Project 3: Assess Learners

Saad Rasool
srasool7@gatech.edu

***Abstract*—**The research below represents development of DTLearner (Decision Tree), RTLearner (Random Decision Tree), concept of bagging and insane learner. The goal of this project was to understand compare the above supervised machine learning algorithms. Concepts used to compare algorithm include, mse (mean square error), mae (mean absolute error), and r2 (coefficient of determination) metrics.

Introduction

The DTLearner and RTLearner algorithms are based on JR Quinlan's paper (Induction of Decision Trees), decision trees are a type of supervised machine learning algorithm, where the data is continuously split according to a certain parameter. Supervised machine learning algorithms are where the input and output are provided in the training data. Bagging and insane learner algorithms were also implemented to address over-fitting issue and how it can improve a learner function. In addition to the algorithm design the following concepts are also investigated:

Over-fitting: is machine learning algorithm behavior, where the model is per-formant on a training data but not on testing

Performance metrics: Certain metric like r2 used to evaluate the model

Recursion: Programming algorithm which calls on itself to solve a problem. Require identification of base cases.

Algorithm selection: Understanding which algorithm is better and should be used

Method

This section will walk through each of the learners and set up of the experiments mentioned later in the report. To explain how learners were developed, first we will walk through the data cleaning part of the code. Data cleaning is an extremely important part of every machine learning project, as any algorithm is only as good as the input data. Following data cleaning tactics were used:

- Removing any null or nan values, this is essential as a developer you never know if data used would have any null or nan values which could have a negative impact on you model
- Removing any headers or timestamp columns, in our project we are not using timestamp as an input feature
- Splitting data into in sample (training) and out sample (testing) data, in our case we used

ratio of 60/40 respectively.

Next, we will look into DT and RT learner:

Training data is passed into the add_evidence method call, which calls the build_tree method with the passed training data. The main goal of this function is to build the decision tree. For the DTlearner, we are using a recursion algorithm. Our basescases to create a leaf are, when data.shape[0] == 1, meaning there is only 1 row, hence it has to be a leaf. If all the data.y are the same value, them there is no point of splitting the data, hence we can make a leaf and last case is where split_val is equal to the max of the data then we would create a leaf. In order to select the feature of the data to use, we are taking the feature with the highest correlation with the target. Our splitting value is calculated by the median of the selected feature. The only difference between the DT and RT learner is that the RT learner selects the feature to split on randomly.

For the Bagging bootstrap aggregation implementation number of bags and the learner is passed. The main goal of this is to reduce variance within the prediction of the learner. In bagging a random sample of data in a training set is selected with replacement, meaning that the individual data points can be chosen more than once. After the data samples have been completed depending on the number of the bags, models are then trained independently. Final prediction of the bagging is calculated by taking the avg of all the model prediction for a feature row. Insane learner is very similar to this, the only difference is that it calls the bagging bootstrap aggregation class and then same aggregation method use.

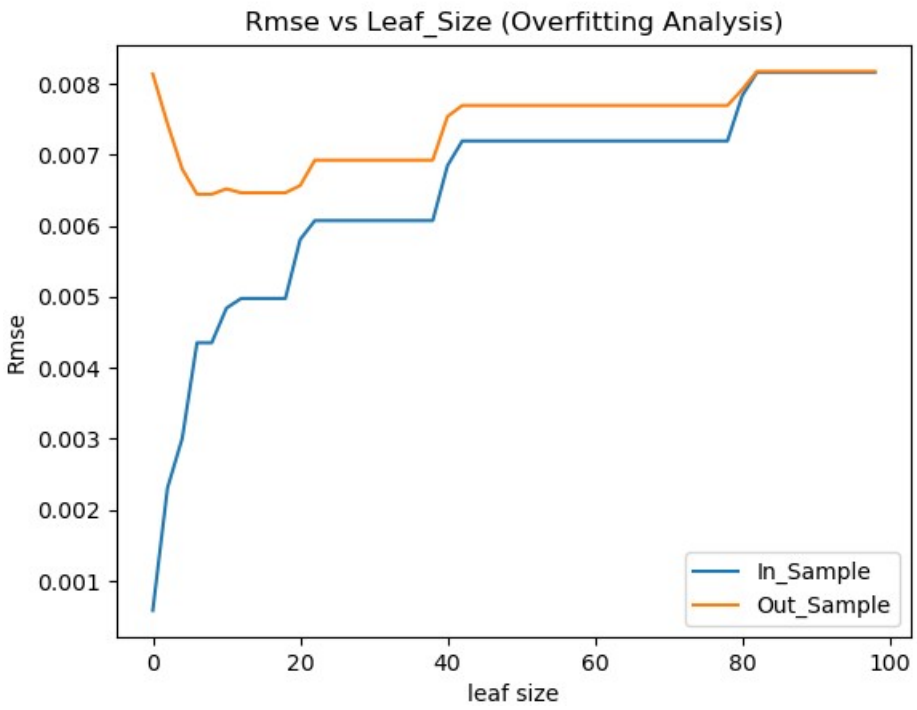
All the function calls and running of the learners were done from testlearners.py, to run the code please run the following command, along with the dataset to be used. It will generate plots which will be saved in the images folder.

PYTHONPATH=../:. python testlearner.py Data/Istanbul.csv

Discussion

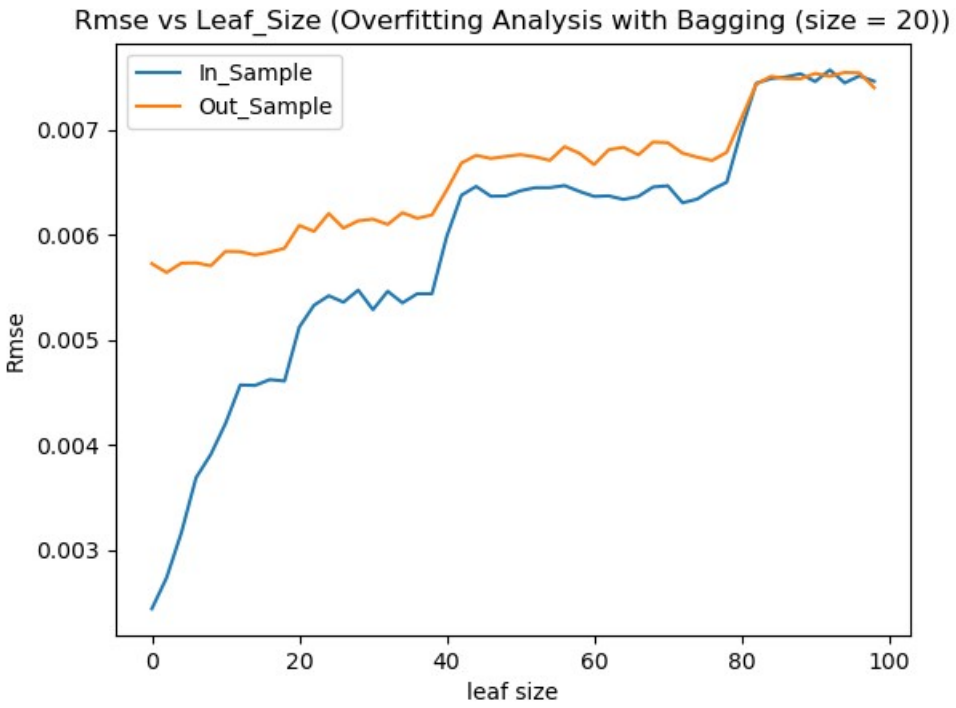
Experiment 1

For the first experiment, we used the DT Learner to understand the concept of over-fitting with respect to leaf_size by using Rmse performance metric. As mentioned before over-fitting happens when model has good metrics on the training data but not on the testing. To identify over-fitting, we need to observe the following, when in-sample error is decreasing and out-of-sample error is increasing. This represents the start of over-fitting. Yes, over fitting does occur with respect to leaf size as shown in the graph below. For a decision tree, lower the leaf size, generally the model would be over-fitted, the reason behind that each leaf is associated with 1 value. If for example leaf size is 5 then each leaf would have an average of 5 or lower values, hence more generic model. The direction of over-fitting is to the left. Meaning the higher the leaf size, less over-fitting the model would be. As shown in the plot be, the over fitting starts at leaf size 1 and ends at leaf size 5 - 6, as soon as the out of sample Rmse flattens.



Experiment 2

In the next experiment, we wanted to understand if we can reduce or eliminate over-fitting as seen and mentioned in experiment 1. To address this issue, we will be implementing bagging bootstrap aggregation algorithm, with a bagging size of 20 and same as experiment 1 we will be varying leaf sizes. Bagging size of 20 was randomly chosen, as mentioned in the introduction bagging size of 20 means that we will be randomly generating 20 data sets with replacement from the 60% training data passed to the call. From the plot below we can see that there is no region of over-fitting. Where the In-sample error is decreasing and out-sample error is increasing. This is happening due to the fact we are taking an average of 20 different DT learner algorithms predictions. From the experiment below we can observe over fitting has been reduced and removed. This experiment was also run multiple times (5) and what we saw that the Rmse value across all leaf sizes gets flatten.

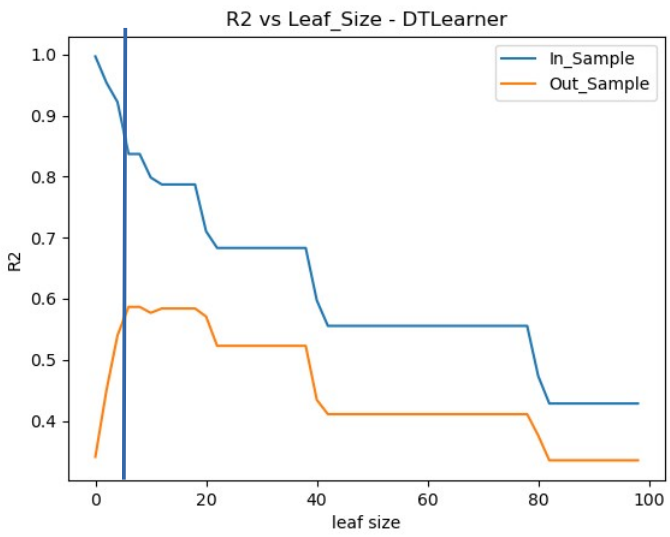


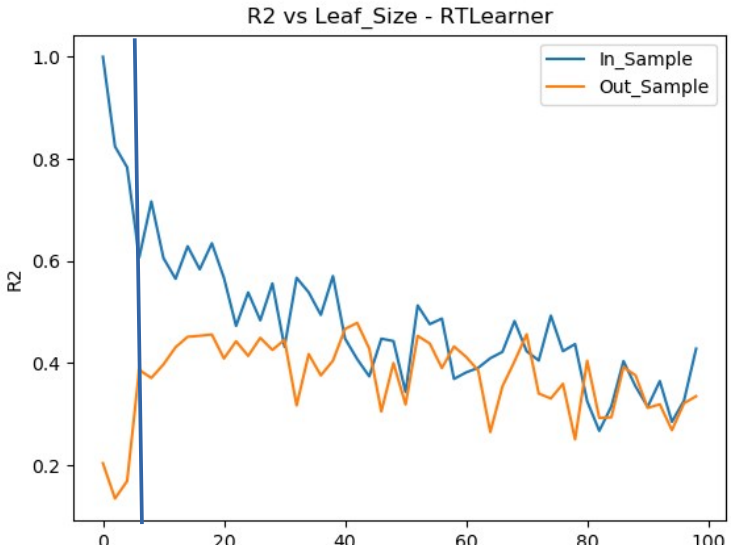
Experiment 3

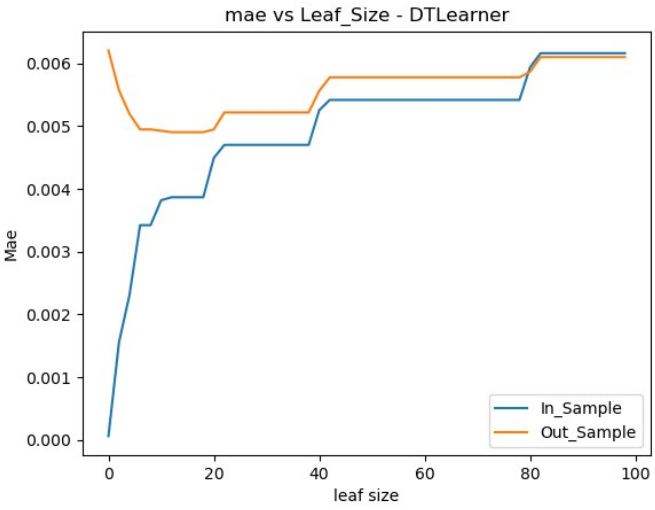
For our last experiment we wanted to compare DT learner and RT learner decision tree algorithms. Where RT learner selects random features. The metrics used to evaluate the models include coefficient of determination and mean absolute error. To understand how the models performed compare to each other, following testing was conducted with leaf size varying from 0 up to 100.

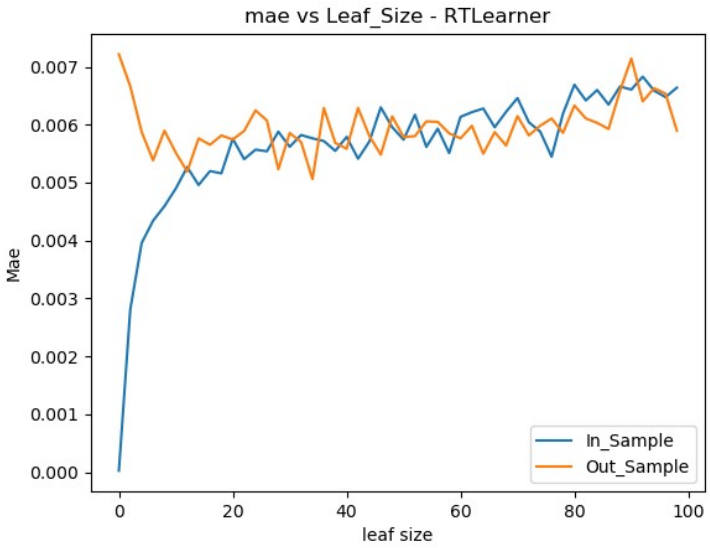
- DT and RT learners both were compared with R2 metric
- DT and RT learners both were compared with mae metric
- DT and RT learners both were compared with R2 metric with bagging of 20
- DT and RT learners both were compared with mae metric with bagging of 20

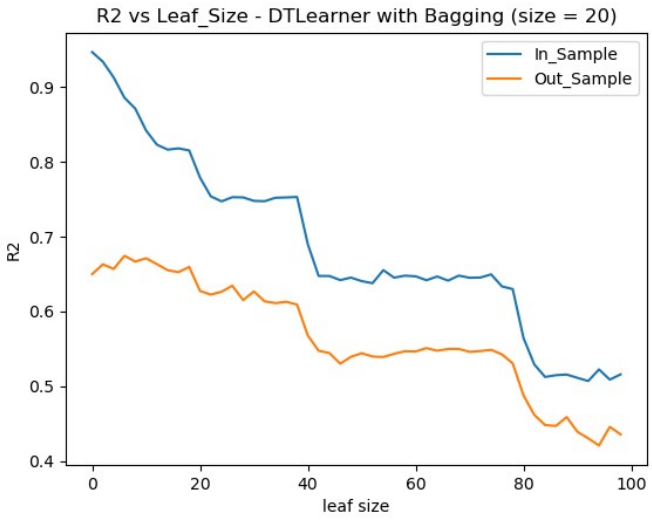
Please see the results below in respective order

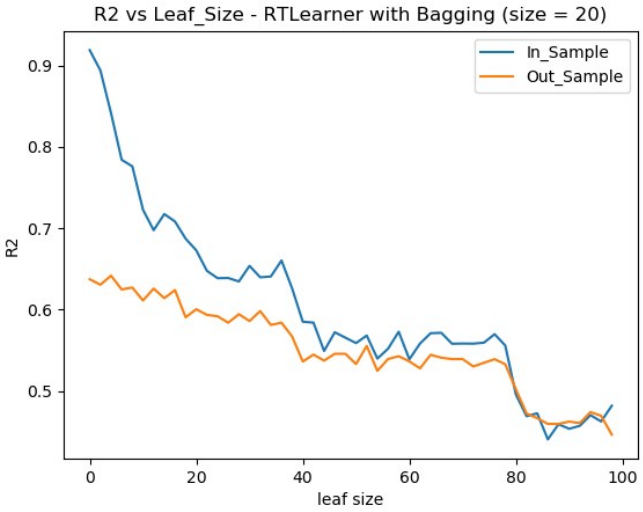

R2 vs Leaf_Size - DTLearner
In_Sample
Out_Sample
R2
leaf size

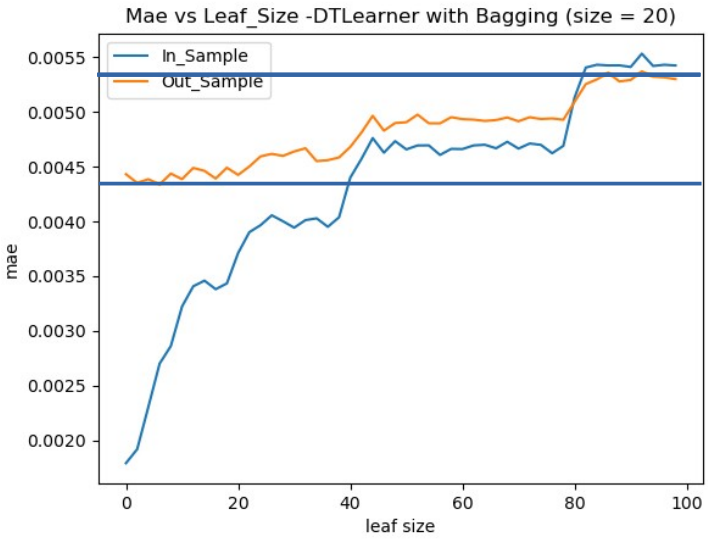

R2 vs Leaf_Size - RTLearner
In_Sample
Out_Sample
R2

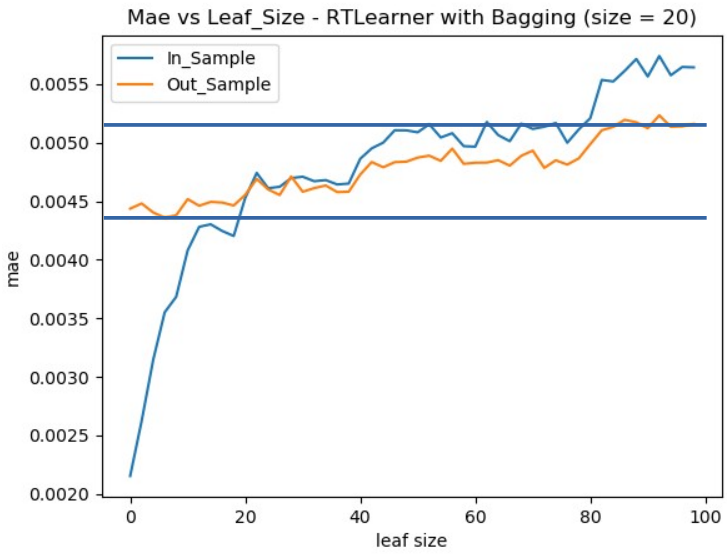

mae vs Leaf_Size - DTLearner
In_Sample
Out_Sample
Mae
leaf size


mae vs Leaf_Size - RTLearner
In_Sample
Out_Sample
Mae
leaf size


R2 vs Leaf_Size - DTLearner with Bagging (size = 20)
In_Sample
Out_Sample
R2
leaf size


R2 vs Leaf_Size - RTLearner with Bagging (size = 20)
In_Sample
Out_Sample
R2
leaf size





From the plots above, lets analyze the plots with out bagging first, we can observe that both models have over fitting present for any leaf size lower than 5 (mae charts). Metric R2 represents the fit of the model to the data, where mae, represents the mean absolute error. If bagging is ignored, and based on the plots above it makes more sense that DTLearner is a better model. The main reason behind this is that the R2 value for the DT learner is much higher than RT learner (0.6 vs 0.4 respectively) right after leaf size 5 for out-sample data, and the same trend is present in mae charts, where error is lower in for DT learner. Another observation made was the variance in the results of RT learner, there are significant of spikes in the plot and that could be the results of random selection fo the factors.

Looking at bagging plots for DT and RT learner, r2 value is fairly similar, but when compared with mae value, for 20 bags RT learner has a lower mae error (total avg of all the leafs) than DT leaner, this could be due to the random selection of the features and data.

DT learner would be better in terms of stability as the selection of features is based on correlation and as seen in the mae plot. RT learner also has significant spikes, meaning the model is not stable. Based on performance of the models on R2 and Mae metrics, RT learner with bagging size of 20 is a better model. We can see that the R2 value is similar but mae value for RT learner is lower overall. Based on the plots, tuning of parameters and usage of bagging makes a significant difference in model performance as seen above, hence one learner is likely not going to be superior to another. But if we do keep every parameter the same, leaf size, bagging, data then RT Learner would be superior as the learner is faster to train and the results are fairly similar or better than DT learner.

Conclusion:

This research illustrated the implementation of decision tree and random decision tree with and with out bagging. We learned that random decision is faster to train than classical tree, but has its own cons (spikes in errors). We also demonstrated the concept of over-fitting and usage of bagging can help improve the model and reduce / eliminate over-fitting. The research above also demonstrated the technique and importance of data cleaning, numpy arrays and recursion programming.

Reference:

Home - springer. (n.d.). Retrieved February 12, 2023, from

https://link.springer.com/content/pdf/10.1007/BF00116251.pdf ‘

Wikipedia Contributors. (2019, February 27). Coefficient of determination. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Coefficient_of_determination

What is Overfitting? - Overfitting - AWS. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/what-is/overfitting/#:~:text=Overfitting%20is%20an%20undesirable%20machine

What is Bagging? | IBM. (n.d.). Www.ibm.com. Retrieved February 12, 2023, from https://www.ibm.com/topics/bagging#:~:text=the%20next%20step-

Decision Trees for Classification: A Machine Learning Algorithm. (n.d.). Xoriant. https://www.xoriant.com/blog/decision-trees-for-classification-a-machine-learning-algorithm#:~:text=Introduction%20Decision%20Trees%20are%20a

Brownlee, J. (2021, January 19). Regression Metrics for Machine Learning. Machine Learning Mastery. https://machinelearningmastery.com/regression-metrics-for-machine-learning/