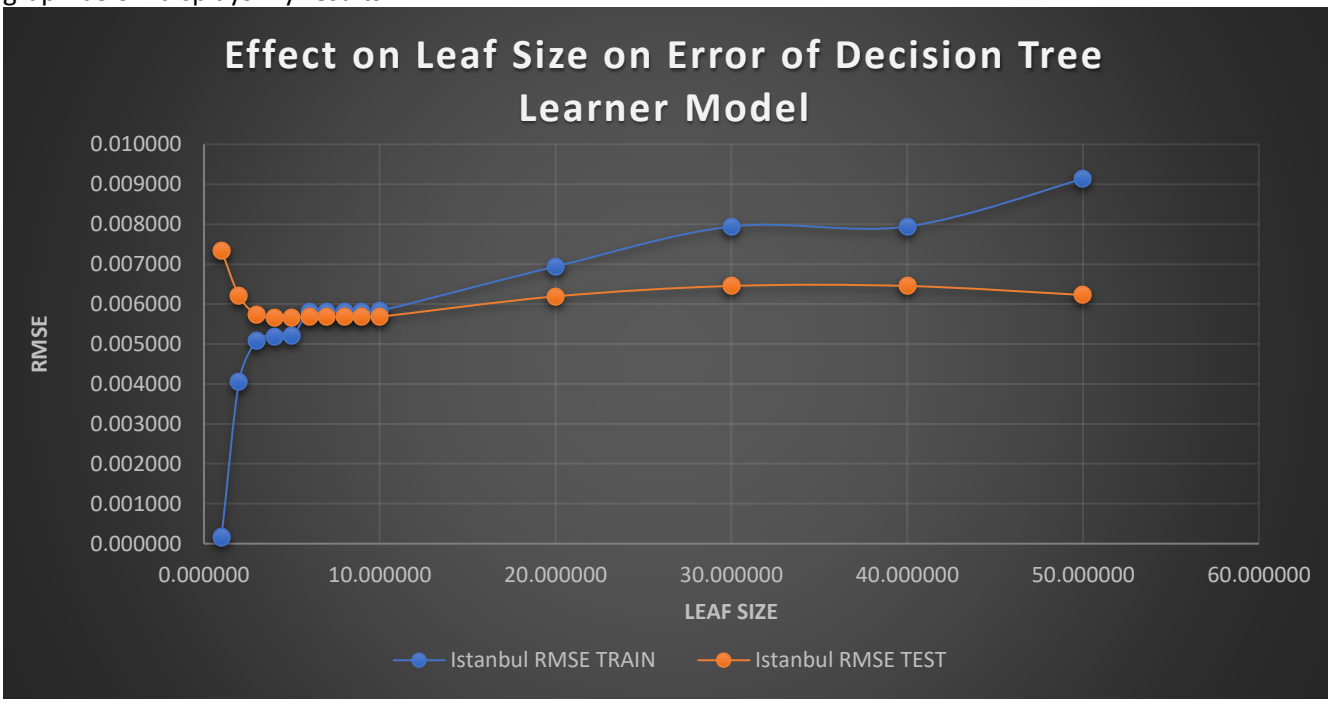
Assess Learners Report

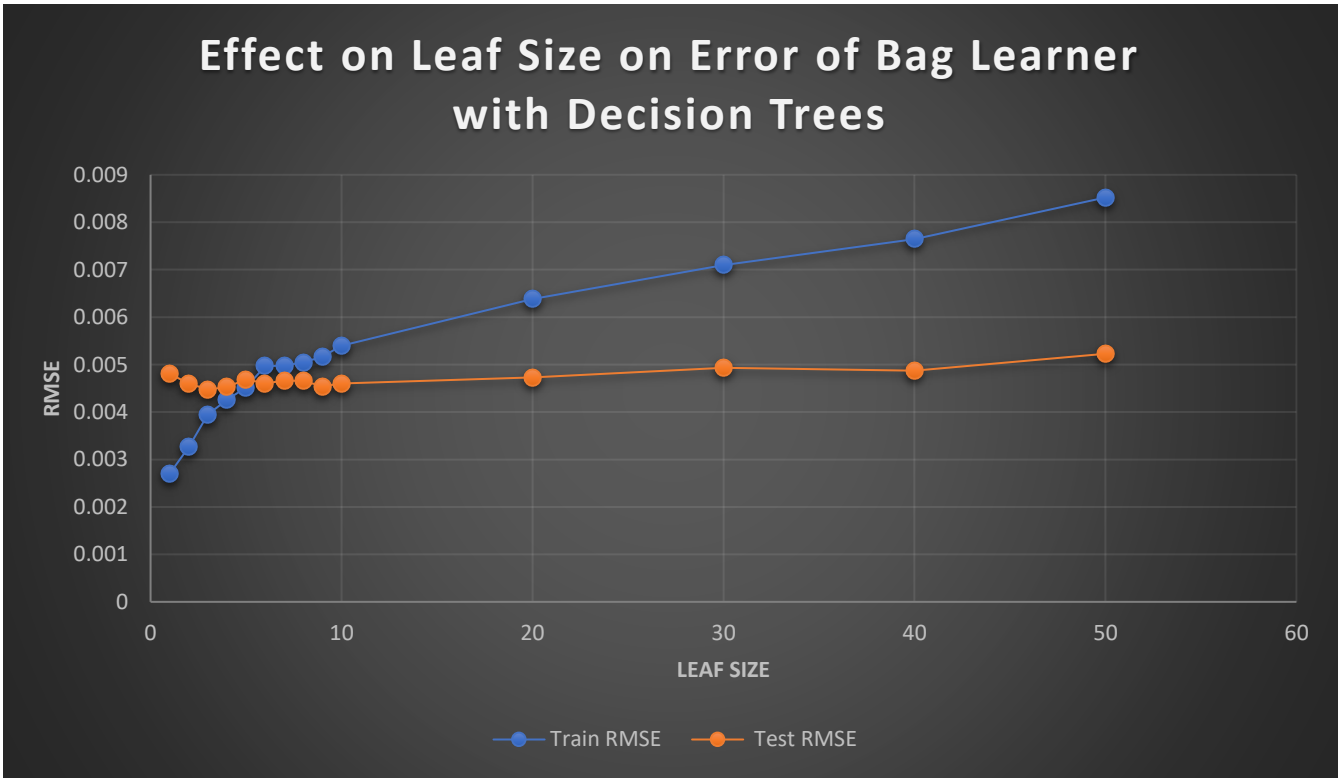
The purpose of this project was to implement and evaluate three different learning algorithms as Python classes: Decision Tree Learner, Random Tree Learner, and a Bootstrap Aggregating Learner. After implementing these classes, we used stock data from Istanbul, treating it as a regression problem and ignoring the time series feature. Overall, root mean square error, RMSE, is used as the default metric for evaluation and comparison between the learners, each also using varying hyperparameters. In other words, RMSE is used to quantitively describe how well a certain learner with certain hyperparameters performed and the RMSEs of each test variable is compared to other RMSEs values of other test variables to see which variables (learners and hyperparameters) perform better under certain conditions. A lower RMSE would indicate that a certain learner performed better than a learner with a higher RMSE as a low RMSE means that the actual answers were not far from the predicted values of the lower RMSE learner. In the following paragraphs, I will address three things that I tested regarding the learners: effect of changing leaf size in a decision tree learner on overfitting of the data, effect of bagging the first problem on overfitting of the data, and a comparison of the Decision Tree Leaner and Random Tree Learner using different numerical comparisons of the algorithms.

First, I tested the effect that varying the minimum leaf size in a decision tree learner would have on that learner overfitting the data. Since overfitting is the error in which the learner analysis of the training data results in a model that will not generalize well for data outside the training data (test data and future data to predict on), in terms of RMSE, this would be indicated by a low RMSE score for the training data and a higher RMSE score for the testing data. In a decision tree, a minimum leaf size indicates the minimum number of cases in that leaf. Too low of a leaf size means that a decision tree will split more trying to fit all the training data so each leaf only has a few points (atleast as many as the minimum leaf size), which could lead to overfitting and a model that is accurate for the training data and not as accurate for the testing data. Too high of a leaf size means that a decision tree will pack a lot of data points into few leaves, cause underfitting and a less predictive model. Therefore, to test the necessary effect, I ran my Decision Tree Learner on the Istanbul data 14 times, each time with a different leaf size, starting at 1, incrementing by 1 nine times, to get to 10, and then incrementing by 10 four times to get to 50. For each leaf size, I noted the RMSE of the training and testing data and the graph below displays my results:



What this graph shows is that there is overfitting occurring from leaf size 1 to 5. In this region, the RMSE of the Istanbul training data is less than the RMSE of the test data, significantly so especially in the first few leaf sizes. Lower leaf size values having overfitting makes sense because in this region, the learner tree is conforming to the testing data, which includes noise, and thus not building a well enough generalized model to use for prediction. After leaf size is greater than 5, from where leaf size is 6, the RMSE for the test data is less than the RMSE of the training data. So, if the graph is viewed from right to left, meaning starting at a high minimum leaf size (50) and working down, we notice that overfitting starts to occur at leaf size 6 and continues in each subsequently lower leaf size, getting worse (very low Train RMSE and very high Test RMSE- huge disparity) as the leaf size approaches 1. As the disparity is very significant from leaf size 1-3, that region is most likely where the overfitting is occurring.

Next, I tested whether bagging reduces, eliminates, or has no effect on overfitting with respect to minimum leaf size. Bagging, also known as Bootstrap Aggregation, is a learner which takes another learner and makes different bags each with the input learner and a data set that is randomly sampled from the initial data set. For this specific test, I used a Bootstrap Aggregation that takes in a Decision Learner, samples the same Istanbul dataset as described in the paragraph before, and makes 25 bags at each instance. I varied the leaf size of the instances similar to the previous test: I started at minimum leaf size equals 1, incremented by 1 nine times, to get to 10, and then incremented by 10 four times to get to 50. For each leaf size, I noted the RMSE of the training and testing data and the graph below displays my results:



This graph showed very similar results to the previous test. There is overfitting occurring from leaf size 1 to 5. In this region, the RMSE of the Istanbul training data is less than the RMSE of the test data, significantly so especially in the first few leaf sizes. After leaf size is greater than 5, from where leaf size

is 6, the RMSE for the test data is less than the RMSE of the training data. So, if the graph is viewed from right to left, meaning starting at a high minimum leaf size (50) and working down, we notice that overfitting starts to occur at leaf size 6 and continues in each subsequently lower leaf size, getting worse (low Train RMSE and high Test RMSE- huge disparity) as the leaf size approaches 1. However, the one difference between this experiment and the one before is that the disparity between the train RMSE and test RMSE is less when there is bagging; the data lines are closer in this experiment. And overall, the actual RMSE values for testing are numerically less than they were in the first experiment for every test leaf size, showing that bagging does help to make slightly more accurate model. Also, although bagging does not eliminate overfitting in the region of leaf size 1-5, as RMSE of the training data is lower than the RMSE of the test data, since the disparity between the two is lowered, and there is better (lower) overall test RMSE values with bagging, it implies that bagging does reduce overfitting. This does bring up a question of whether the actual number of bags affects the overfitting of data and if I were to repeat this experiment, I would add an extra section where I would vary the number of bags to test that effect.

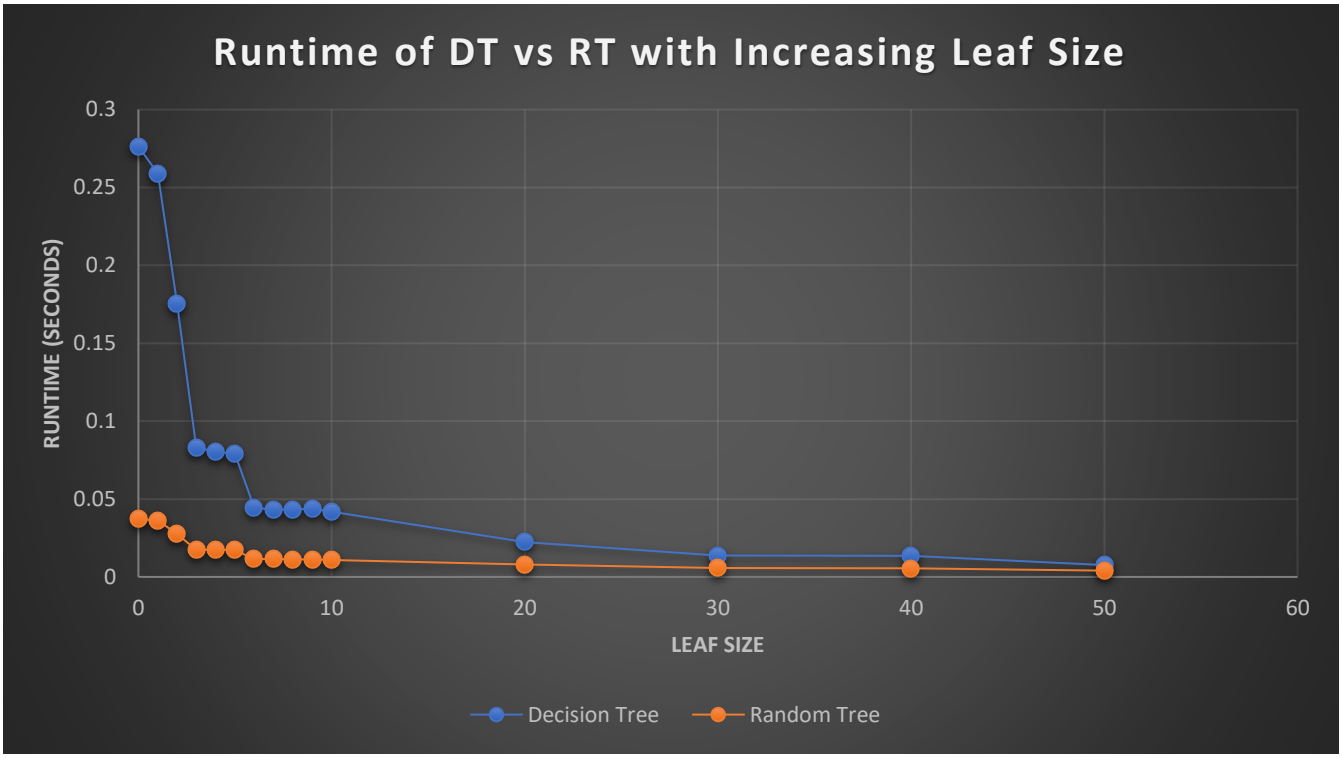
For my last test, I did two numerical comparisons of the Random Learner and the Decision Tree learner. First, I compared the size of each learner's tree, with a varying leaf size, to see if each learner has a different number of nodes. To do this, I reran the same process as I did for the first experiment: I ran my Decision Tree Learner on the Istanbul data 14 times, each time with a different leaf size, starting at minimum leaf size equals 1, incrementing by 1 nine times, to get to 10, and then incrementing by 10 four times to get to 50. First, I trained on the 60% percent of the data that was defaulted by the assignment to be training set. After training on the data, I calculated the size of the tree, or how many nodes there are in that tree. I then repeated this for each hyperparameter and each learner. This was the following result:

	Number of Nodes for Each Tree	
Leaf Size	DT Nodes	RT Nodes
1	633	637
2	257	255
3	131	131
4	127	127
5	123	123
6	63	65
7	63	63
8	63	63
9	63	63
10	61	55
20	29	31
30	15	15
40	15	15
50	7	7

So overall, each tree expands almost the same number of nodes, with only a small difference when leaf size is less than 3. This makes sense because although each choose a different feature to split on, one chooses based on correlation, one chooses randomly, both are going to split as many times as they can within the given constraints of the minimum leaf size. Also as a general trend, you can see that as the

minimum leaf size goes up, the number of nodes goes down and this makes sense, because as explained earlier, a higher leaf size means a tree will pack a lot of data points into fewer leaves, causing less splits and thus, a smaller tree.

For my second comparison, I compared the runtime of each learner, with a varying leaf size. To do this, I reran the same process as I did for the first experiment: I ran my Decision Tree Learner on the Istanbul data 14 times, each time with a different leaf size, starting at minimum leaf size equals 1, incrementing by 1 nine times, to get to 10, and then incrementing by 10 four times to get to 50. Before running the algorithm on each leaf size, I calculated the current time as starting time, using Python's inbuilt time library, and then after the algorithm ran for that specific leaf size, I got the time again as the end time. Subtracting those two values gave me the overall runtime, for training and testing, of that algorithm for that specific leaf size. I repeated this whole process again with Random Tree Learner, using the same varying leaf size hyperparameters to get the runtime of that algorithm. The below chart shows my results:



So overall, the Decision Tree is definitely slower than the Random Tree as the blue line (Decision Tree runtime) is always above the orange line (Random Tree runtime). This makes sense as the Decision Tree Learner must iterate the columns of the training data and calculate each feature columns correlation with the labels column to figure out the best feature to split on. For a Random Tree Learner, this process is randomized in that a random feature column is picked and then split upon, thus, not having to iterate multiple columns of data each time, the Random Tree Learner is significantly faster. Also, analyzing the trend, you can see that the first 5 leaf sizes have higher runtime than the rest of the leaf sizes, for both trees. This can be tied to the previous observation of overfitting within that region as overfitting means that the tree structure was conforming to the data with more splits, thus causing higher runtimes. Eventually, after leaf size crosses 20, you can see the runtime of both learners starts to plateau and this can be attributed to a higher minimum leaf size meaning more data points per leaf and thus, less splitting and a faster runtime.