Project 3: Assess Leaners
CS 7646

Aarushi Gupta
agupta857@gatech.edu

***Abstract*—This project implements four types of regression learning algorithms using python - Decision Algorithm, Random Tree Algorithm, Bag Learner and Insane Learner. These algorithms are then evaluated to understand the difference between implemented learners. The comparison is drawn in terms of overfitting and root mean squared error. The data used to train the algorithm is static and not time order specific. Finally, some results are shared for the experiments run on the implemented algorithms.**

1. INTRODUCTION

The machine learning algorithms have various factors which helps in deciding which algorithm will fit better for what type of data, under what circumstances. The secret to the performance of specific algorithm lies in the understanding of its implementation. Therefore, the project required students to implement and evaluate the four regression learning algorithms - Decision Tree, Random tree, a Bootstrap Aggregating learner, and an Insane Learner. These learners are then tested and evaluated by performing three experiments. In each experiment charts are drawn using the final results along with a brief description about the conclusions that can be drawn from those charts.

The first two experiments are modelled around Decision Tree, however functions are parameterised accordingly to support any arbitrary learner. During evaluation of the third experiment three metrics have been developed to compare the decision and random tree learners - Time, Space and Precision Count.

2. METHODS

The project requires python installed with packages - numpy, matplotlib, scipy, time, math. To replicate the project results keep Data folder, testlearner.py and all the learners python file at same location. The data used should be static and

kept in Data folder only. Istanbul.csv data is used here which contains 536 records with 8 features. To repeat the experiments run main function of testlearner.py file from any IDE or run the command PYTHONPATH=../:. python testlearner.py Data/Istanbul.csv. Any data file can used in place of Istanbul.csv. This should produce the below shown experiment figures.

3. DISCUSSIONS

This section will discuss the three experiments performed to asses the implemented learners. The data is split into training and test dataset by a ration of 60:40.

1.1. Experiment 1

For this experiment, the learner decision tree was trained for leaf size varying from 1 to 100. And the root mean squared error for the predicted results - in sample and out sample data, were plotted against the leaf size.

Figure 1 shows the chart obtained from the experiment and can be used to check for overfitting. The right graph (magnified) shows that **overfitting exists near leaf size around 9** (close to 10). The conclusion is drawn due to the decreasing out-sample rmse and increasing in-sample rmse. Any such situation indicates overfitting. Therefore, it can be safely stated that high overfitting is observed with a lower leaf size, that is, **overfitting is inversely proportional to leaf size**. Overfitting occurs in the region lower than the leaf size 10 (approx. 9.8). The results are supported by the graphs in figure below.

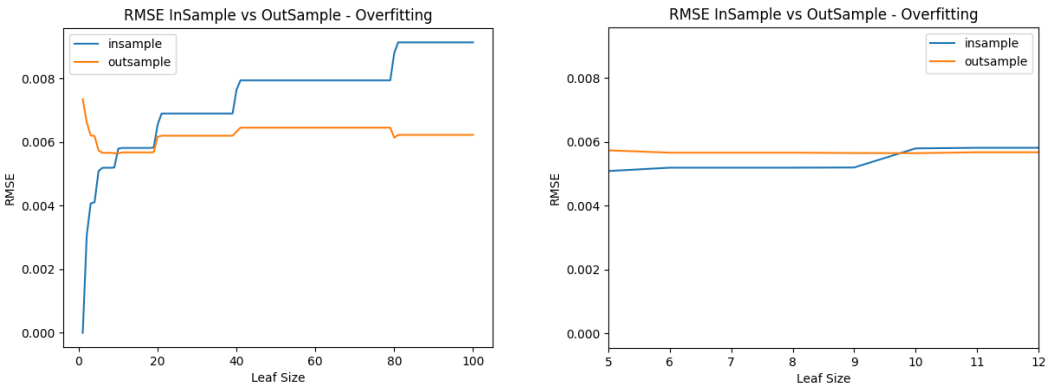

Figure 1—Experiment 1, Plot root mean squared error for in sample and out sample using Decision tree Algorithm.

1.2. Experiment 2

This experiment is an extension to experiment 1. It uses Bagging algorithm on 20 decision tree instances, that is 20 bags were used with decision tree learners and training data sampled with replacement. Varying the leaf size, we have tried to capture the effect of bag learner on overfitting. Figure 2 shows the plot of new rmse for in-sample and out-sample results predicted.

Bag learner is supposed to reduce the overfitting as 20 learner instance nullify each other's bias. **However, the bag learner has failed to remove overfitting completely,** but if we see closely it has reduced the region where overfitting exists. For the **leaf size around 9.5 (below 10 but closer to 9)** the overfitting exists and contains an inverse proportionality to the leaf size.

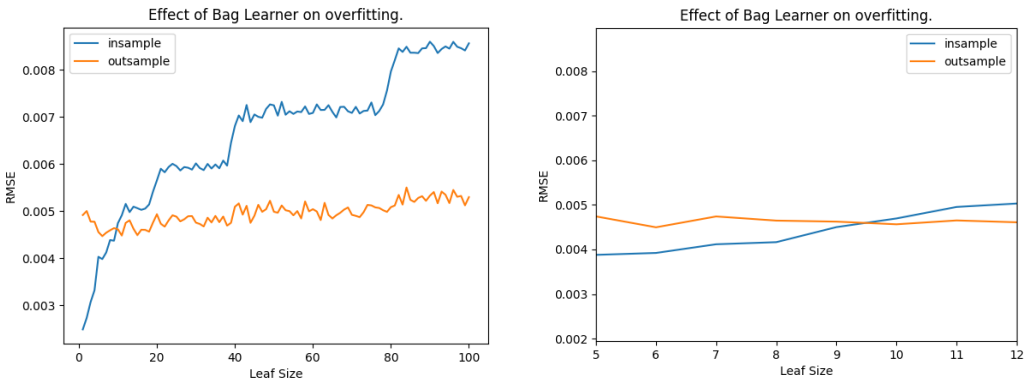

Figure 2—Experiment 2, Plot root mean squared error for in sample and out sample using 20 Decision tree instances for Bag Learner.

From the Figure 1 (right) and Figure 2 (right) it can be supported that the starting point for overfitting area has shifted a bit towards left reducing the region of overfitting. The shift occurred from 9.8 to 9.5 (approximately). Hence, using the graph plots an inference can be drawn that the bagger has reduced the overfitting for some area.

1.3. Experiment 3

This experiment compares classic decision tree with random tree by the means of three new metrics developed to analyse the difference in performance.

1.3.1. Time Taken

This metric measure time that the respective algorithms took while executing - learning (fig 3 left) and querying (fig 3 right).
Firstly, on focusing "training time" the graph plot shows that decision tree has taken much longer to train that random tree. This difference of time gradually decreases with the increasing leaf size, in fact, becomes almost equal after the leaf size 40. With smaller leaf size **decision tree takes 7 times longer to train when leaf size is low**. Which might have a huge effect if the data size is much larger. Therefore, it can be safely concluded that **for faster training random tree is recommended for lower leaf size.**

Secondly, "query time" is calculated in nano-seconds and shows a minute difference in time taken to query the same data. The graph plots show that random tree takes longer to query which is reasonable as random tree is not built by selecting best feature for every node. This might have huge implications when the data becomes larger when querying the data occurs on bigger scale. Therefore, it can be safely concluded that **although random tree takes less time to train but longer than decision tree to query the data (nanoseconds difference).**

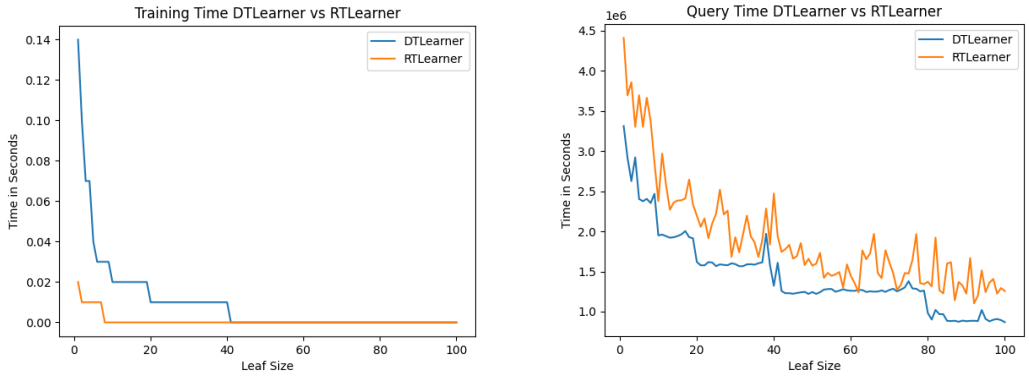

Figure 3—Experiment 3, Plot time taken by Decision tree and Random tree to learn data (in seconds, left) and query data (in nanoseconds, right)

1.3.2. Precision Count

This is a little tricky metric, where the aim is to calculate the number of predictions (probability of prediction values) that lie under a certain allowed error percentage (here 5%). Here, we calculate the in-sample and out-sample "precision count", which measures the number of values which are predicted within the allowed range of error. This measure increases the reliability of the algorithm which we want to use. Figure 4 shows that in general precision count is higher for in-sample predictions (above 70%) than the out-sample predictions (below 50%), which is expected.

Now focusing on "in-sample values" it can be seen that for a range of leaf size values (20 to 40, or 40 to 80) decision tree is very stable in terms of the precision count that it provides. Hence, this makes **decision tree much more reliable, when an algorithm with stable output precision is required.** With increasing leaf size, the in-sample precision count becomes even more spread out for random tree but stays stable for the decision tree.

Now, for "out-sample" values it can easily be seen that how random tree is very unstable in terms of the precision count that it provide for any leaf size. It spreads out over a range of 56% to 70%. However, decision tree is much more stable and reliable around 60%. Therefore, **it is difficult to predict the precision count for random tree in out-sample, unlike decision tree.**

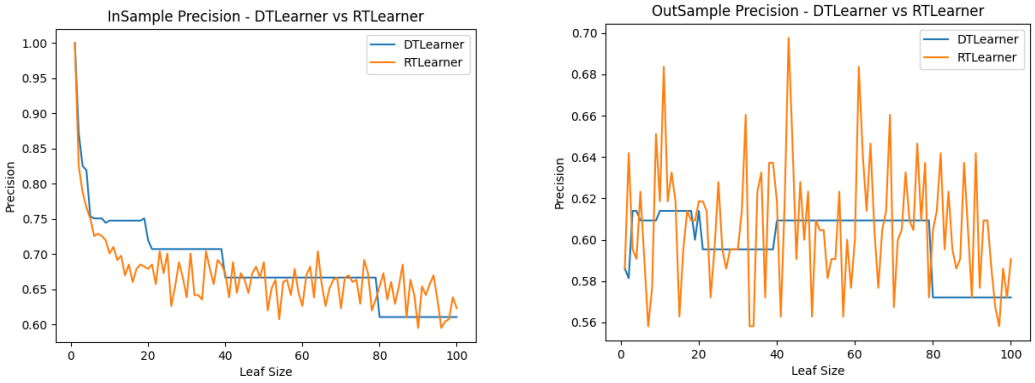

Figure 4—Experiment 3, Plot number of predictions within 5% of error in value with the leaf size: in-sample (left) and out-sample (right)

1.3.3. Space Consumed

This metric counts the number of nodes in a tree that are created after training the data by decision and random tree learners respectively. The metric is plotted against leaf size. Both the algorithms show an exponential decrease in number of nodes or space occupied with the increase in leaf size. However, decision tree has a sharp drop in comparison to the random tree. Decision tree uses the correlation to decide the best feature, hence, space occupied is lower as compared to random tree where the decrease in number of nodes is gradual. Therefore, it can be concluded that the **decision tree will serve well where the space occupied (number of nodes in the tree) is a deciding factor.**

Average number of nodes in DT: 41.78
Average number of nodes in RT: 48.7

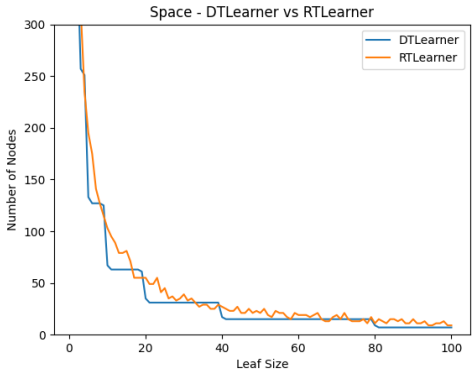

Figure 5—Experiment 3, Plot the number of nodes in the respective tree against leaf size.

4. SUMMARY

The project and experiments performed has provided a detailed evaluation on 1) using bagging to improve the overfitting present in lower leaf sizes; and 2) algorithms required to train and query data depends highly on the requirements of the situation. We need correct metrics to understand the comparison and pointing out the pros and cons for each algorithm. Like, we could tune the bagging more by increasing the number of bags or decreasing the number of bags to analyse the impact on overfitting.

5. REFERENCES

1. https://numpy.org/doc/stable/reference/index.html
2. https://stats.stackexchange.com/questions/398330/accuracy-and-precision-in-regression-vs-classification
3. https://drive.google.com/drive/folders/1xDYIomn9e9FxbIeFcsclSbXHT-tHROD1j
4. https://medium.com/@xaviergeerinck/artificial-intelligence-how-to-measure-performance-accuracy-precision-recall-f1-roc-rmse-611d10e4caac
5. https://edstem.org/us/courses/3788/discussion/241571
6. https://numpy.org/doc/stable/reference/random/generated/numpy.random.choice.html