Vincent La (Georgia Tech ID: vla6) Assess Learners Report

Overfitting can occur with respect to size of the leaf. The reason for this is that if you have a small leaf size you are training your data such that the model is more prone to capturing noise in the training data. However, if you have a large leaf size, then you are aggregating more of the outcome variable in a leaf, so you will be capturing less noise. Inherently, there is a bias-variance tradeoff. If you have a large leaf size, then you have less variance but more bias. For example, in the extreme if your leaf size is the size of the entire training set, then you'll always just predict the average of the training set. However, if you use a leaf size of one, which is the smallest leaf size, you will have low bias but very high variance.

To illustrate this, we conduct an experiment. With a 60/40 Train/Test split, we fit a Decision Tree Learner (using DTLearner.py) on the Istanbul.csv data set. The results are represented in the table below where each row corresponds to the Test RMSE for a given Leaf Size.

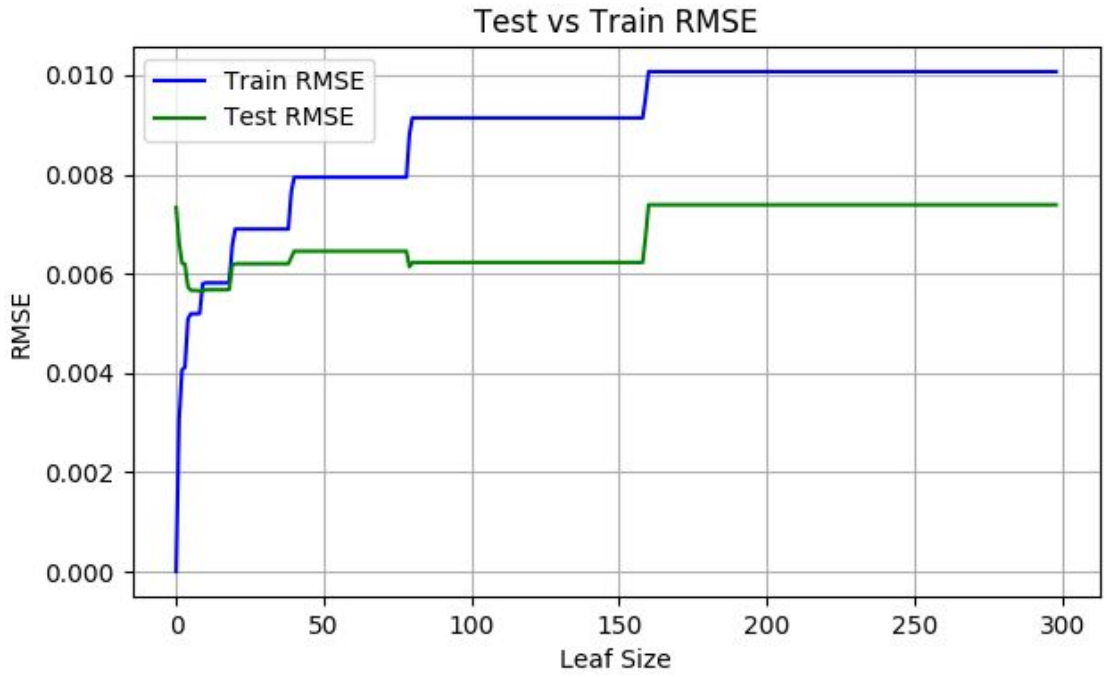
Table 1: Train vs Test RMSE (60/40 Train/Test Split) Data: Istanbul.csv

leaf_size	test_corr	test_rmse	train_corr	train_rmse
1	0.691174	0.007332	1	0
2	0.73771	0.006629	0.96582	0.003051
3	0.755732	0.00621	0.938435	0.004066
4	0.75707	0.00619	0.937029	0.004111
5	0.76725	0.005735	0.901767	0.005087
6	0.772267	0.005662	0.897443	0.005192
7	0.772267	0.005662	0.897443	0.005192
8	0.772267	0.005662	0.897443	0.005192
9	0.773108	0.00565	0.897204	0.005198
10	0.768863	0.005646	0.87039	0.005795
11	0.765677	0.005675	0.869437	0.005815
21	0.704048	0.0062	0.810256	0.006898
40	0.680159	0.006327	0.76027	0.007646
41	0.657005	0.006453	0.738011	0.007942
80	0.680125	0.006139	0.664161	0.008799
81	0.6651	0.006226	0.630305	0.009138
159	0.6651	0.006226	0.630305	0.009138
160	0.610406	0.006753	0.583917	0.009555
161	0.499099	0.007386	0.517763	0.01007
299	0.499099	0.007386	0.517763	0.01007

In the table above, we show results (RMSE) as a function of leaf size, given a 60/40 train/test split. As you can see above, with a leaf size of 1, the Train RMSE is equal to 0 and the correlation between the predicted Y and the actual Y values is equal to 1. This makes sense since if leaf size is 1 you will completely and perfectly fit your training data, at risk of overfitting. In this case, given the table above, we can see that overfitting occurs for leaf sizes 1 through 5. The test RMSE is minimized with leaf sizes

of 6 through 8. After that, the test RMSE increases. This is because at this point there is an inflection in the bias-variance tradeoff. That is, at very low values of leaf size, the variance is very high, but if we increase the leaf size too much then the bias term becomes very high. We can also see this in the graph below.

Figure 1: Train vs Test RMSE (60/40 Train/Test Split) Data: Istanbul.csv; Learner: DTLearner



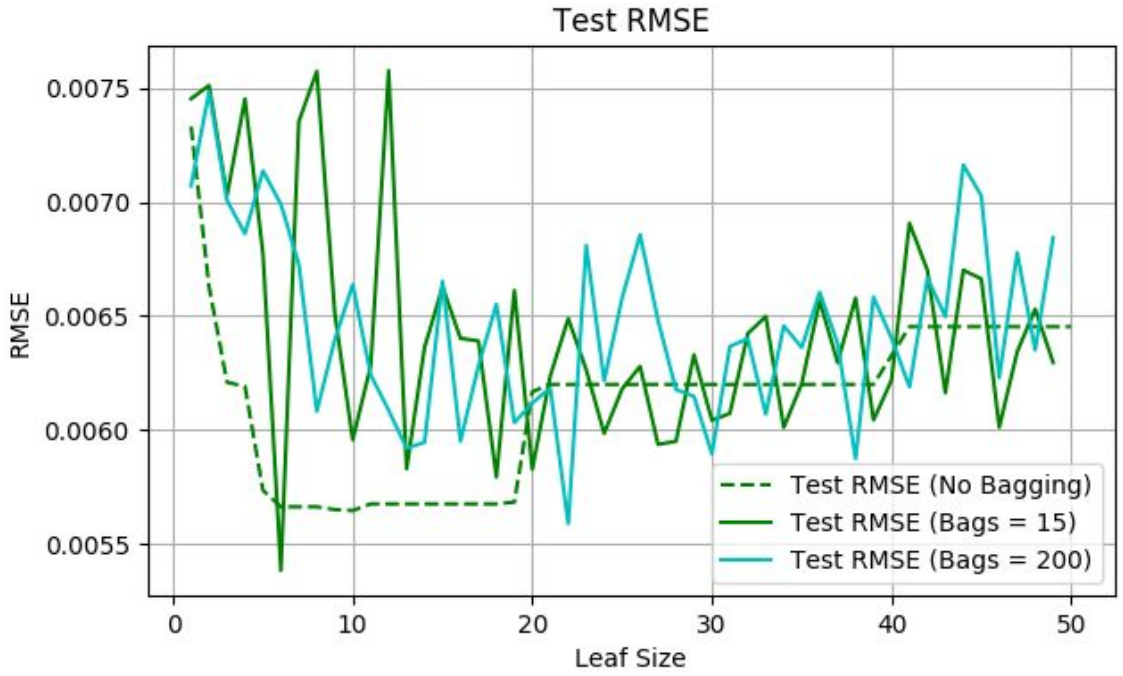
Secondly, in general, bagging may reduce overfitting with respect to leaf_size. This is because bagging takes the average of many different learners that are trained over datasets that are sampled with replacement of the original data set. By taking the average of many different learners, this helps reduce the variance associated with training with lower leaf_size.

However, bagging with decision trees is unlikely to yield much better results. This is because while we are using bagging to reduce the variance, since the decision tree chooses to split using the highest correlated feature, all the trees are very highly correlated with each other. Taking the average of very highly correlated trees does not actually reduce the variance by much.

We illustrate this by choosing two different bag sizes: 15 and 200. We then create the same plots of train and test RMSE but with those different bag sizes. Again, we are using DTLearner with a 60/40 train/test split and the data we are using is the Istanbul dataset.

As you can see in the Figure 2 below, we can see that the test RMSE of (1) no bagging, (2) Bag size = 15, and (3) Bag Size = 200 seems to be very similar to each other (very minimal difference). Again, this is likely due because when averaging highly correlated trees together, we don't actually reduce variance by much.

Figure 2: Test RMSE Bagging Comparison; Data: Istanbul.csv; Learner: DTLearner

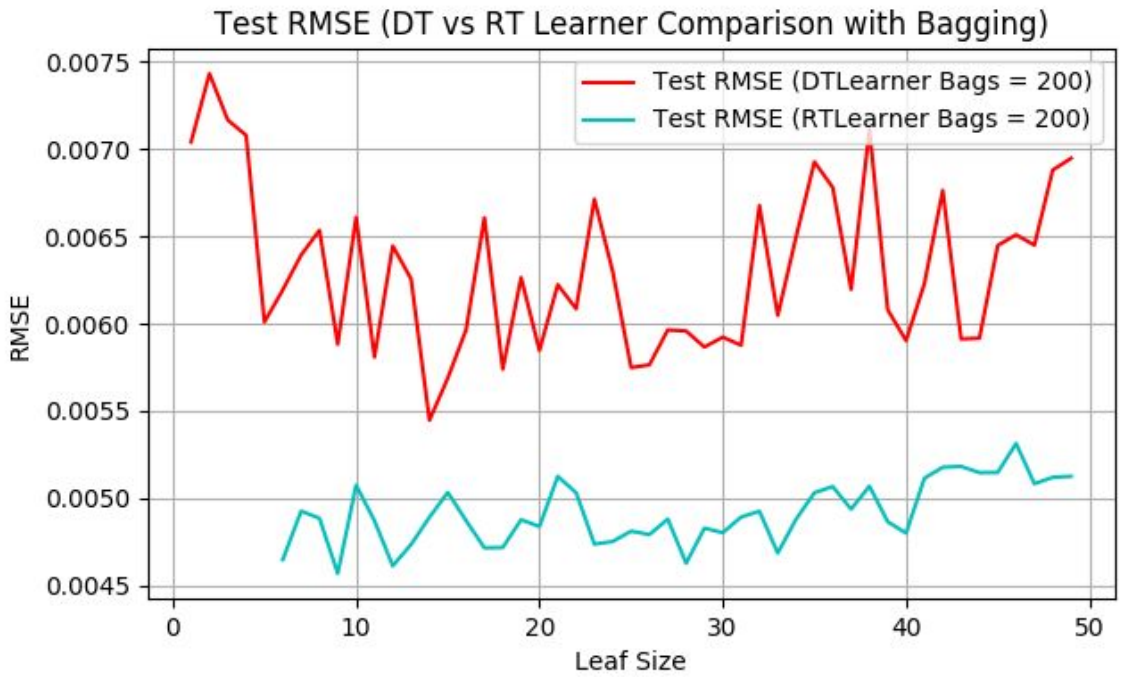


Thirdly, we compare "classic" decision trees (DTLearner) versus random trees (RTLearner). Again, this comes down to a bias-variance tradeoff. With random trees, the learner behaves exactly like the decision tree learner except that instead of picking the feature that has the highest correlation with the Y variable, it picks a feature randomly. This means that bias will be higher but variance will be much lower, so RT Learner will more likely not overfit. This also is the basis for the Random Forest model. In fact, when RTLearner is used in conjunction with bagging, this is actually essentially the Random Forest model.

When only fitting one tree, it's plausible that a Decision Tree Learner performs better than a Random Tree Learner. This is because the bias associated with a Random Tree Learner can be higher since it's choosing a random feature to split whereas Decision Tree Learner is choosing a feature with highest correlation. However, Random Tree Learner is probably better when used with bagging. This is because while bagging can reduce variance with the Decision Tree Learner as well, since in Decision Tree Learner you are always picking the feature with the highest variance, each tree will be very highly correlated with each other. Thus, averaging the result of lots of correlated trees actually may not reduce the variance by that much. On the other hand, with Random Tree Learner because you are choosing a different feature each time randomly, the trees will not be as correlated with each other so by bagging you are actually reducing variance.

To test these hypotheses, we conducted the following analyses. Again we use the Istanbul data set. We then fit a DTLearner with 200 bags and an RTLearner with 200 Bags. We then repeated this over various leaf sizes ranging from 1 to 50. We then plotted the results in Figure 3(a) below.

Figure 3(a): Test RMSE DT Learner Vs RT Learner With Bagging; Data: Istanbul.csv



In the figure above, the red line is the Test RMSE from DT Learner using Bagging (Bags = 200) and the blue line is from RT Learner using Bagging (Bags = 200). As you can see the RT Learner actually performs substantially better than DT Learner with bagging. Again, this is likely due to the fact that with RT Learner, the correlation between trees is small so there truly is a reduction in variance. In terms of the bias-variance tradeoff, while the DTLearner may be less bias, there is so much variance and the reduction in variance the RT Learner achieves with bagging is much greater because the trees are much less correlated with each other since the split is done by choosing a random feature each time. On the other hand, with a DTLearner, the split is chosen using the highest correlated feature so all the trees are very correlated with each other.

Finally, the second thing we test is DTLearner vs RTLearner with no bagging. In the context of bias-variance tradeoff here, because we are doing no bagging, the RT Learner does not achieve the same massive reduction in variance and the DTLearner is less biased. We can see the results of this below (again we are using the Instanbul data set with 60/40 train/test split).

In Figure 3(b) below, we show the results of testing DT Learner vs RT Learner with no bagging. As you can see at lower leaf sizes, the DT Learner actually does better than RT Learner with no bagging. This is because at lower leaf sizes, DT Learner is less biased, and RT Learner does not achieve same reduction in variance without the bagging. At higher leaf sizes though we can see the performance between DT Learner and RT Learner more or less the same. This is because at higher leaf sizes, the bias increases so much and variance decreases so performance between DT and RT Learner is the same.

Figure 3(b): Test RMSE DT Learner Vs RT Learner No Bagging; Data: Istanbul.csv