I. Introduction

In this assignment, I implemented and compared the performance of a Classic Decision Tree Learner and Random Tree Learner and Bagging /boost aggregating method on Istanbul data.

2. Learning Algorithms

2.1 Decision Tree Learner

Decision tree learning uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value.And it contains 5 elements: Factors, labels, nodes, root and leaves. The factors are the Xtrain columns for each instance, labels are the Ytrain data. Classic decision tree use a best feature to split the tree, and generate left-tree and right-tree, and recursive repeat this process until iterate all the nodes/leaves in the tree.

2.2 Random Tree Learner

Random tree learning use exactly same algorithm as Decision tree learning, except it randomly pick a factor as the split feature, work as the “best feature” in Classic decision tree. And compared to the classic tree, which use the mean/median value of all the instance for the “best feature”, Random tree use the mean value of the 2 randomly picked instances’ split feature.

2.3 Bagging/Boosting Aggregation

Bagging or Boosting Aggregation itself is not an algorithm/method for implementing decision tree. But it is an improvement of the existing decision tree algorithms. By using the same algorithm with training them on a different set of learner, it helps lower the error on each single decision tree model and also, it helps reduce the overfitting.

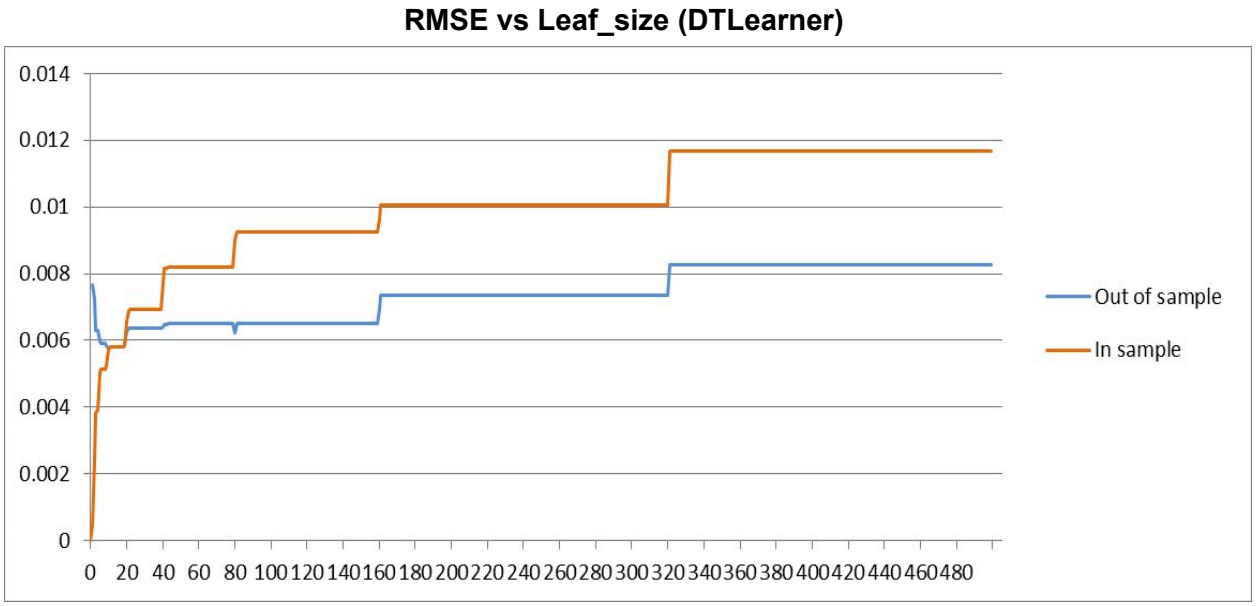
3. Experiments & Methodology

Decision Tree Learner, Random Tree Learner and Bagging were implemented in Python 2.7 as part of this project. After verifying that my implementation was correct using the Auto-Grader, I created new test scripts by following the template testlearner.py to setup the experiments. Based on my codes and testing, I evaluated three conditions listed in the requirements, the detailed explanation are as follows:

*3.1 Does overfitting occur with respect to leaf_size (for DTLearner)?*Which values of leaf size does overfitting occur?

Consider the dataset istanbul.csv with DTLearner, I use RMSE as the metric for assessing overfitting, change the leaf_size from 1 to 499. For each leaf size, I trained the DTLearner on the training data set, and then ran a query using the testing data set.

When plot the graph between the leaf_size and RMSE, I use the value of the test, and the graph is as follows:

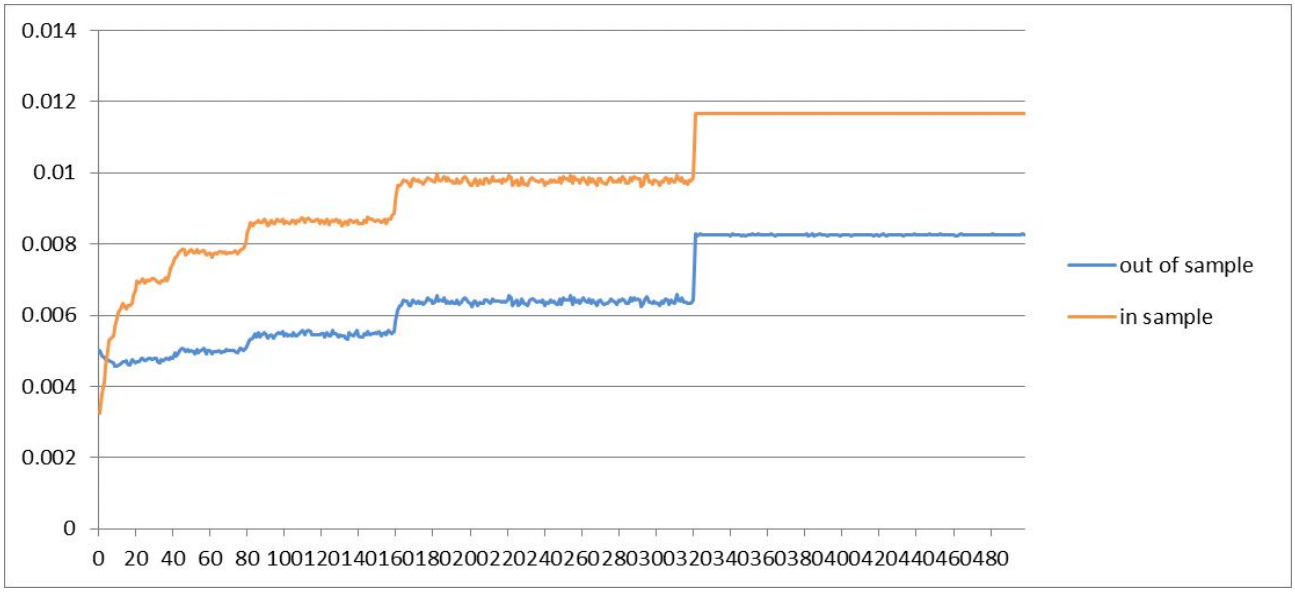


As the lectures states, overfitting occurs when the RMSE for the Out of Sample test data set starts to increase after a decreasing trend that follows the RMSE for In Sample test data set. From above graph, it shows that overfitting occurs for Leaf Sizes that are less than 20. Less than 20, even though the In-Sample RMSE continues to decrease, the Out-of-Sample RMSE starts to increase.

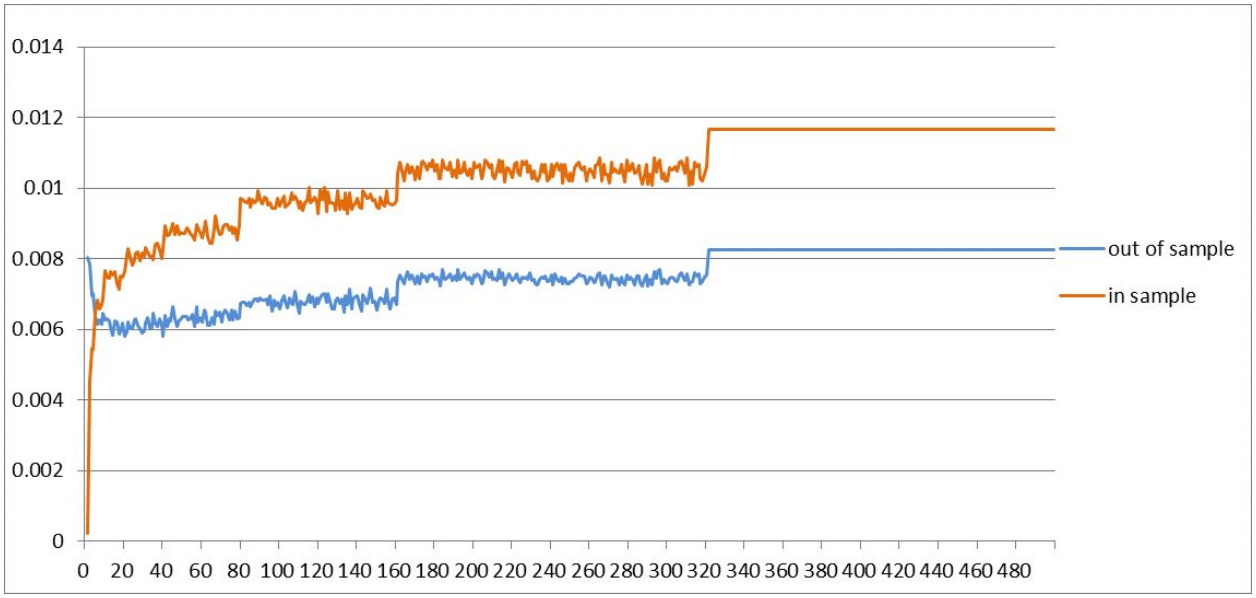
3.2 Can bagging reduce or eliminate overfitting with respect to leaf_size?
To answer this question, I write a testBagLearner.py, and use fixed number 20 as bags, and use vary leaf_size from 1 to 499, plus use RTLearner to evaluate. And also, I repeat this process totally 4 times, and use the mean value of RMSE to graph the bagging. In addition, since in the 3.1 test, I use DTLearner instead of RTlearner to test the leaf_size and overfitting occurance, I think it is better to test RTLearner to compare with RTLeaner with Bagging.

To do this, I also use same approach to just graph RTLearner for compare with RTLearner with Bagging, the two graphs are as follows:

RMSE vs Leaf_size (Bagging using RTLearner)

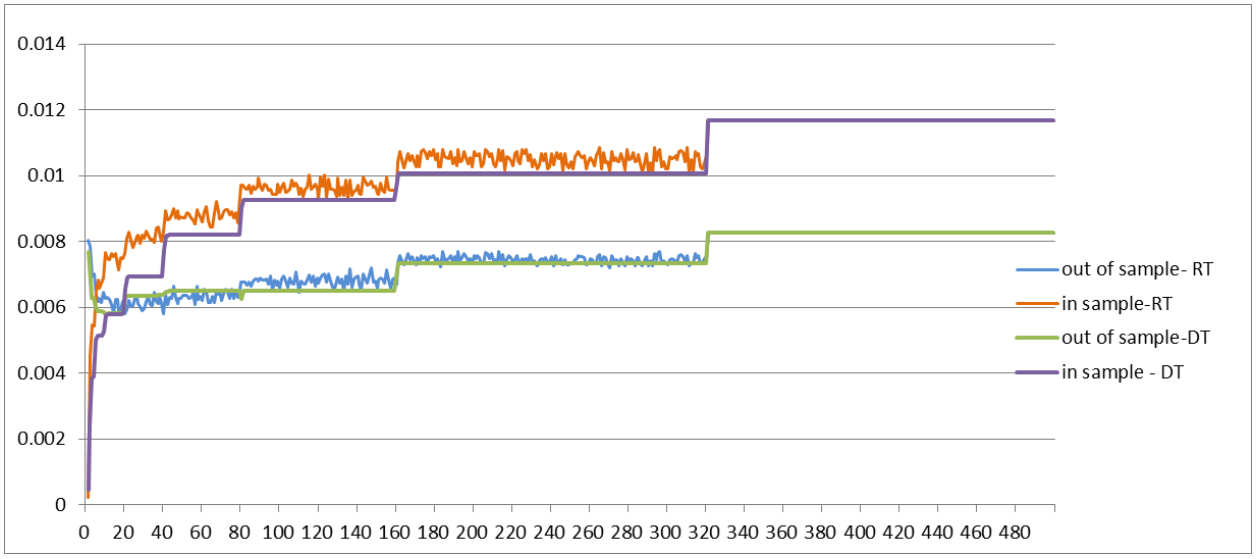


RMSE vs Leaf_size (RTLearner)



From the graph above, it's clear that contrary to the results from the 3.1 testing, Bagging, when applied to a Random Tree Learner is more resistant to overfitting than just using a Decision Tree Learner or Random Tree. It overall lower the RMSE when the leaf_size is less than 320. After 320, the out of sample RMSE value are the same as only using Random Treem but we can still find that bagging provide a slightly fluctuation after leaf_size over 320. And the bagging still shows a slight increase in the RMSE for the Out of Sample test data set as the Leaf Size decreases below 10, but the rate of the RMSE increase is drastically lower than the rate observed in just the Decision and Random Tree Learner. However, it does not eliminate it completely.

3.3 Quantitatively compare “classic” Decision Tree and Random Tree, in which way is one method better than other?
To answer this question, I combined my tests on RTLearner and DTLearner, similar to previous, I use the mean value of RMSE for RTLearners amonf 4 tests, and compare with DTLearner. The leaf_size is changing from 1 to 499 and I get the following graph:



Decision Tree overfitting start from around 20, and Random Tree ‘s overfitting start from 10. So, based on the overfitting, Decision tree perform better than Random tree.
And also by checking the RMSE, DTlearner’s general out of sample value and in sample value are lower than RTLearner’s. Which support the conclusion via overfitting point, that is Classic Decision performing better.

However, during the testings, I find that RTLearner actually run faster than DTLearner, which also states in the lectures, that Random tree is running faster since it randomly select split feature instead of computing the best feature.