Project 2: assess_learners

CS 7646 Machine Learning for Trading

Question #1 Does overfitting occur with respect to leaf_size?

Initial understanding

Overfitting is the most common criticism of traditional decision trees. Overfitting area occurs when in-sample error is decreasing and out-of-sample error is increasing. The leaf size for a model is the number of samples or observations that fit into that leaf. Smaller leaf size frequently results in overfitting, as smaller leaf size means that you allow the tree to have every leaf represent fewer and fewer samples of the training data. At a leaf size of one, each leaf represents one sample of the training data. As you increase the leaf size, you increase the broadness of data that may fit into that leaf. Thus, we would expect for overfitting to occur with smaller leaf size.

Experimental design

To test overfitting in the DTLearner as a function leaf size, I adjusted the testlearner.py file to run the learner for leaf sizes in range 1 to 30 as a *for loop*, storing the input leaf sizes and resulting root mean square error (RMSE) into lists that I then plotted. I did this once for the in-sample (training) data and then again for out-of-sample (test) data.

- *(dataset: istanbul.csv, learner: DTLearner, overfitting metric: RMSE, leaf_size: vary from 1 to 30+)*

Learnings

The resulting plot demonstrates that the ideal leaf size, with respect to minimizing overfitting, for the istanbul.csv file is around 10-18 samples per leaf. At these leaf sizes, the in-sample and out-of-sample RMSE are roughly equivalent, and neither is increasing.

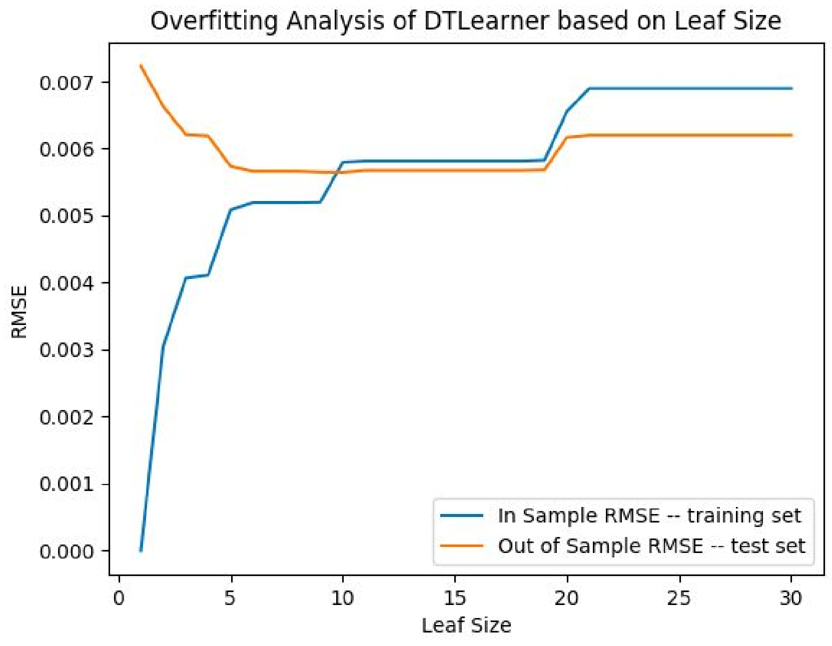


Figure: overfitting analysis of istanbul.csv with a single DTLearner and leaf size in range(1,30)

Prompt 1a: For which values of leaf_size does overfitting occur? Overfitting area occurs when in-sample error is decreasing and out-of-sample error is increasing. For the Istanbul.csv data file, overfitting occurs when the leaf size is fewer than 10 samples per leaf, though overfitting is greatest as leaf size approaches 1. The direction of overfitting, then is as leaf size decreases from 10 down to 1.

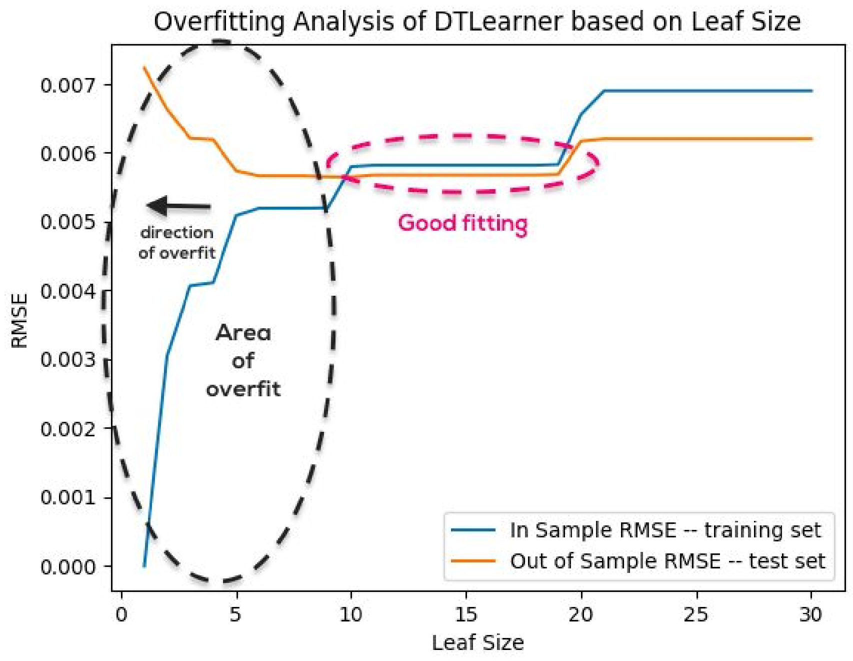
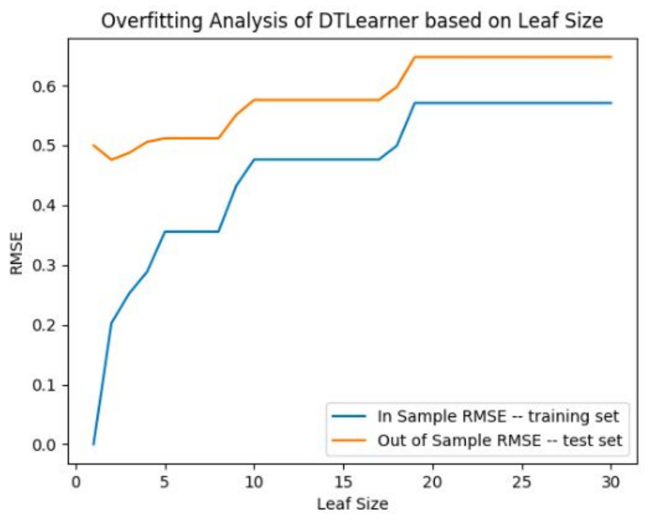


Figure: overfitting analysis of istanbul.csv with a single DTLearner and leaf size in range(1,30)

Additional experiments and learning

To satisfy curiosity, I also tested this out with other data files in the provided data set folder, and with both 3_groups.csv and ripple.csv (both of which have ~1000 samples, about double (2x) the number of samples in istanbul.csv). With these two files, I see a similar (expected) pattern where overfitting is significant until about leaf_size of 5.



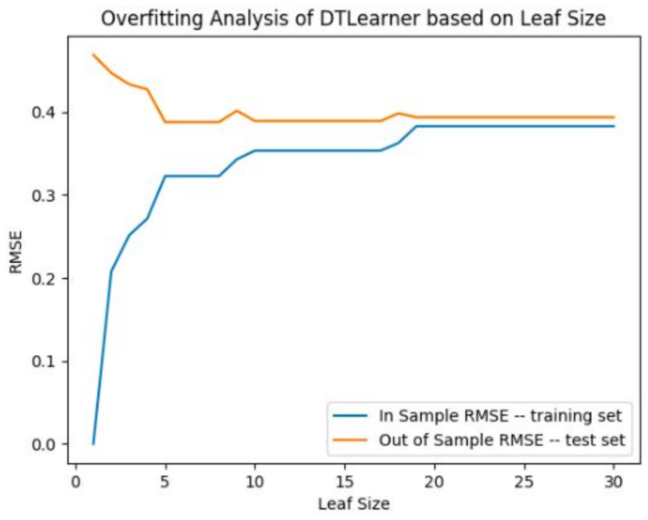


Figure left: overfitting analysis of Ripple.csv; **Figure right**: overfitting analysis of 3_groups.csv

I found a different result with respect to the point in leaf_size where overfitting begins to reduce. Analysis of the overfitting area in winequality-red.csv, a file with 1600 samples, demonstrates that overfitting area ends after about 35 leaf size. I read in various online forums that the ideal leaf size is around 50. This leads me to believe that ideal leaf size is correlated with number of samples in a data set, though this is a more complex issue and other changes (ie pruning) can be made to a tree to reduce overfitting after training is complete.

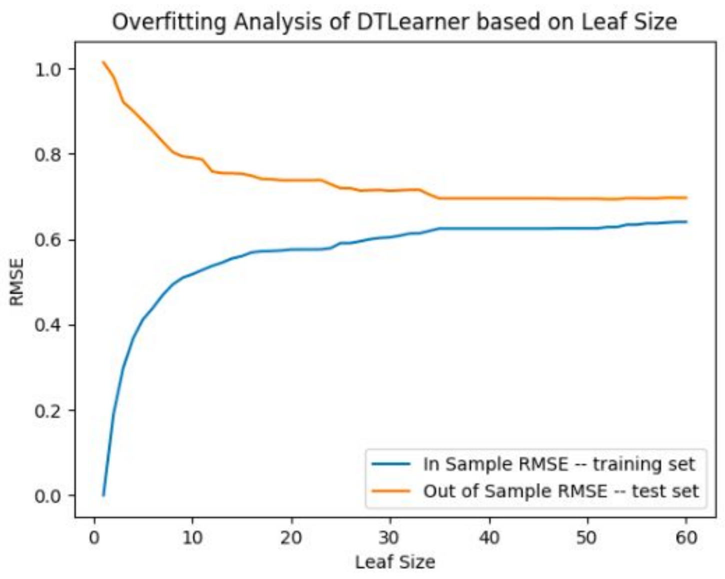


Figure: overfitting analysis of winequality-red.csv with leaf size in range(1,60)

Question #2 Can bagging reduce or eliminate overfitting with respect to leaf_size?

Initial understanding

Bagging is an ensemble learning technique that can decrease the variance of a single estimate by combining several estimates from different models and by generating additional data while training. In bagging, N additional training data sets are produced by random sampling with replacement from the original set. By sampling with replacement, some observations may be repeated in each new training data set, and any element has the same probability to appear in the new data set. These multiple sets are used to train the same learner algorithm and therefore different classifiers are produced. With bagging, each model is built in parallel (independently from others). Bagging usually results in models with higher stability and **less overfitting than a single model.** Thus we expect that bagging should reduce (but likely not eliminate) overfitting given a specific leaf size.

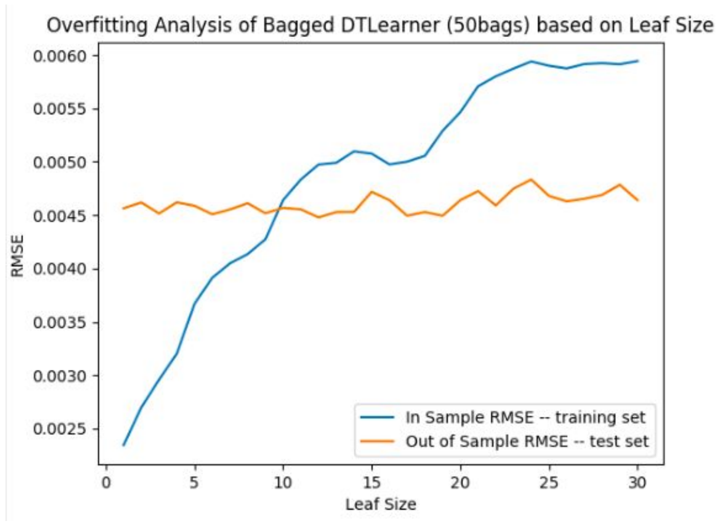
Experimental design

To test this hypothesis, I modified the testlearners.py file once again, still iterating over leaf size in range(1,31), but this time using the BagLearner with 50 bags.

- *(dataset: istanbul.csv, learner: DTLearner in BagLearner, bags: 50, leaf_size: iterate from 1 to 30).*

Learnings

The resulting plot demonstrates that the addition of bagging the DTLearner models stabilizes and reduces the out of sample RMSE to around 0.0045, lower than the lowest out-of-sample RMSE in the single DTLearner. In our bagged DTLearner experiment at a leaf size of ~10, you see that the RMSE of in sample matches that of out of sample RMSE, the same leaf-size that this phenomena occurred in the single DTLearner, but in the bagged example, the intersection occurs at a much lower RMSE (~0.0045 vs 0.0057 in the single DTLearner). This means that bagging does reduce overfitting, and depending on how you define the threshold for defining an overfitting region, you can argue that overfitting is nearly eliminated by bagging. Why? The RMSE in the bagged model is nearly flat, and it would be hard to differentiate between noise of the small sample size versus meaningful slope changes in the region.



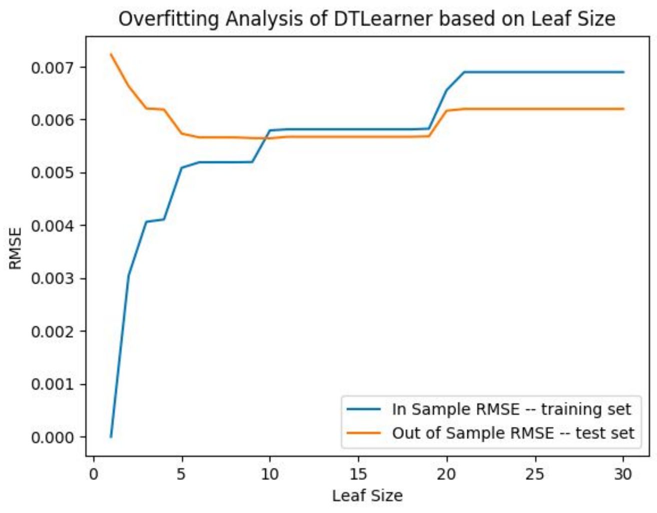


Figure left: overfitting analysis of DTLearner (50 bags). **Figure right**: overfitting analysis of a single DTLearner

Question 3: Quantitatively compare "classic" decision trees (DTLearner) versus random trees (RTLearner)

Initial understanding

Decision trees are prone to overfitting, especially with deeper trees. They can also be a bit greedy with respect to making decisions for optimizing each node vs. global optimization. Both of these issues lead to error in predictions (error due to bias and error due to variance). The thinking is that random forests can mitigate these two sources of error by acting as a collection of random trees, aggregated into one final result. Another advantage of random trees is that they are cheaper to produce since the split features are chosen randomly versus through statistical analysis as in traditional decision trees.

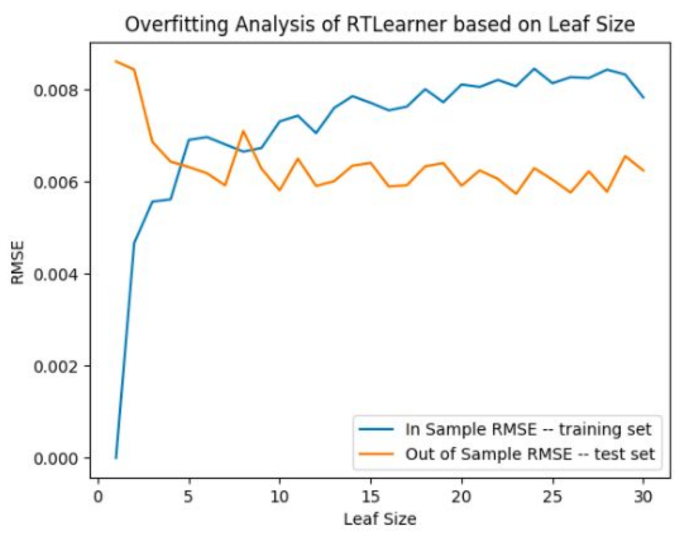
Experimental design

To quantitatively assess traditional decision trees versus random trees, I modified the testlearners.py file once again to allow me to compare a single DTLearner to a single RTLearner over various leaf sizes in the range of 1 to 30. I also ran both learners bagged to compare them as ensembles, out of curiosity.

- *Single tree: (dataset: istanbul.csv, overfit metric: RMSE, leaf_size: 1 to 30)*
- *Bagged: (dataset: winequality-red.csv, bag size: 20, overfit metric: RMSE, leaf_size: 1 to 60)*

Learnings

Unbagged and as single trees, as you can see from the data plots below, the random tree never achieves the tightness of difference between in-sample and out-of-sample RMSE that the DT achieves. The single RT average out of sample RMSE is in the range of 0.006, whereas the DT learner is around 0.055. The RT learner was built very quickly, however. Also interesting to note is that the area of overfit for the RTLearner ends at leaf size of 5, whereas it ends at a leaf size closer to 10 in the classic decision tree. Also, the RTLearner maintains a fairly stable in-sample and out-of-sample RMSE for all leaf sizes >5, whereas the DTLearner in-and out-of-sample RMSEs diverge at very large leaf sizes.



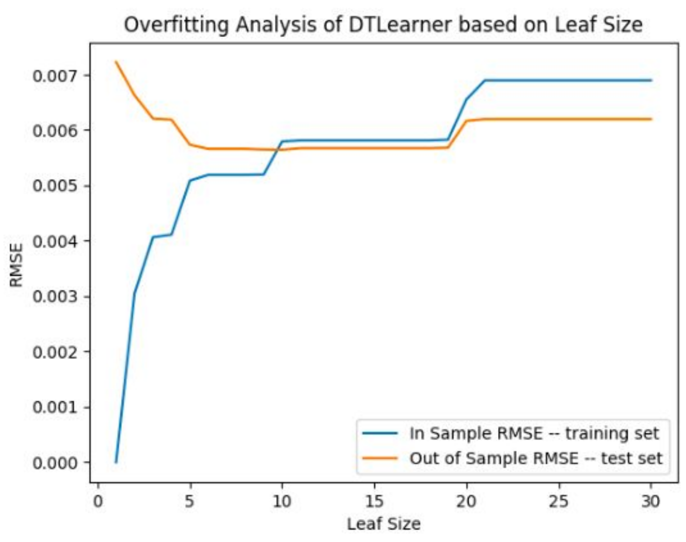
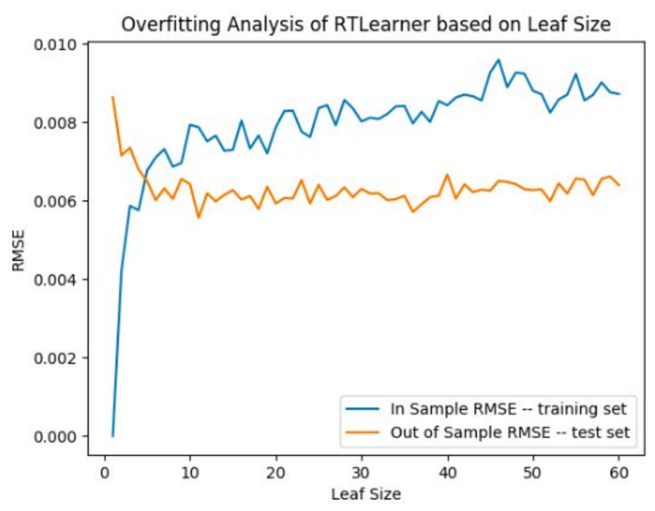


Figure left: overfitting analysis of a single RTLearner. **Figure right**: overfitting analysis of a single DTLearner



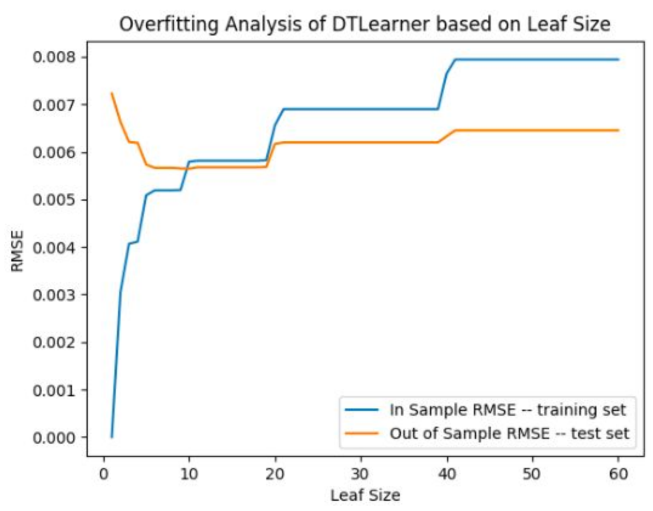


Figure left: overfitting analysis of a single RTLearner **Figure right**: overfitting analysis of a single DTLearner

Prompt 3a: In which ways is one method better than the other?

Random tree

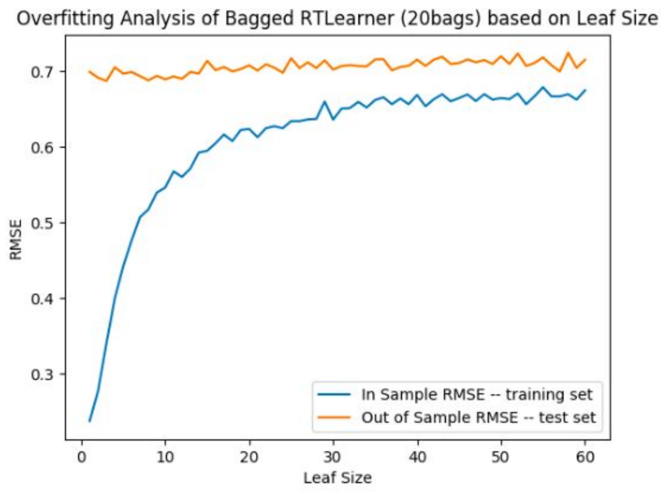
- **Speed:** Faster to build by many (pick random feature to split on vs calculating correlations)
- **Less Overfitting:** Less prone to overfitting than traditional decision trees (overfitting area ends at a smaller leaf-size in RT vs DT)

Decision tree

- **Lower RMSE:** Compared to random tree, higher quality output / predictions (lower RMSE at ideal leaf size)
- **Less Stable RMSE:** At higher leaf sizes, more divergence in in-sample vs out-of-sample RMSE (vs stable diff in RT)

Curiosity experiment, not part of the rubric
Things could get interesting when we start to bag. With bagged traditional (correlation) decision trees, you are getting a reduction in overfitting as we noted in the second question, but it's still just more of the same type of tree, amplified by data. With bagged random trees, you add in diversity of randomness which pairs speed of construction with mitigations for the error sources of decision trees (bias and variance).

In the plots below, you can consider the error looking at the distance between each line for each leaf size value. While both models have similar RMSE for out of sample (~0.7), the difference between in-sample and out-of-sample RMSE is lower for the random forest. Note also that due to the speed with which we might be able to generate random forests, it's possible that if we ran many many more trials, we would find an outstanding random forest that far excels versus the decision tree. In any case, if speed is of interest, the bagged random trees below perform comparably in RMSE to the bagged decision trees, which means that random trees win for efficiency and speed.



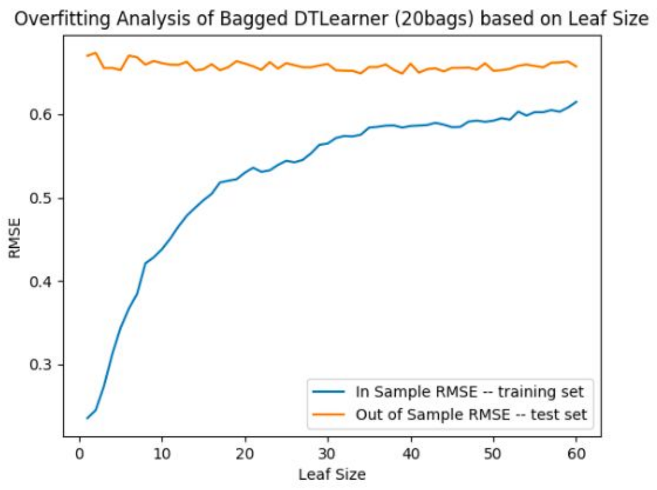


Figure left: overfitting analysis of a 20 bag RTLearner. **Figure right**: overfitting analysis of a 20 bag DTLearner

Resources of great use in creating this report

- Question2: https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/
- Question3: https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd