Stephen Via
9/22/2019
ML4T
Learners Report

Q1.

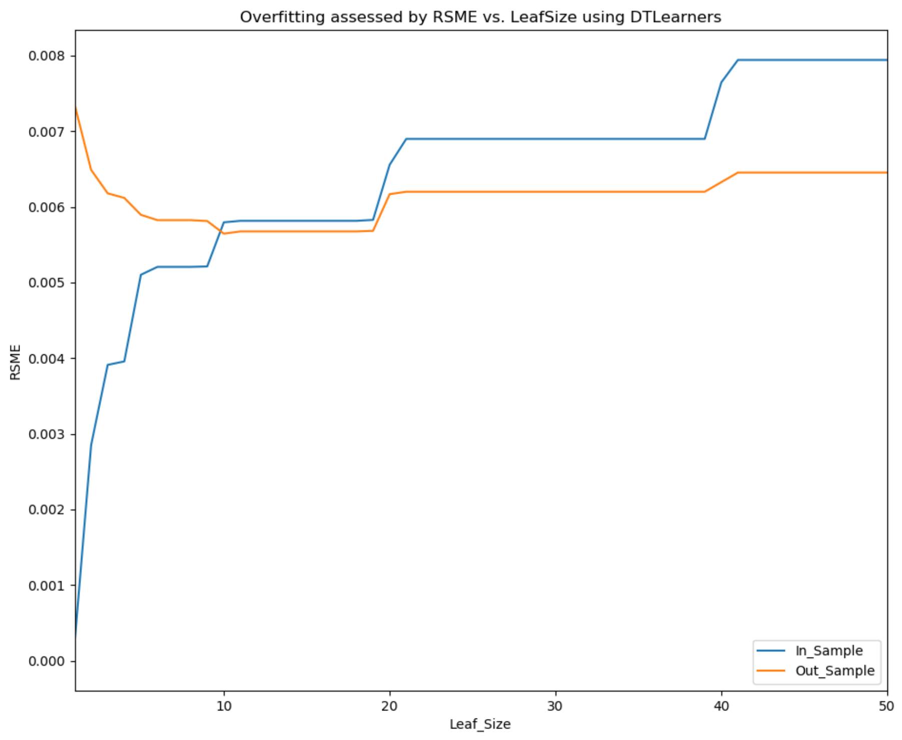

Figure 1. Assessing overfitting using a RSME metric on Instanbul.csv data

Creating a decision tree that is overfit for our data implies that the degree of flexibility in the model allows said model the freedom to potentially hit as many data points as possible, generalizing the data so well that it now cannot predict for new out-of-sample data. Increasing the degree of the model's polynomial to capture every point of data produces the aforementioned phenomenon known as "overfitting." Because of this assumption, our theoretical model now predicts the noise in the data along with the relationship, which means that new data will also map to an incorrect output driven up or down by noise.

In the above example, overfitting is portrayed as the section of the graph where the in sample and out of sample error approaches a leaf size of 10 and intersect at a RSME just below 0.006. This metric portrays how accurately the predicted output is hitting the value of the actual output, evaluated to represent larger mistakes with a larger penalty, using a squared term in the root square mean error formula. The overfitting occurs at a leaf size value of 10 because the error for the in-sample data begins to surpass the error for predicting data the is out of sample. Overfit data will perform much better, with a smaller error metric as a decision tree has a node with a specific output to perfectly fit each of the input values that it is trained on. When in sample data cannot be predicted as accurately as out of sample data, we see that the tradeoff of for using a larger leaf size proves to be useful in producing a generalization for new out of sample data.

Q2.

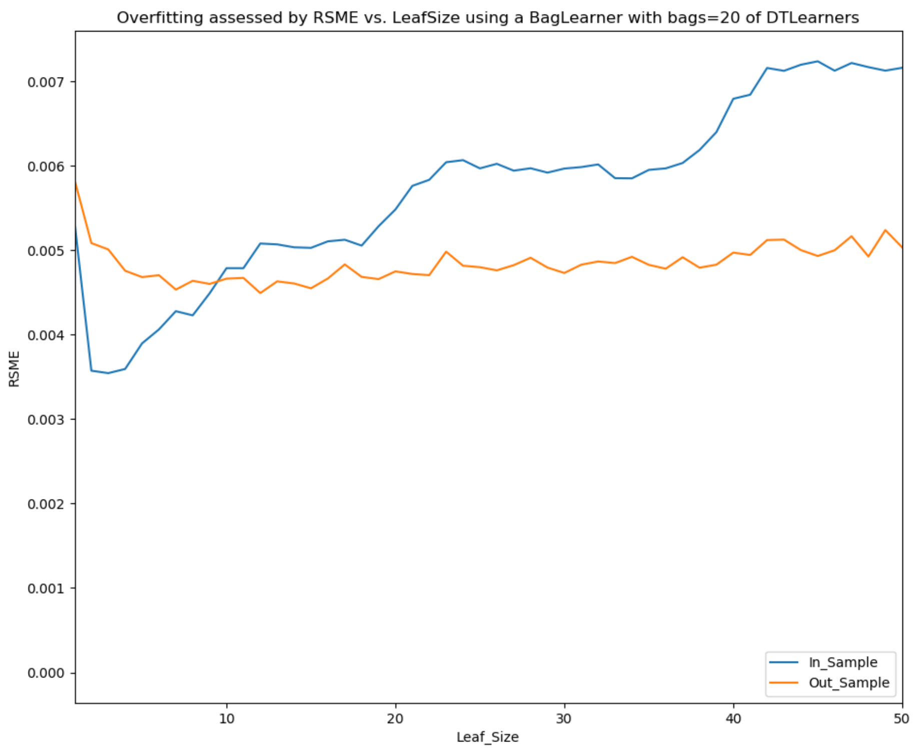

Figure 2. Assessing overfitting using a RSME metric on Instanbul.csv data using a bag size of 20 on DTLearners

The above graph portrays that bagging can reduce overfitting because the RSME for out of sample data because the RMSE is consistently lower than when using a simple DTLearner of varying leaf size, hovering around a RSME of 0.005. The in sample RSME appears to fail at accurately predicting data as the leaf size rises above 10 due to the fact that the leaf nodes begin to aggregate more of the training data's rows into leaves, meaning that the model is too general to predict on data that it has previously been trained on, in sample. This is a benefit of using a bag learner due to the fact that generalizing our model performs relatively similar throughout the entire range of leaf sizes between 10 and 50, again hovering around 0.005.

The reason that bag learning is so effective is due to the fact that each bag aggregates 20 separate DT learners that each produce their own array of y prediction values. These values are averaged together to form an even better estimate of what output data will map to, using 20 DT learning models instead of one. Using random sampling in a bag learner introduces more variability through random chunks of training data to remove biases into our model, also aiding in the fight against overfitting. The DTLearners in the BagLearner model split baised off of the best correlation on the training data, which will vary due to the random sampling used in bag learning. The best split feature is now varied on the training data, and thus this increases the amount of randomness in the model to make sure that the training data does not fit noise in one specific sequence of input data. This means that each of the training models, each of the 20 DT Learners takes in a different subset of data which is produced as a random selection of rows from the 'trainX' data, indexed by a masked array of random integer in the range of 0 to the number of rows with a size of trainX's number of rows. This is how random sampling is performed on the BagLearner model.

Q3.

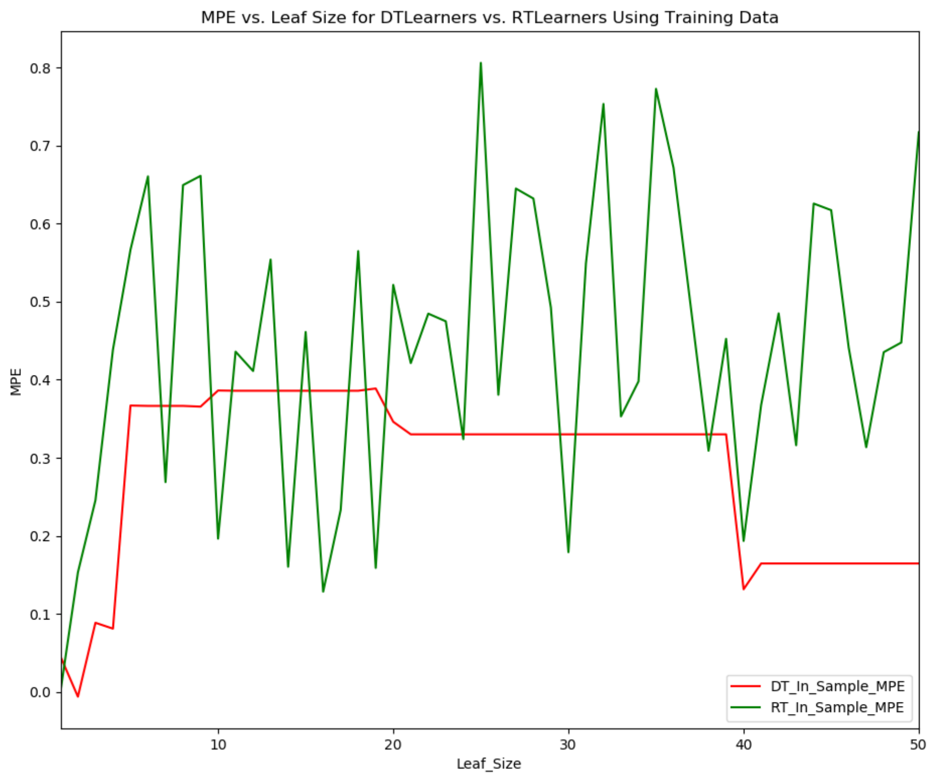

Figure 3. MPE metric of DTLearner vs. RTLearner using in sample data from Istanbul.csv

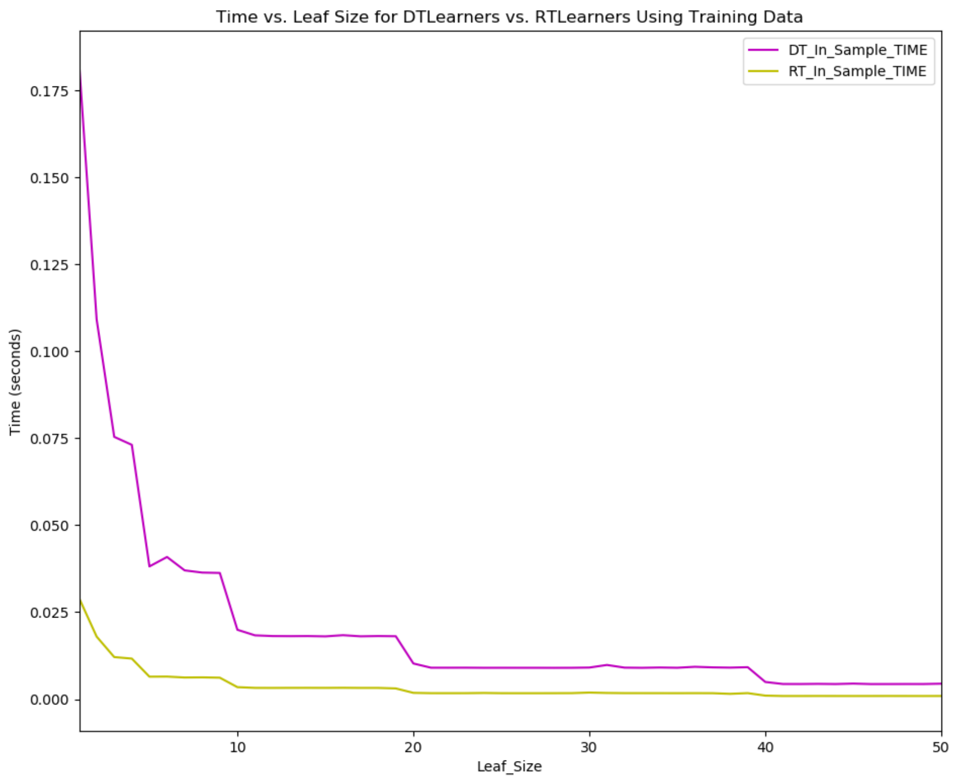

Figure 4. Time metric assessing build time of DTLearner vs. RTLearner models using in sample data from Istanbul.csv

The above two graphs perform two new metric measurements on the Decision Tree Learner and the Random Tree Learner models in order to assess the ways in which one model is better than the other. The Mean Percentage Error, as depicted in Figure 3. for in sample data, is an assessment on how close the model's predictions are to the true value of the input data resulting in an average, much like root square mean, but incorporates the bias for positive or negative errors that Mean Absolute Percentage Error. The second metric portrayed in Figure 4. is assessed as the time it takes to build the trees at resulting aggregated leaf sizes for in sample data when comparing the two models.

In order to analyze the first metric, we must first look at what the Mean Percentage Error's purpose is as a cost function. Because positive and negative errors will cancel out, because the absolute value operation is voided, it is difficult to make an assessment on how well the models performed overall. This tradeoff is used to assess how the models perform systematically by viewing whether or not the model overestimates or underestimates based on the in-sample data. Knowing this aspect of our models is important because it allows us to see how varying the types of inputs to our models change the ways in which our models overshoot or undershoot predictions. In Figure 3. It appears that both learners overshoot predictions, with positive errors, but it appears that using a larger leaf size aggregation, closer to 50, drives the Mean Percentage Error for a DTLearner model significantly closer to zero than for the RTLearner. This makes sense, because our decision tree learner is splitting based off the best value, which should theoretically drive our output in the right direction much more frequently. The RTLearner model appears to fluctuate very rapidly at leaf size intervals, with a much higher MPE average across the range from a leaf size of 0 to 50. This information is helpful in assessing the direction in which our models is performing, positive or negative, and not necessarily the degree to which it is performing overall.

The second metric is driven by the amount of time it takes to build the decision trees for varying leaf sizes on both of the learners. As we can see in Figure 4. The RTLearner takes 25 milliseconds initially to create its largest decision tree on an aggregated leaf node size of 1, whereas a DTLearner takes 175 milliseconds respectively. This is very useful information, as it should be noted that a DTLearner model has much more overhead in calculating and calling the absolute correlation method to determine the best split value. This appears to add to the execution time by a factor of 7 for both models when creating their largest trees on a completely overfit model of leaf size equal to 1. As both models begin to create trees on leaf sizes closer to 50, their graphs begin to approach a horizontal asymptote close to zero. However, DTLearner still appears to drive toward an asymptote above zero whereas a RTLearner model drives horizontally at a value of zero.