CS7646 Spring 2022 Project 3 Assess Learners

Fung Yi Yuen

fyuen3@gatech.edu

Abstract—Machine learning algorithms help to analyze numerous stock prices and predict trends. This project demonstrates the implementation of four supervised learning machine learning algorithms using Python. The algorithms will train models by fitting in historical stock data and then use the models to predict future stock data. The training performance and prediction performance will be evaluated by varying the algorithms' hyperparameters in several experiments.

1. INTRODUCTION

This project focuses on four supervised learning algorithms which belong to Classification and Regression Tree (CARTs). They are Decision Tree learner, Random Tree learner, Bootstrap Aggregating learner ('bag learner') and Insane Learner.

There are features and corresponding split values in a decision tree. Entropy, Correlation or Gini index can be used to determine the best feature. Decision Tree learner (DT learner) in this project uses correlation to determine the best feature index and then calculates the median to determine the splitting value.

Unlike DT learner, Random Tree learner (RT learner) does not need to determine the best feature index. It randomly picks a feature index to split on.

Bag learner is an ensemble learner which can accept any learner. In this project the bag learner uses the same learning algorithm (e.g. twenty DT learners or ten RT learners) and trains each learner with different subsets of the training data.

Insane learner contains 20 bag learners where each bag learner contains 20 linear regression learners.

Here are the initial hypotheses of implementing the four learners:

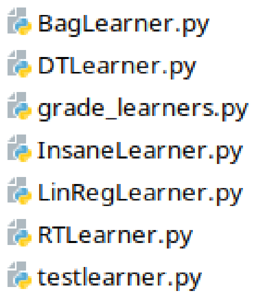
- Assume the future stock price is based on historical stock price, i.e. no sentimental factors such as investor behavior added into calculation.

- Assume the change of a company's fundamentals has no effect on its stock price
- Assume the change of the directors' shares of a company has no effect on the company's stock price.

2. Methods

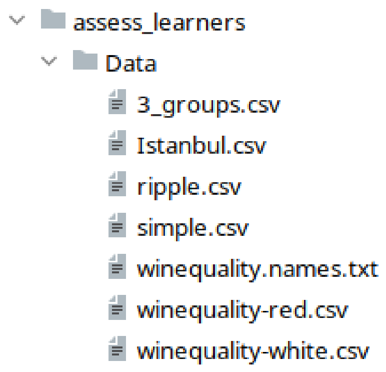
To perform the experiments in this project, the assess_learners folder needs to be structured as follows[1]:

- All learners (.py files) put in the assess_learners directory:



```
BagLearner.py
DTLearner.py
grade_learners.py
InsaneLearner.py
LinRegLearner.py
RTLearner.py
testlearner.py
```

- All training and testing data (.csv files) put in the assess_learners/Data directory:



```
assess_learners
  Data
    3_groups.csv
    Istanbul.csv
    ripple.csv
    simple.csv
    winequality.names.txt
    winequality-red.csv
    winequality-white.csv
```

- Open terminal, go to assess_learners directory and run this command: PYTHONPATH=../:. python testlearner.py Data/Istanbul.csv
- Change Istanbul.csv to any other csv file for your own training and testing purpose.

[1]Assume your local environment has been set up according to https://lucylabs.gatech.edu/ml4t/spring2022/local-environment/

3. Discussion

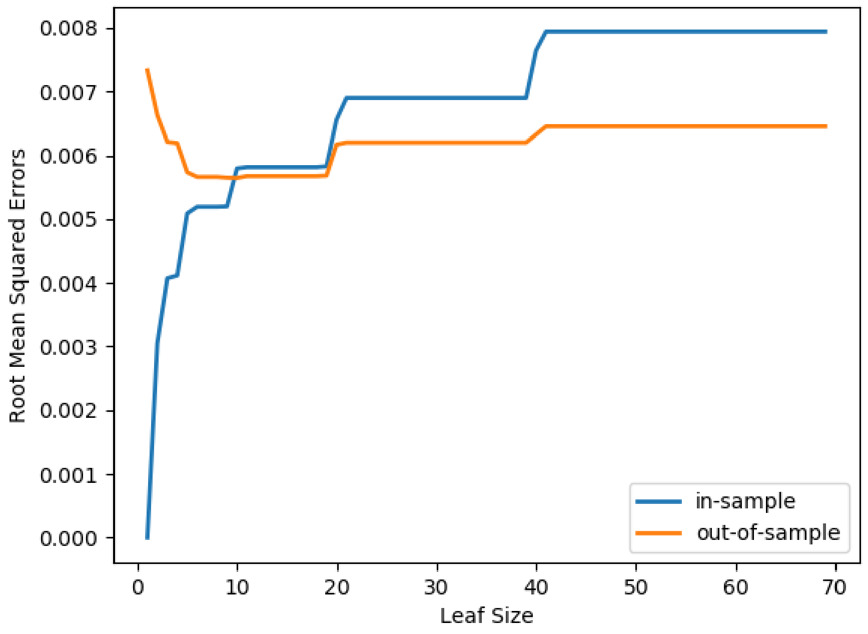
3.1 Experiment 1

- Does overfitting occur with respect to leaf_size?
- For which values of leaf_size does overfitting occur? Indicate the starting point and the direction of overfitting. Support your answer in the discussion or analysis. Use RMSE as your metric for assessing overfitting.

Yes, overfitting occurs when the leaf_size goes small. By plotting a graph of RMSE vs leaf size (Figure 1 below), RMSE of in-sample drops sharply while RMSE of out-of-sample increases significantly when leaf size is around 5 or smaller. Overfitting occurs at the point when in-sample REMS drops but out-of-sample REMS rises.

As leaf size increases (larger than 5) both in-sample and out-of-sample REMSs become correlated.

Figure 1: RMSE of Decision Tree Learner with different leaf size



3.2 Experiment 2

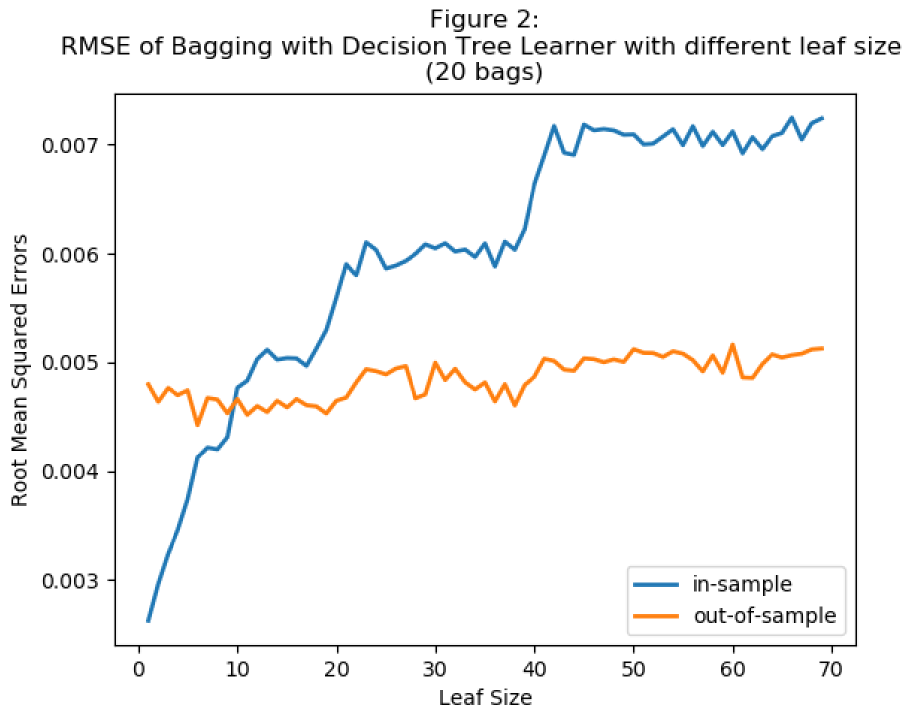
- Can bagging reduce overfitting with respect to leaf_size?

- Can bagging eliminate overfitting with respect to leaf_size?

Bagging can reduce overfitting for out-of-sample data. In experiment 2, twenty bags of DT learners were used to make a bag learner. A graph of RMSE vs leaf size is plotted (Figure 2).

The in-sample RMSE is similar to the one plotted in experiment 1. When the leaf size is around size 6 or smaller, in-sample RMSE drops significantly which indicates overfitting occurs in in-sample data.

However, the out-of-sample RMSE is much lower than the one in experiment 1 when the leaf size is around 6 or smaller. Figure 2 shows out-of-sample RMSE vibrates steadily below 0.005 instead of rising significantly like the one in experiment 1. Thus bagging helps to reduce overfitting in out-of-sample data.



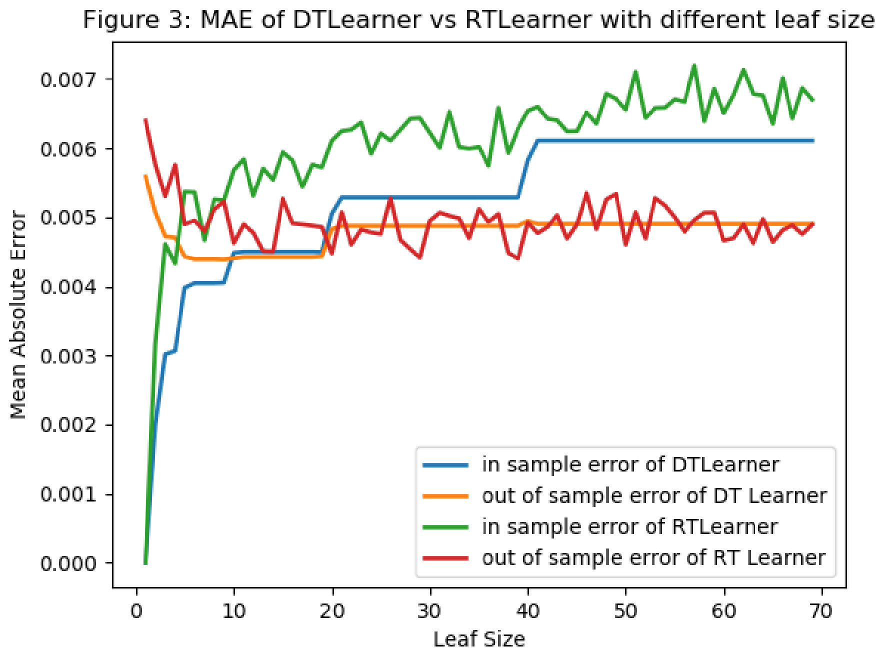
3.3 Experiment 3

Quantitatively compare “classic” decision trees (DTLearner) versus random trees (RTLearner).

- In which ways is one method better than the other?

- Which learner had better performance (based on your selected measures) and why do you think that was the case?
- Is one learner likely to always be superior to another (why or why not)?

Experiment 3 uses Mean Absolute Error (MAE) and Absolute Percentage Error (MAPE) as measures to compare the performance of DTLearner and RTLearner.

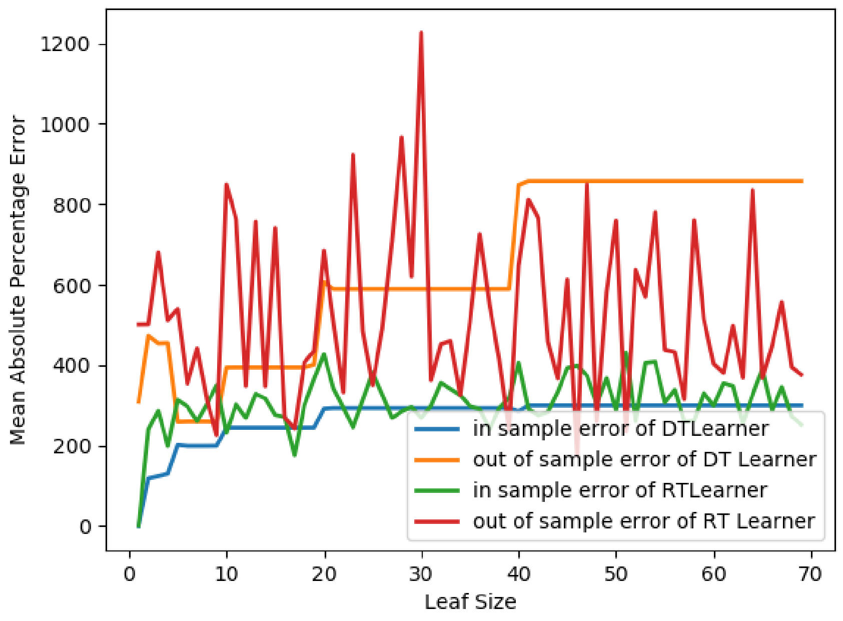


By plotting a graph of MAE versus leaf size (Figure 3) for DTLearner and RTLearner (for both in-sample and out-of-sample data), we can see overfitting occurs in both learners when leaf size is around 6 as both learners' in-sample errors drop significantly while out-of-sample errors rise when the leaf size goes down to zero.

Comparing the in-sample error of both learners (for leaf size > 6), DTLearner has a lower MAE than RTLearner (see the blue line is below the green line). DTLearner also has a more steady line than RTLearner. Since RTLearner randomly chooses a feature to split on instead of calculating correlation to choose the best feature as a split value, such randomness makes RTLearner’s graph more volatile. Thus, DTLearner performs better than RTLearner for in-sample data due to DTLearner’s lower MAE and more steady MAE value than RTLearner.

Comparing the out-of-sample error for both learners (for leaf size > 6), DTLearner has lower MAE than RTLearner at the beginning when leaf size is between 6 to 20 (see the orange line lower than the red line). As the leaf size goes larger than 20, DTLearner's MAE becomes stable at around 0.0048 while RTLearner's MAE fluctuates around 0.004 to 0.005 which is close to DTLearner's error value (0.0048). We can say DTLearner performs better than RTLearner when leaf size is between 6 to 20. But for leaf sizes larger than 20, both learners perform more or less the same.

Figure 4: MAPE of DTLearner vs RTLearner with different leaf size



By plotting a graph of MAPE versus leaf size (Figure 4) for DTLearner and RTLearner (for both in-sample and out-of-sample data), we can see overfitting occurs at leaf size near 6 as the error value of in-sample drops while the error value of out-of-sample increase for both DTLearner and RTLearner.

Comparing the in-sample error of both learners (for leaf size > 6), DTLearner (the blue line) has a more steady error value than RTLearner (green line). RTLearner's MAPE value keeps fluctuating (sometimes higher than DTLearner's error value while sometimes lower). Thus we cannot tell which learner performs better for in-sample data.

Comparing the out-of-sample error of both learners (for leaf size > 6), DTLearner's MAPE value (orange line) rises steadily as the leaf size becomes larger while RTLearner's MAPE value (red line) fluctuates in a large range. However, when leaf size is larger than 40, RTLearner's error value is always lower than DTLearner's error value. We might say RTLearner performs better when leaf size larger than 40. But due to the highly volatile MAPE of RTLearner, we cannot tell which learner always performs better for out-of-sample data.

To summarize the MAE and MAPE metrics analysis, there is no certain way to tell one learner is always superior to another. Both DTLearner's and RTLearner's performances depend on multiple factors e.g. leaf size, sample data, error metrics etc. Sometimes DTLearner performs better in a specific condition while sometimes both performances are more or less the same.

4. Summary

To conclude, experiment 1 shows overfitting occurs when leaf size is smaller than a certain value (size 5 in this case for DTLearner). Experiment 2 shows bagging can reduce overfitting by putting 20 DTLearners in a learner. Experiment 3 shows no single learner is always superior to another since a learner's performance depends on factors like leaf size, sample data and error metric. This could also be the reason why bagging is used since combining multiple learners into one can improve accuracy and reduce overfitting.