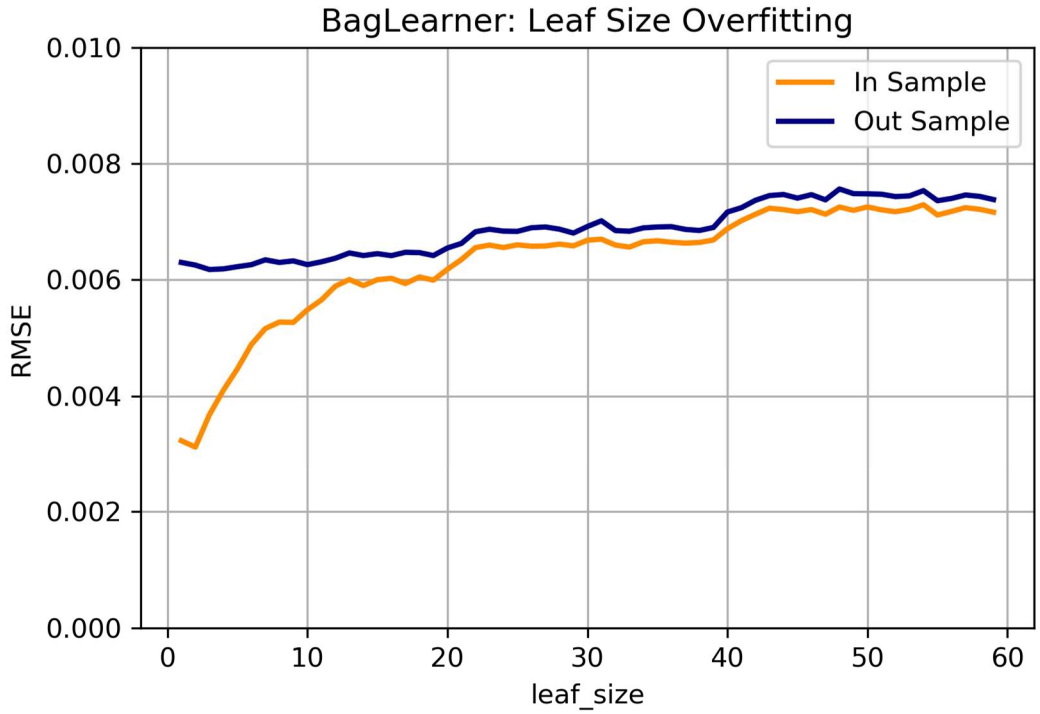
So, for the question “Does overfitting occur with respect to leaf_size?”, the answer is: yes, overfitting increase when our leaf size is getting smaller.

## Question 2

I conducted the same experiment as question 1: I ran our BagLearner with DTLearner, for 59 different values (1-59) of leaf size. For each value, I measured the RMSE of the in-sample and the out-sample. I took the mean of 4 runs like that, each run had a different dataset (shuffled), so we can have a more general sense of the algorithm.

What we would like to see here, is a smaller gap between the in-sample error and the out-sample error, or no gap at all.

The results:




As you can see, the overfitting again starts at around leaf_size = 11, but now, the error gap is way smaller and the error itself of the in-sample is higher (which mean that we don't have a perfect **fit** like before).

So, for the question "Can bagging reduce or eliminate overfitting with respect to leaf_size?", the answer is: yes, it can reduce overfitting but not eliminate it completely.

# Question 3

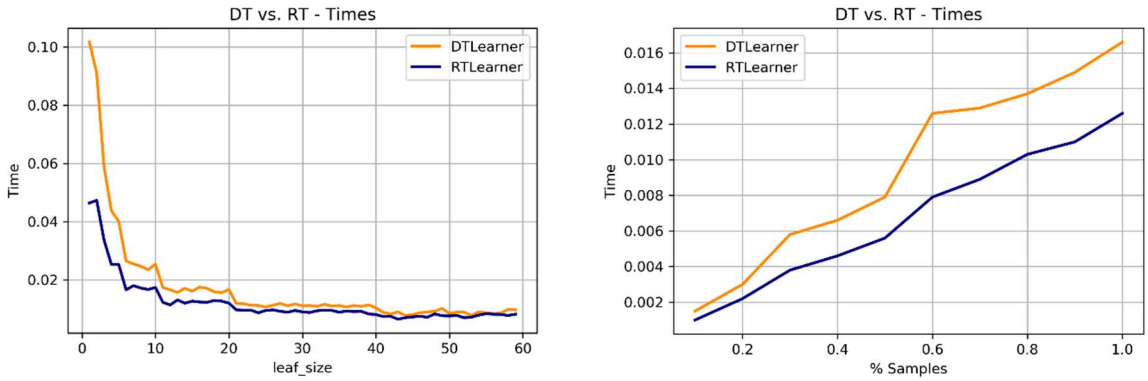
## Metric 1 – Time

The first metric I compared between DTLearner and RTLearner is time. I checked the total time of training and predicting for each method. To do so, I made 2 experiments:

1. I compared the time for different size of datasets, to see if there is any change as our dataset size increase (when leaf_size=11).
2. I changed the leaf_size and checked the effect on the times.

Like before, I measured the mean of 10 runs, to get better accuracy of reality.

The Results:




The x-label stated the percentage of our dataset that is being used.

As you can see, the time difference between the 2 grows as the dataset grow. We can conclude from that, that not only that RTLearner is faster in a constant time, but that the difference in time will be more significant for larger datasets.

On the other hand, we can see that the time gap shrinks as our leaf_size getting smaller. Even though the gap shrinks, we can clearly see that **RTLearner is faster**.

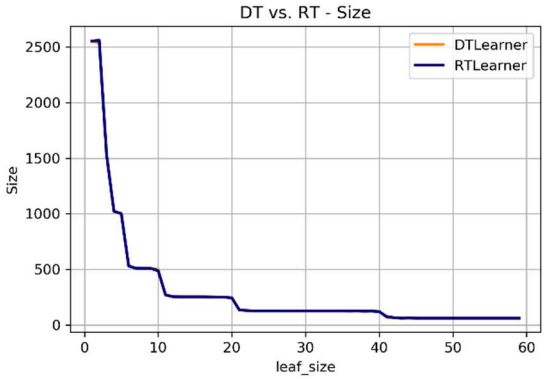
## Metric 2 – Space

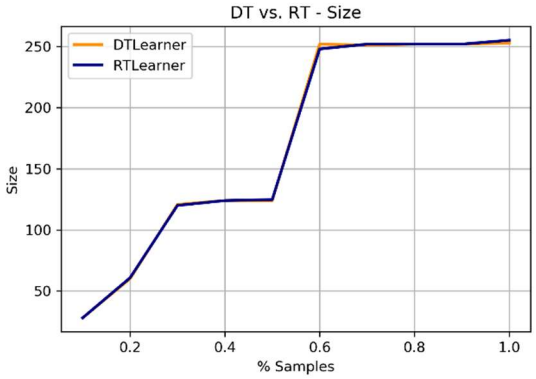
The Second metric I compared between DTLearner and RTLearner is the amount of space it is required for them. I checked the total size of the decision table for each method. To do so, I made 2 experiments (like the time metric):

1. I compared the size for different size of datasets, to see if there is any change as our dataset size increase (when leaf_size=11).
2. I changed the leaf_size and checked the effect on the sizes.

Like before, I measured the mean of 10 runs, to get better accuracy of reality.

The Results:




As we can see, no matter if we change the leaf size or the size of the dataset, the space the table takes for both algorithms **is roughly the same**.