# Machine Learning for Trading
# Project 3 Assess Learners

Author Name
lzhang699@gatech.edu

***Abstract***— In this project, 4 CART learners as regression learners are built using Python. The experiments focus on the Decision Tree, Random Tree and Bagging. Statistical metrics such as RSME, MAE, etc. are used to measure the corresponding performances among these models. The data used for the calculation of the metrics are returns of multiple worldwide indexes and the goal is to predict the return for MSCI Emerging Markets Index

## 1 INTRODUCTION

Decision Tree and Random Tree are the two major models utilized in this report to predict future Y values based on historical X factors and Y values. Models are firstly trained with 60% of the dataset. They are then validated with the same in-sample data and the rest 40% out of sample data. The dataset is a n column matrix with 0th...n-1th X factor columns and nth Y value column. Coefficient of correlation, Root Mean Square Error, Mean Absolute Error and Time to Build are the metrics calculated and utilized in the experiments below to build and compare performances of the models. The bagging technique is also implemented in order to observe its impact on overfitting. It is expected that without bagging, Decision tree should provide a better performance overall due to its Coefficient of correlation based modeling algorithm. By using bagging, overfitting should be reduced.

## 2 METHODS

In this report, the Decision Tree is built as a binary tree using numpy.ndarray (e.g. [root. left_sub_tree, right_sub_tree]) with a recursive method. At each recursion stack, the Coefficient of Correlation is calculated between each X factor column against the Y value column. The X factor column with largest coefficient is then selected as the feature column, of which the median of all rows is used to split the given matrix by rows. Rows with smaller or equal feature column value are passed back into the recursive function to build the left_sub_tree and the rest rows will be passed to build the right_sub_tree. In the case when all rows fall into one side of the median, the mean will be used instead as the split value.

The Random Tree follows a similar building process but the feature column is selected randomly instead.

The BagLearner is implemented with the bagging technique. It initiates a group of the same learners, each acts as a bag. At each bag, a learner is trained on the same dataset which is sampled with replacement. The output is the mean of all outputs from each bagged learner.

The dataset is first split into 60% training and 40% testing set. Each above-mentioned model is trained with a training set and then tested with a testing set. Leaf size is used as a variant from 1 - 50 that controls fitting of tree nodes against data points. A smaller Leaf Size leads to creation of more tree nodes and more closely fitted nodes whereas a larger Leaf Size leads to a more loosely fitted tree. In order to better observe the trend in each metric, each model is trained and tested for >=10 times in order to obtain the mean value. The entire dataset is shuffled (sampling without replacement) between each run so that the training and testing sets include different rows at each run.

## 3 DISCUSSION

### 3.1 Experiment 1

In this experiment, 2 charts are generated. Figure 1 is created with the default dataset(unshuffled) and Figure 2 is created with 10 runs average result (shuffled in-between each run). Leaf size ranges from 1 - 50. From both figures, we can generally see that the overfitting in out of sample results starts to occur ~5 leaves since the RMSE increases drastically as the leaf number decreases further from 5. From Figure 1 out of sample results exhibits a lower RMSE than that of in-sample results as the Leaf Size increases. However with 10 runs, this strange discrepancy is eliminated as shown in Figure 2. On the other hand, the in-sample results do not show any sign of overfitting since the training data are used for testing.

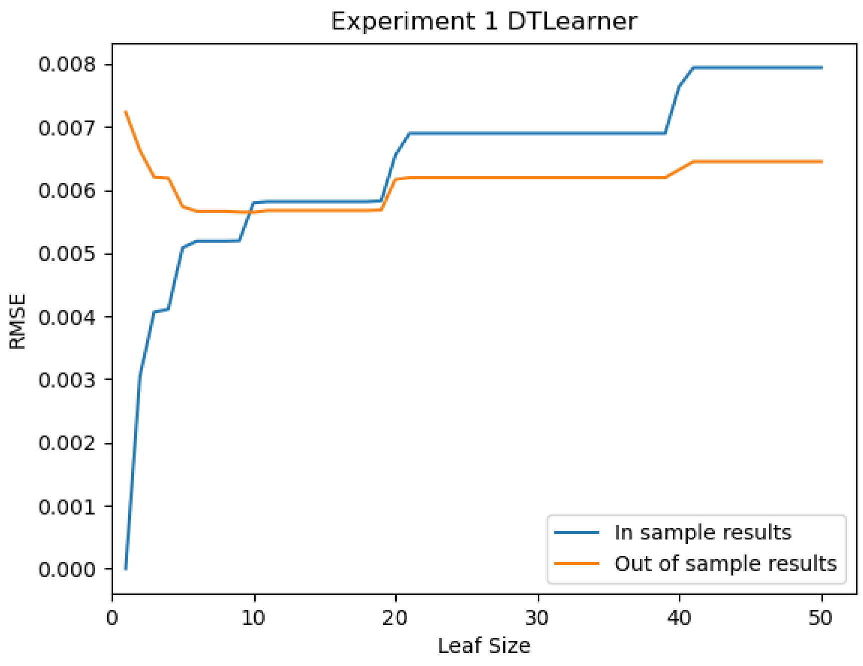

***Figure 1***—Experiment 1 DTLearner single run

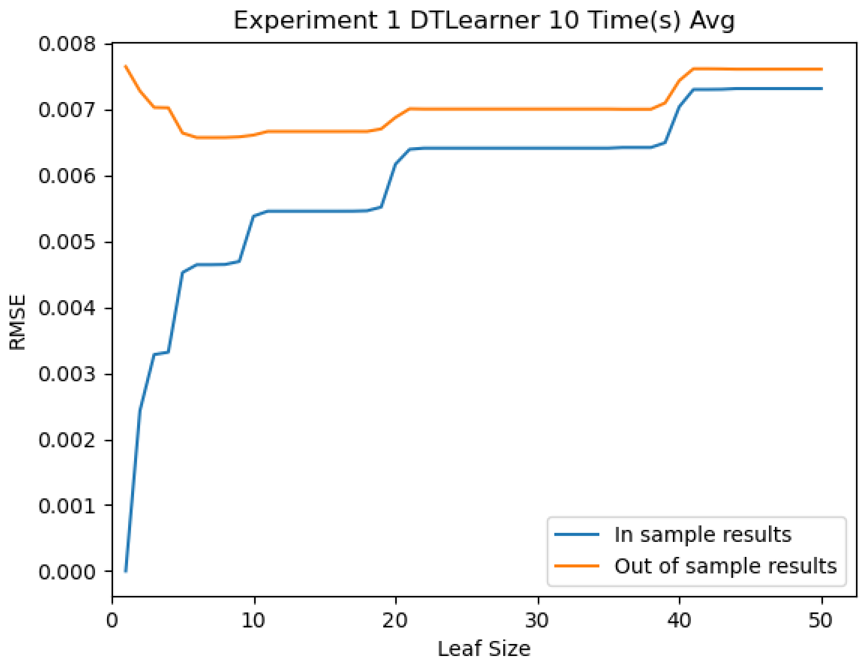

***Figure 1***—Experiment 1 DTLearner 10 runs

### 3.2 Experiment 2

A BagLearner with each bag formed by a Decision Tree is used to build the model in this experiment. The BagLearner has a constant bag number 20 and Leaf Size again ranges from 1 - 50. Contrarily to Figure 1 & 2, Figure 3 does not reveal a deterministic point where overfitting occurs. The RMSE at Leaf Size 5 seems to be a minimum. It is obvious that the overfitting has been reduced by using the bagging technique. However whether it is eliminated or not is still debatable. The size of the dataset can be a variant as the one used here has 536 items vs Leaf Size 1 - 50. Bag size is also treated as a constant in this experiment but it is another potential variant.

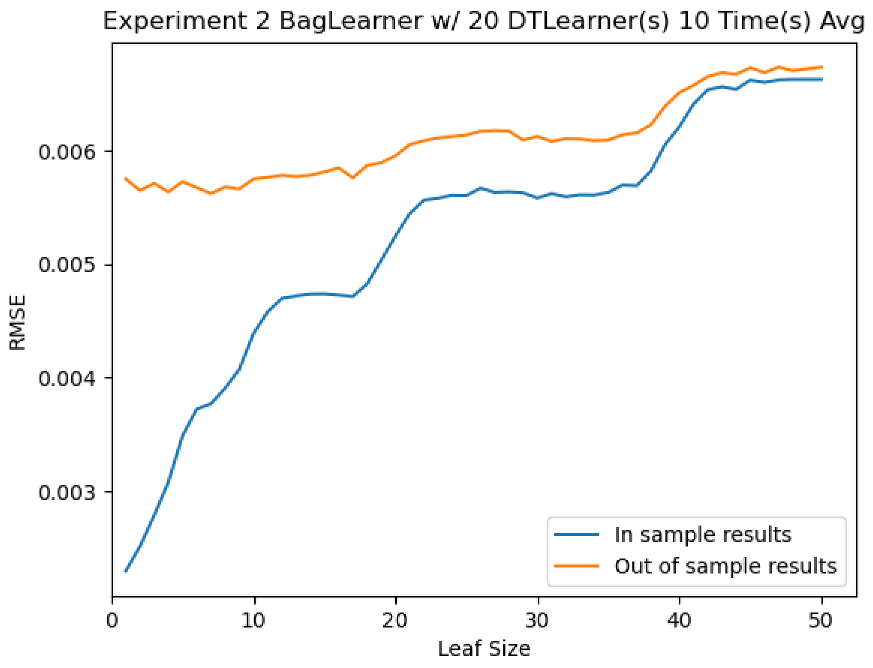

*Figure 3*—Experiment 2 BagLearner with DTLearner 10 runs

### 3.3 Experiment 3

In this experiment, two different metrics are used - MAE (Mean Absolute Error) and T2B (Time To Build).

$$MAE = \sum^{n} | y_{predict} - y_{observe} | / n$$

The main advantage of using MAE over RMSE is that it presents a closer interpretation. RMSE penalizes large errors more due to its given formula. Here in Figure 4, we can see that the MAE between DTLearner and RTLearner on out of sample results are in fact not too far apart. If we look closely at the starting point of overfitting at ~5 leaves, The MAE difference is actually extremely small.

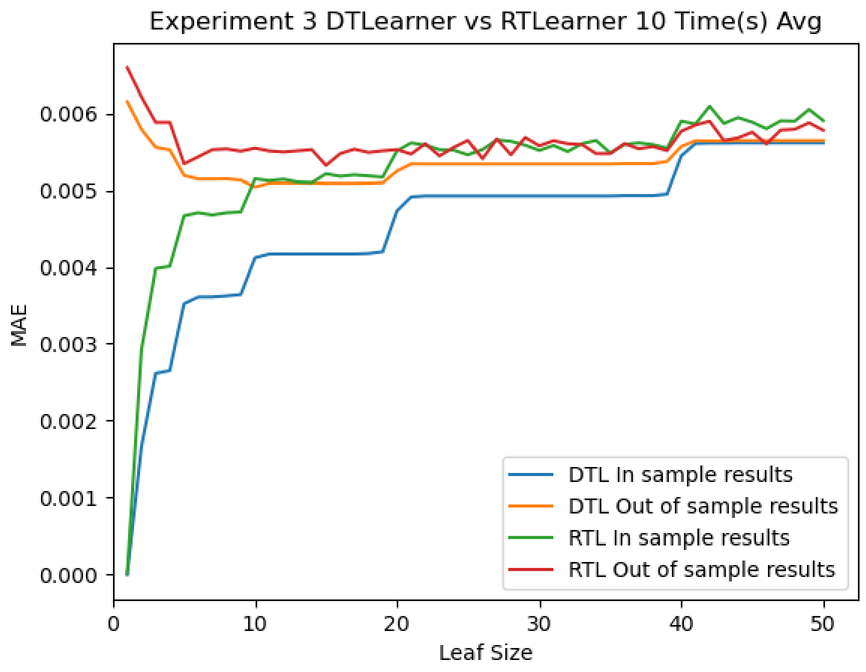

***Figure 4***—Experiment 3 DTLearner vs RTLearner 10 runs

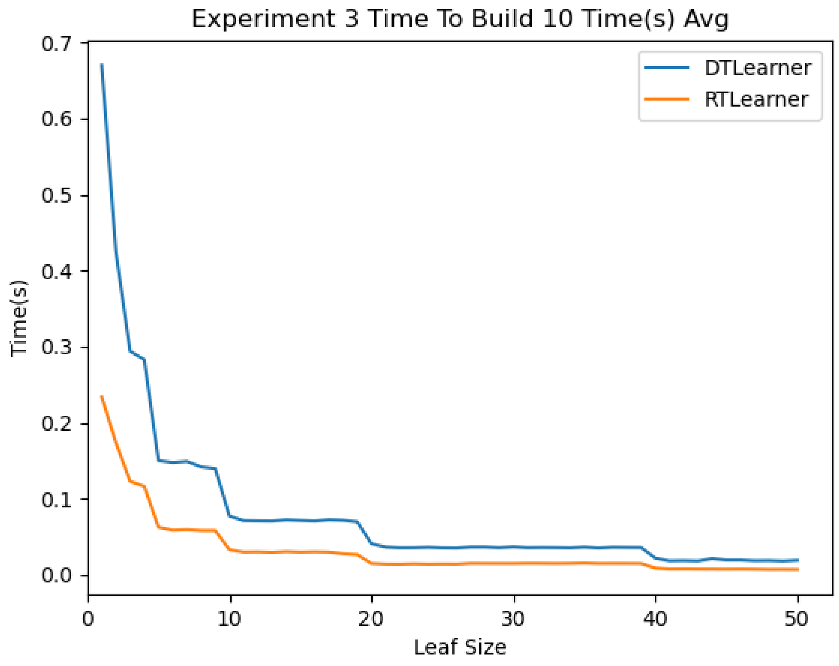

***Figure 5***—Experiment 3 Time To Build DTLeaner vs RTLearner 10 runs

This slightly better performance comes at a cost of almost 3 times the T2B as shown in Figure 5. For the given dataset, a RTLearner takes ~0.05s to build whereas a DTLearner takes ~0.15s. Therefore, if precision is a big factor in a model, then the DTLearner is better. If a model needs to be built quickly, then the RTLearner is better.

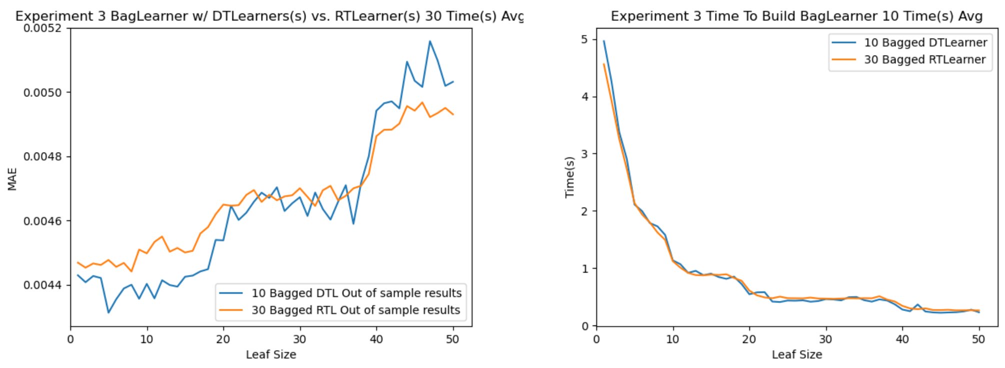

***Figure 6***—Bagged DTLearner & RTLearner MAE vs T2B.

The above Figure 6 shows the out of sample MAE of a BagLeaner with 30 bags of DTLearner and a BagLearner with 10 bags of RTLearner, and their T2B. With bagging, the MAE is reduced from ~0.0055 in Figure 4 to ~0.0045 in Figure 6. Overfitting still seems to be present for the 10 DTLearners occurring at ~5 leaves. The MAE difference is present and very noticeable whereas the T2B is almost the same (10 DTLearners T2B = 30 RTLearners T2B). Therefore with bagging, DTLearner is still superior in terms of precision.

## 4 Summary

Both DTLearner and RTLearner are able to produce predictions with low error (RMSE, MAE). This is mostly due to the fact that the MSCI Emerging Market Index as the Y value captures large and mid cap representations and X factors are all global indexes. Their prices are already inherently correlated. This is also another reason why RTLearner seems to perform almost as well as the DTLeaner does.

For further investigation, different types and amounts of dataset can be used to further compare the performance of DTLearner and RTLearner.

## 5 References

1. https://www.msci.com/documents/10199/c0db0a48-01f2-4ba9-ad01-226fd5678111
2. https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d
3. https://towardsdatascience.com/r%C2%B2-or-r%C2%B2-when-to-use-what-4968eee68ed3
4. https://en.wikipedia.org/wiki/Mean_absolute_error