A Comparison of Single Tree Models

Jiang, X.

Universiteit
Leiden
The Netherlands

A Comparison of Single Tree Models

Xintong Jiang

Thesis advisor: Dr. E. Dusseldorp

Defended on 24 / 8 / 2022

MASTER THESIS

STATISTICS AND DATA SCIENCE

UNIVERSITEIT LEIDEN

Abstract

It is very easy to understand and to interpret a single tree model. However, it is often unstable and relatively inaccurate. The aim of this article is to evaluate and improve the performance of single tree algorithms. In total, three single tree algorithms including Classification and Regression Tree (CART) applied with R package 'rpart', Evolutionary Tree applied with R package 'evtree' and a new method that combining Bayesian Additive Regression Trees (BART) and Born Again Tree were evaluated. We did a bechmark study on six differnet datasets and found that the evolutionary trees and born-again trees both perform better than CART in terms of accuracy. The relative performance between evolutionary and born-again trees depended on the dataset. Evolutionary trees performed better on relatively larger datasets and born-again trees performed better on relatively smaller datasets. However, these single tree methods still showed a huge gap in performance compared to BART, especially when applied to large datasets. we conclude that there is still room for the improvement of single trees compared to ensemble methods.

Contents

1 Introduction

A decision tree is a tree based model which can show the connections between features and outcomes. This method partitions the feature space into a set of rectangles, and then assigns each rectangle a value (Hothorn et al. 2005). The conditions along the path form the decisions and the predicted outcome is based on the observed outcome values of the objects in the terminal node. CART is a kind of decision tree model. Figure 1 is an example of CART fitted with a dataset about house prices in the Boston area containing 506 samples. If owner-occupied homes in an area have an average number of rooms per dwelling (rm) less than 6.974 and percentage of lower status of the population (lstat) equal or more than 5.185, the expected median value (in 1000's) is 41.22.

CART is highly interpretable and widely used in a variety of circumstance. However, it is often unstable and ends up in a local minimum (Grubinger, Zeileis, and Pfeiffer 2014). To address these problems, one way is to apply an ensemble method to our decision tree model. Ensemble methods are techniques that create and combine multiple trees to help improve the stability and performance of the decision tree model. For example, Random Forest (Breiman 2001), Bayesian Additive Regression Trees (Chipman, George, and McCulloch 2010), XGBoost (Chen and Guestrin 2016). However, these ensemble methods can make the solution difficult to interpret. Breiman and Shang introduce born-again trees in 1996 which can build a representer tree that has test accuracy close to multiple-trees methods (Breiman and Shang 1996). By combining this born-again and ensemble method, the performance of the single tree method can be improved as well as keep its excellent interpretability. Breiman and Shang combined bagged trees and born-again method. In this project, we will combine Bayesian Additive Regression Trees, a new ensemble method with better performance, and the born-again method to obtain a single tree.

Another way to improve the stability of the decision tree model is to find a global optimal tree rather than a local optimal tree. Because the search space is too huge, a full-grid search is computationally infeasible. To make the computation less intensive, Grubinger et al. introduced an Evolutionary Trees method to find a global optimal tree (Grubinger, Zeileis, and Pfeiffer 2014). In Evolutionary Trees, a stochastic optimization method is used. With the injected randomness, local optima can be escaped and a global optimum can finally be obtained.

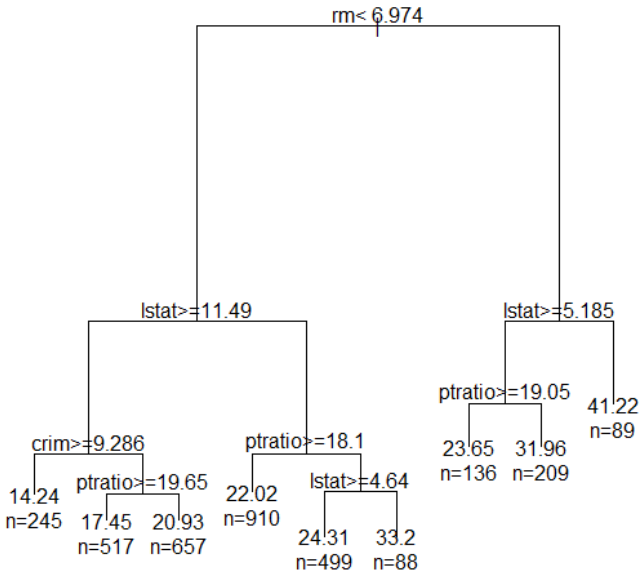

Figure 1: Example Regression Tree for Boston Housing Data (n=506)

The aim of our study is to describe the data with a shallow single tree which has the maximum accuracy on future data. We aim to compare three single tree methods, Classification and Regression Trees, Evolutionary Trees, and born-again trees with the Bayesian Additive Regression Trees.

In order to achieve our aim, methods that end up in a single tree model will be applied to a variety of benchmark datasets and the performance of them will be compared based on predictive accuracy, efficiency and complexity. When the depth of the tree increases, the model becomes more flexible and may become more accurate. But since the number of terminal nodes also increases quickly with depth, the model will become difficult to interpret. So in this study, these three single tree methods are compared when their depths are equal. Next to the three above mentioned single methods, we also apply the ensemble method BART and use it as a golden standard. These three single tree methods are compared with BART to see how much prediction accuracy is lost while applying a single tree method.

2 Method

2.1 Classification and Regression Tree (CART)

CART (Breiman et al. 1984) is a basic method for developing a single tree. Regression tree and classification tree are two main types of CART. The regression tree is used when the predicted outcome is continuous and the classification tree is used when the predicted outcome is discrete. In this thesis project, regression tree is used because all of the outcome variables of our benchmark datasets were measured at a numeric level.

2.1.1 Algorithm

I will describe in the following the algorithm to split one parent node into two left and right child nodes (denoted with L and R).

1. Assume that our data includes p features and one outcome variable, for each of N observations: that is, (x_i, y_i) for $i = 1, 2, ..., N$, with $x_i = (x_{i1}, x_{i2}, ..., x_{ip})$. The sum of squared errors of the two child nodes L and R decided on a series of values a and a series of features A are calculated, and the one with the smallest sum of squared errors (as shown in the following formula) is selected as the optimal feature and optimal split point. l is the output value (defined in step 2) of area L and r is the output value (defined in step 2) of area R.

$$\min_{A,a}[\min_{l} \sum_{x_i \in L(A,a)} (y_i - l)^2 + \min_{r} \sum_{x_i \in R(A,a)} (y_i - r)^2]$$

2. According to the optimal feature A and the optimal split point a, the data set of this node is divided into two left and right child nodes, and the corresponding output value is given.

$$L(A, a) = (x, y)|A(x) \leq a$$

$$R(A, a) = (x, y)|A(x) > a$$

$$l = average((y_i|x_i) \in L(A, a))$$

$$r = average((y_i|x_i) \in R(A, a))$$

3. Replicate the procedure 1 and 2 until convergence.

If there are few samples after pre-pruning and segmentation or the total error drop from the parent node to the error in the two child nodes does not meet the threshold during iteration, in this case, the splitting procedure should be stopped and the average value of the data in each terminal nodes is used as the predicted value.

2.2 Evolutionary Trees

Evolutionary Trees (Grubinger, Zeileis, and Pfeiffer 2014) is a method that can learn globally optimal classification and regression trees. Its idea is inspired by the natural Darwinian evolution algorithm. Some of the concepts used in this algorithm are similar including inheritance, mutation, and natural selection.

2.2.1 Algorithm

In the natural Darwinian evolution algorithm, we have 'population' and in this Evolutionary Trees algorithm, we have many randomly initialized trees. Inheritance in this method is reflected in the fact that two trees are selected as parent trees and child trees are generated based on them. In total, four mutation operators can be chosen: Split, Prune, Major split rule mutation, Minor split rule mutation, and 1 crossover operator can be chosen as the change between parent trees and child trees. And natural selection here means only trees that perform better and have a lower mean squared error in regression problems can be kept for the next iteration and others will be deleted. This algorithm terminates when the predictive accuracy of some of the best-performed trees stabilizes for some iterations. In this way, there is a large chance that we can finally get a globally optimized single tree.

2.3 Bayesian Additive Regression Trees

The ensemble method BART was applied alone as a golden standard and was combined with the born-again method (explained in 2.4) as a new way to get a single tree. In BART, many small subtrees are grown and then the predictions of these small subtree models are summed to get our final prediction. Each subtree is limited by a regularization prior to be a weak learner to avoid overfitting.

2.3.1 Algorithm

So there are two main concepts of the BART model: a sum-of-trees model and regularization priors. First, we will explain the sum-of-trees model:

The BART model is a sum of many single trees. So it can be written as follows:

$$Y = \sum_{j=1}^{m} g(x; T_j, M_j) + \epsilon, \quad \epsilon \sim N(0, \sigma^2)$$

T denotes for a binary tree including decision rules and terminal nodes, and M denotes the mean values in all terminal nodes of T; ϵ denotes the residual; m denotes the number of trees in the sum-of-trees formulation, j is the count of the trees. In computation, instead of fitting the original outcome variable y, we fit the residual R (y minus the predictions of the other subtrees). The subtrees are updated one by one several times. When updating a single subtree we first update its T using the Metropolis-Hasting approach and then update its M. After several loops, we can get our final BART model.

Second, we will explain the regularization priors:
To avoid overfitting, regularization priors are applied. Due to the independence assumption, we only need to define three priors on $p(T), p(M|T), p(\sigma^2)$. The prior on $p(T)$ controls the depth of each subtree. The prior on $p(M|T)$ can make our predicted value end up in a specific range. And the prior on $p(\sigma^2)$ is specified as an inverse χ^2 distribution.The inverse χ^2 distribution is the prior of the normal distribution when its variance is unknown.

2.4 Born Again Trees

Born again trees is a method that can find a single tree representing a more complex predictor like neural nets or bagged/arced trees (Breiman and Shang 1996). Usually, this method is more stable and accurate than CART. In this thesis project, Born again trees was combined with BART to get a representation tree of BART.

2.4.1 Algorithm

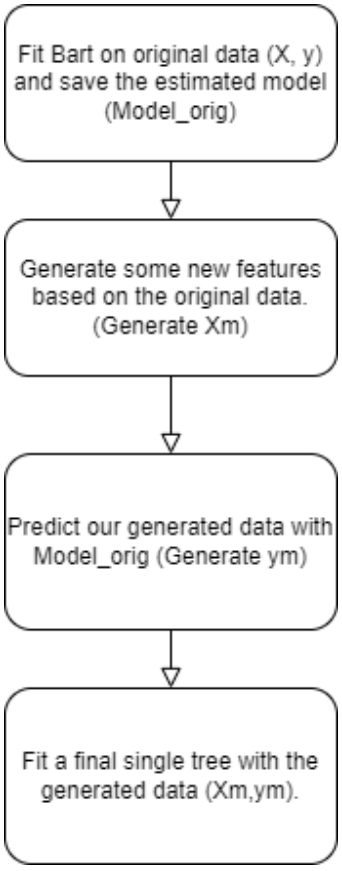

Figure 2: Algorithm of Born Again Trees

A special way of generating data suggested by Breiman and Shang Breiman and Shang (1996) was used here. It helps here when you consider a data-matrix with n rows (cases) and p columns (features). First, a random case $x_i = x_{i1}, x_{i2}, \cdots, x_{ip}$ of the original data was selected with replacement. Then with a possibility of 0.5, a random feature $x_j = x_{1j}, x_{2j}, \cdots, x_{nj}$ was selected and x_{ij} in x_i is replaced by a random value in x_j. Then this x_i was stored as our generated data. This procedure was replicated several times until enough generated cases are produced. We set the number of generated cases ten times as the original dataset. Because usually, no more splitting occurs when more generated data passes down the tree.

3 Benchmark Study

3.1 Data

Datasets of different sample sizes and characteristics were chosen. The types of attributes include binary and numeric. All outcome variables are measured at a numeric level. The sample sizes varied from 167 (Servo) to 1030 (Concrete). These datasets can be obtained from R package 'mlbench' (Leisch and Dimitriadou 2021), 'MAVE' (Weiqiang and Yingcun 2021) and 'MASS' (Venables and Ripley 2002).

Dataset	Instances (n)	Attributes (p)			
		Binary	Nominal	Ordered	Metric
BostonHousing	506	1	-	-	12
Servo	167	-	4	-	-
Hitters	263	3	-	-	16
Cpus	209	-	-	-	6
Auto	392	-	-	-	7
Concrete	1030	-	-	-	8

Table 1: Description of the evaluated benchmark datasets

3.2 Settings

CART is applied with R package 'rpart' (Therneau and Atkinson 2022). Evolutionary trees is applied with R package 'evtree' (Grubinger, Zeileis, and Pfeiffer 2014). The minimum number of observations that must exist in a node before splitting was set to 20 and the minimum number of observations in any terminal node was set to 10 and the maximum value of depth was set to 3, 4 and 5 for these two methods. The root node of the tree means the depth of it is 0. The maximum depth corresponds to the maximum size of the tree, for example, the maximum depth of 3 means that the maximum number of terminal nodes is 8. BART is applied with R package 'dbarts' (Dorie 2021). All other hyper-parameters are set as default values.

3.3 Analysis

3.31 Comparison three single tree models from each other

The relative changes in mean squared errors (MSEs) were used to compare the accuracy and the relative changes in the number of terminal nodes were used to compare the complexity between CART, evolutionary trees, born-again trees. Mean squared errors on model accuracy were estimated with the 0.632+ bootstrap (Efron and Tibshirani 1997). This method can estimate

the sampling distribution with low variance and moderate bias compared to cross-validation and leave-one-out bootstrap. The number 0.632 comes from the sampling probability. If n draws from n samples are randomly made with replacement, the average number of distinct observations in each sample is about $0.632n$. And the overall error can be estimated as follows:

$$Err.632 = 0.368\overline{Err} + 0.632Err_{boot},$$

where $\overline{Err}$ is the training error, and Err_{boot} is the predictive error. In this thesis, first, for each dataset with instances $X = x_1, x_2, \cdots, x_n$, a bootstrap sample of size n was chosen randomly with replacement. Second, for each bootstrap sample, the out-of-bootstrap observations were used as the test sample. For each bootstrap sample, the mean squared error between the predicted and true responses of the out-of-bag samples was used to estimate the Err_{boot} and the mean squared error between the predictive and true responses of the bootstrap samples was used to measure the $\overline{Err}$. The analysis was based on 250 bootstrap samples for each dataset. This ended up in 250 values of $Err.632$.

Then, the relative change in $Err.632$ and the number of terminal nodes M were computed as follows:

$$\text{relative change(Err.632)} = \frac{Err.632 - Err.632_{reference}}{Err.632_{reference}}$$

$$\text{relative change(M)} = \frac{M - M_{reference}}{M_{reference}}$$

The relative change in $Err.632$ and the number of terminal nodes M among three methods were calculated, using each time a different method as the reference.

Due to multiple comparisons, Dunnett's correction (Dunnett 1955) was applied for testing the significance of model accuracy and complexity. If the confidence intervals do not include zero, the two models under comparing are significant differences at the 5% level. For accuracy, if the confidence interval lies to the left of the zero line, the test model outperforms the compared model. If the confidence interval lies to the right of the zero line, the compared model outperforms the test model. For complexity, if the confidence interval lies to the left of the zero line, the test model is more complex than the compared model. If the confidence interval lies to the left of the zero line, the test model is more complex than the compared model.

3.32 Comparison between single tree models and BART

The accuracy of the single tree methods applied to each dataset with the best maximum depth was selected based on the average result of the 250 bootstrap MSE. Then the selected results were compared to BART to see how much prediction accuracy is lost while applying a single tree

method.

4 Result

First, we compare the performance of the single tree methods for a fixed maximum value of the depth of the tree. Figures 3, 4 and 5 show the relative performance of evtree and born again tree compared to rpart.

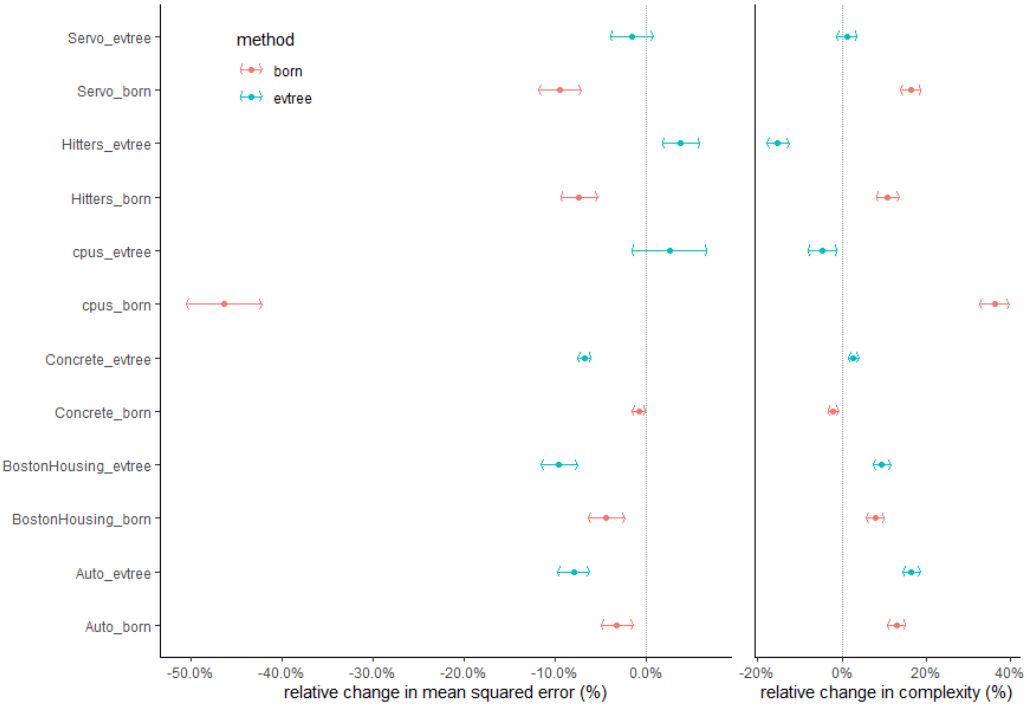

Figure 3: Performance comparison of born-again tree and evtree vs. rpart relative change in mean squared error (left panel) and relative change in number of terminal nodes (right panel) tree depth 3

The relative changes in mean squared error (left panel) and relative changes in number of terminal nodes (right panel) when tree depth is 3 is summarized in Figure 3. Performance differences are displayed relative to rpart's performance. In terms of mean squared error, on 3 out of 6 datasets evtree models significantly outperform rpart models and on 6 out of 6 datasets born-again models significantly outperform rpart models. In terms of complexity, born-again models are significantly more complex on 5 and less complex on 1 datasets and evtree models are significantly more complex on 3 and less complex on 3 datasets compared with rpart models.

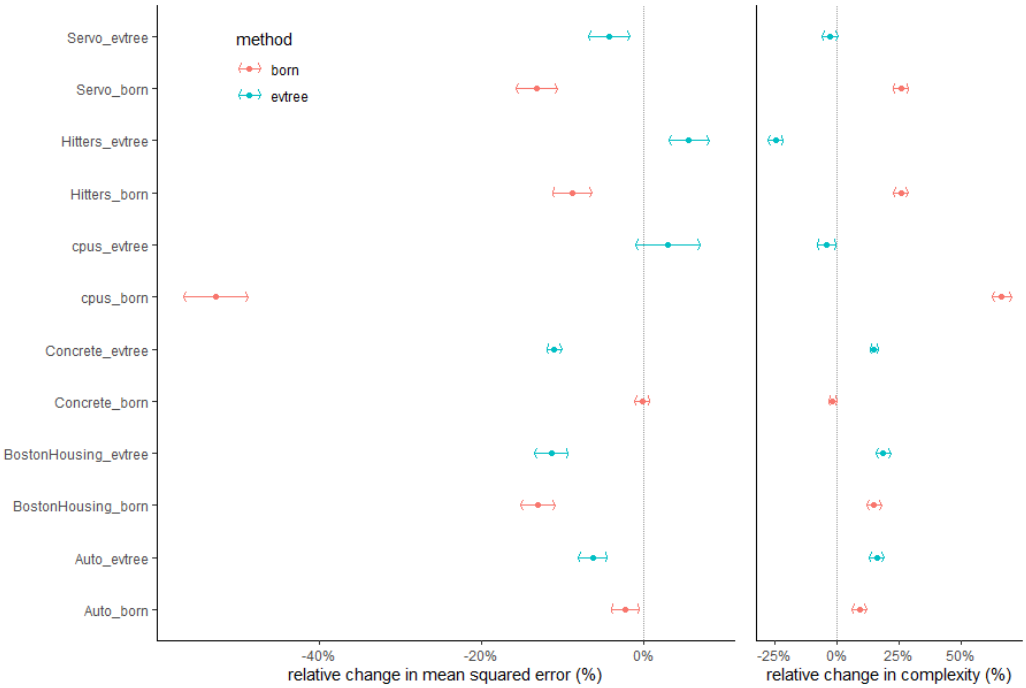

Figure 4: Performance comparison of born-again tree and evtree vs. rpart relative change in mean squared error (left panel) and relative change in number of terminal nodes (right panel) tree depth 4

The relative change in mean squared error (left panel) and relative change in number of terminal nodes (right panel) when tree depth is 4 is summarized in Figure 4. Performance differences are displayed relative to rpart's performance. In terms of mean squared error, on 4 out of 6 datasets evtree models significantly perform better than rpart models and on 5 out of 6 datasets born-again models significantly perform better than rpart models. In terms of complexity, evtree models are significantly more complex on 3 and less complex on 1 datasets and born-again models are significantly more complex on 5 and less complex on 1 datasets compared with rpart models.

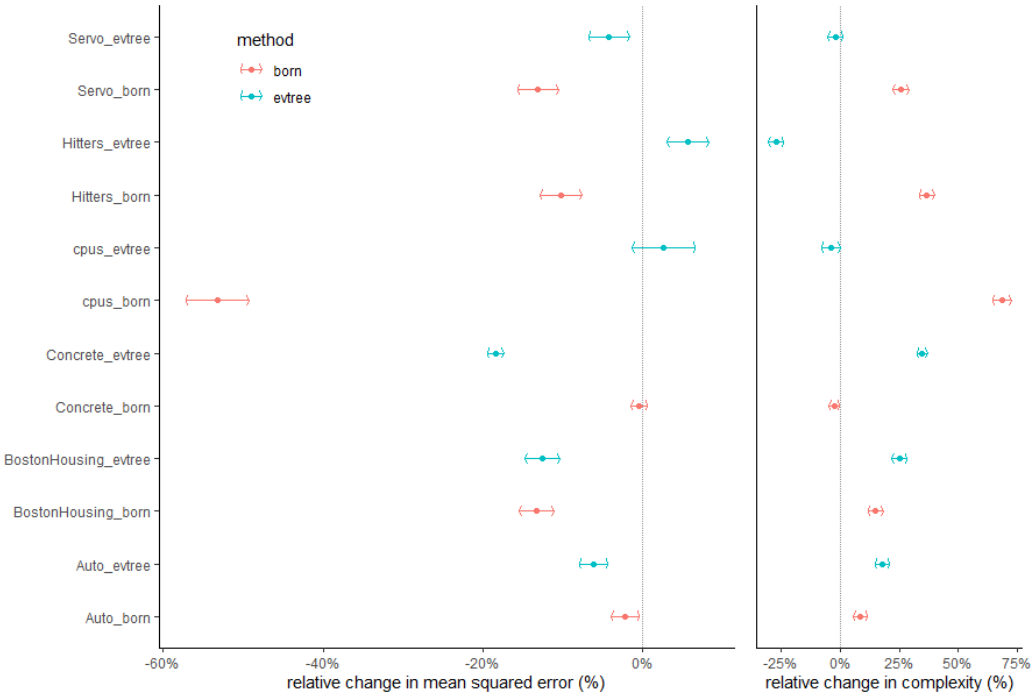

Figure 5: Performance comparison of born-again tree and evtree vs. rpart relative change in mean squared error (left panel) and relative change in number of terminal nodes (right panel) tree depth 5

The relative change in mean squared error (left panel) and relative changes in the number of terminal nodes (right panel) when tree depth is 5 is summarized in Figure 5. Performance differences are displayed relative to rpart's performance. In terms of mean squared error, on 4 out of 6 datasets evtree models significantly perform better than rpart models and on 5 out of 6 datasets born-again models significantly perform better than rpart models. In terms of complexity, evtree models are significantly more complex on 3 and less complex on 1 datasets and born-again models are significantly more complex on 5 datasets compared with rpart models. The results of the comparisons are the same as in figure 4.

Comparing the left panel and right panel, it can be found that when a tree is more complex it usually performs better. Combing all three figures above, as the depth gets deeper, the relative differences in mean squared error between evtree and rpart, born-again tree and rpart get larger and the relative differences in complexity also get larger. Relative differences of mean squared error from maximum depth 3 (Figure 3) to maximum depth 5 (Figure 5) change from 0 to 10% to 0 to 20%. Relative differences of complexity also change from 0 to 20% to 0 to 25%.

To find out whether the difference between the performance of born-again trees and evtree is significant. Performance comparison of born-again tree vs. evtree was also plotted. Figure 6 shows the relative performance of born-again trees compared to evtree when the maximum tree depth was set to 3. Figures, when maximum tree depth was set to 4 and 5, are attached in the appendix.

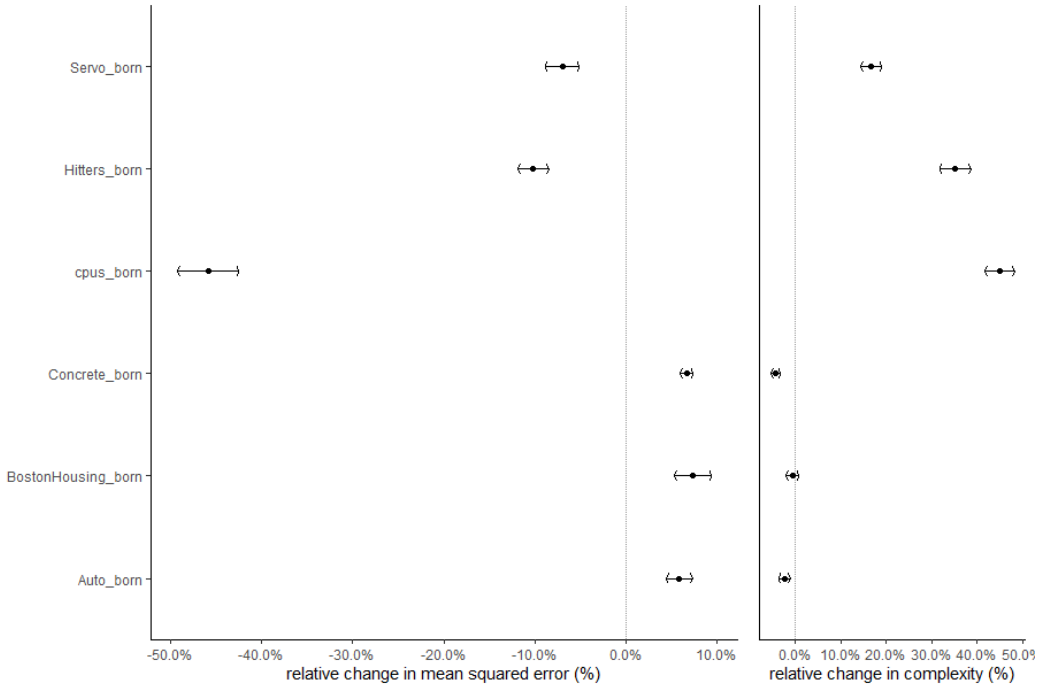

Figure 6: Performance comparison of born-again tree vs. evtree relative change in mean squared error (left panel) and relative change in terminal nodes number (right panel) tree depth 3

The relative change in mean squared error (left panel) and relative change in terminal nodes number (right panel) when tree depth is 3 is summarized in Figure 6. Performance differences are displayed relative to evtree's performance. It can be observed that on 3 out of 6 datasets born-again models significantly perform better than evtree models and on 3 out of 6 datasets evtree models significantly perform better than born-again models in terms of mean squared error. In terms of complexity, born-again models are significantly more complex on 3 and less complex on 2 datasets.

When the maximum tree depth is set to 4, on 3 out of 6 datasets born-again models significantly perform better than evtree models and on 2 out of 6 datasets evtree models significantly perform better than born-again models in terms of mean squared error. In terms of complexity, born-again models are significantly more complex on 3 and less complex on 2 datasets.

When the maximum tree depth is set to 5, on 3 out of 6 datasets born-again models significantly perform better than evtree models and on 2 out of 6 datasets evtree models significantly perform better than born-again models in terms of mean squared error. In terms of complexity, born-again models are significantly more complex on 3 and less complex on 3 datasets.

Combing the three figures above, a majority number of relative changes in MSE is between -20% and 20%. Only born-again method applied to 'cpus' dataset shows an around 50% difference.

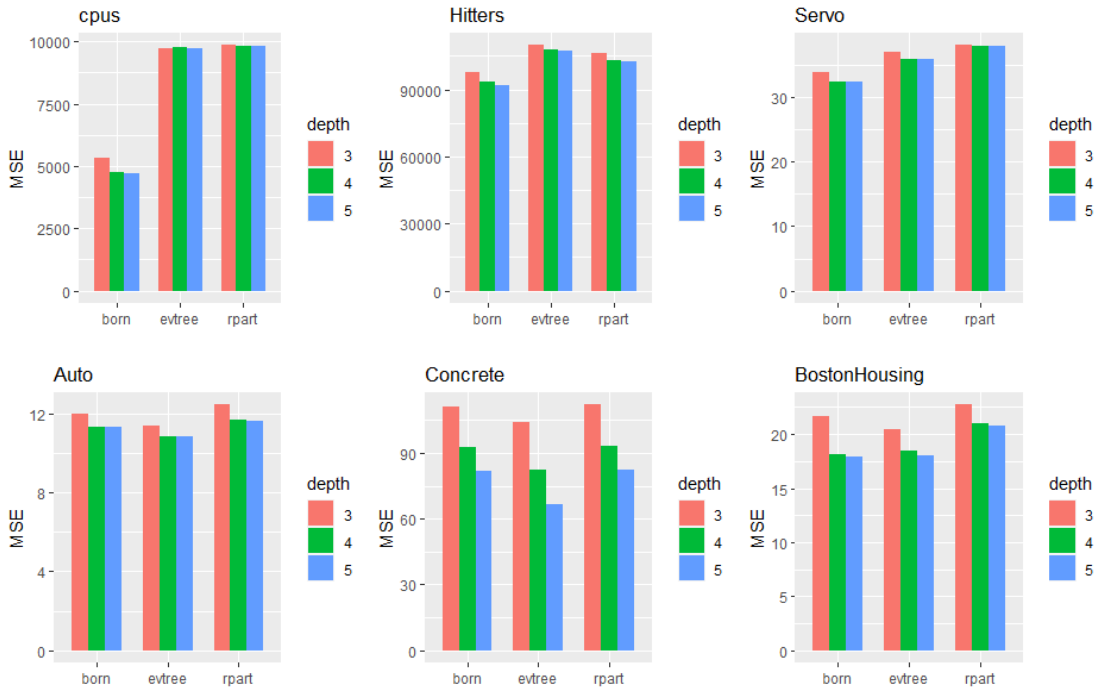

Figure 7: Comparison of the average (absolute value of) $Err.632$ across the 250 bootstrap samples of different maximum tree depth

It can be observed that almost all models perform best when the maximum tree depth is 5. Only the results obtained from evtree and rpart applied to 'cpus' dataset is not clear. The exact number of MSE was checked. For 'cpus' dataset, evtree performs the best when the maximum tree depth is set to 3 and rpart performs the best when the maximum tree depth is set to 5. Another thing worth noting is that the MSE decreases a lot when the maximum tree depth varies from 4 to 5 in dataset 'Concrete'. This could mean that the maximum tree depth 5 is not enough to describe this dataset and the MSE can be further reduced when the tree depth gets deeper. For other datasets, the performance difference between maximum tree depth 4 and 5 is relatively small.

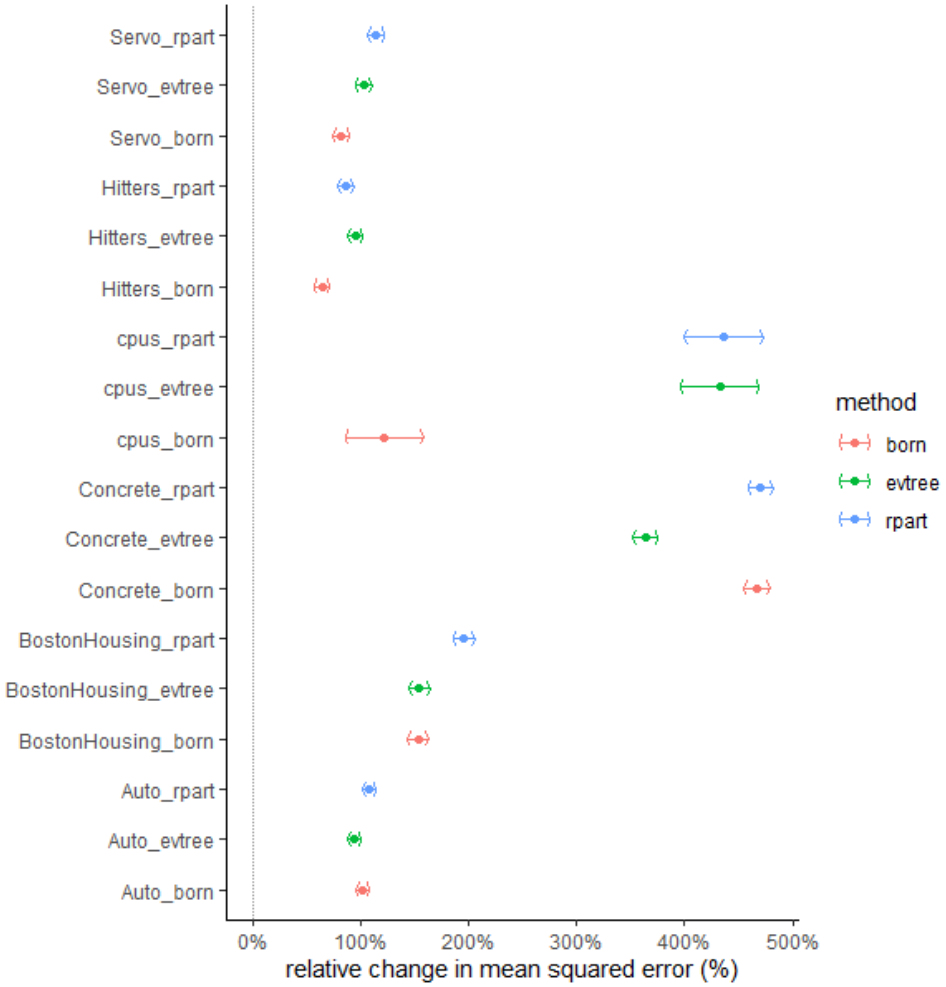

Figure 8: Performance comparison of rpart, evtree and born-again tree vs. BART relative change in mean squared error

This is a plot showing confidence intervals comparing BART to the other methods. It can be observed that all the confidence intervals lie to the right of the zero line. So BART models are significantly outperforming rpart, evtree and born-again models. The highest relative difference appears in dataset 'Concrete' and it is around 600%. Other relative differences are almost around 200%.

To measure the efficiency, 25 bootstrap samples were drawn and the average computation time for each dataset was recorded. Again, trees of different maximum values of depth (3,4 and 5) were fitted. Rpart has on average the shortest computation time. When the dataset is small, there is not much difference in computation time between the evtree and born-again models. But when the datasets are larger, as the tree depth goes deeper, the computation time for evtree increases much quicker than born-again models.

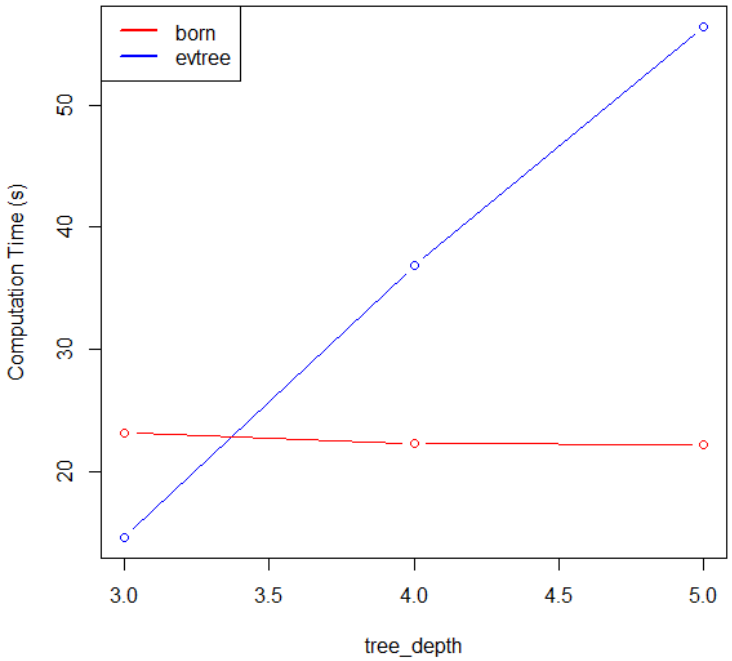

Figure 9: Computation time of evtree and born-again models with dataset 'Concrete'

Take dataset 'Concrete' for example, the computation time grows from 15s to 56s for evtree models and does not change much for born-again models. This dataset has 1030 instances and we can assume that if we try larger datasets the computation time will increase much quicker.

5 Discussion

5.1 Summary

In this study, we described 6 datasets with a shallow single tree to investigate which single tree method can give the best prediction. Three single tree methods, classification and Regression Trees, Evolutionary Trees, and born-again trees were compared with each other and the Bayesian Additive Regression Trees was used as a golden standard. 250 bootstrap samples per dataset were created and for a fair comparison, each method was applied to the same sample keeping the maximum possible depth of the tree equal.

We found that CART does not perform the best in any of the 6 datasets. And the relative performance of evolutionary trees and born-again trees depended on the datasets. Evolutionary trees performed the best for the dataset 'BostonHousing', 'Concrete' and 'Auto' and born-again trees perform the best for the dataset 'Servo', 'Hitters' and 'cpus'. From these results, it is not clear what features influence the performance of the methods. But 'BostonHousing', 'Concrete' and 'Auto' are the three datasets with the largest number of instances. So we can assume that evolutionary trees perform better when the dataset is relatively larger and born-again trees perform better when the dataset is relatively smaller. But we should be careful that if the dataset is large, as the tree depth goes deeper, the computation time of evolutionary trees grows very quickly and this may cause problems in practical situations.

No matter which single tree model we applied and which dataset we applied to, the MSEs of the three single tree methods are 2 to 6 times as higher as the MSEs of BART. And the relative difference is extremely high when the dataset is large, which means that BART has an irreplaceable advantage compared to a shallow single tree especially with a large dataset (instances > 1000). In general, there is a big room left for further improvements.

Previous studies have found that the evolutionary trees and born-again trees have huge advantages over CART in terms of accuracy. Grubinger, Zeileis, and Pfeiff Grubinger, Zeileis, and Pfeiffer (2014) also did a benchmark study comparing evolutionary trees and CART and found that on 12 out of 17 datasets evolutionary trees significantly outperform CART in terms of accuracy. 'BostonHousing' dataset was also included in their study. The relative difference between them is around 17%. Our study also found a difference of around 17% when the maximum depth equals 5. This is also consistent with our study. Breiman and Shang developed the born-again method and combined it with bagged trees. They also did a benchmark study and found that it can decrease the predictive error of 'BostonHousing' dataset by 22% compared to CART. In our study, the relative difference of our born-again trees and CART is around 17%. It seems that

our tree performed worse than the born-again trees developed by Breiman and Shang. But in their study, they used fully grown trees with an average number of terminal nodes of 58. In our study the tree depth is shallow and the average number of terminal nodes, when the maximum tree depth is 5, is around 20. Our result shows that a small born-again tree also significantly outperforms CART.

5.2 Limitation and Future research

This study still leaves much room for further improvement. First, evtree, born-again trees and rpart we applied in this study can be further optimized. Almost all the hyper-parameters were set as the default values and these hyper-parameters can be tuned to get a more accurate model. For example, the hyper-parameter alpha in evtree, regulates the complexity part of the cost function, may have influenced the actual tree sizes. And the number of instances we generated in born-again trees may also have influenced the model accuracy and the tree sizes. Besides issues related to hyper-parameters, pruning is another thing that should be considered. To compare the three methods under the same maximum tree depth, post-pruning is not applied in this study. In further studies, we can grow a large tree and then prune it using cost-complexity pruning. This method can find a tradeoff between tree size and its goodness of fit to the data and thus may lead to better performances (Hastie et al. 2009).

Second, although we set the maximum tree depth of each model to an equal value, the number of terminal nodes is still different. That is because some of the nodes stop splitting before the maximum depth is reached. The reasons for stopping can either be 1. the minimum number of observations that must exist in a node in order for a split to be attempted is not fulfilled. 2. the decrease in sum-of-squares due to the split does not exceed some threshold. So the comparison of the three single tree methods is still unfair. Maybe, the pruning method mentioned above is a good choice for getting trees of the same number of terminal nodes.

Third, it is still unclear what patterns influence the performances of our tree-based methods. The benchmark datasets that were used in this study are too few. More datasets should be tested to give a more certain answer on what patterns influence the performances. Also, another way to find the important patterns of the datasets is to do simulation studies. In this kind of study, the patterns can be generated by ourselves and thus the 'truth' of the dataset is known. For the reason above, it can give us a better understanding of the behavior of the single tree algorithms.

5.3 Conclusion

In conclusion, the evolutionary trees and born-again trees both perform better than CART in terms of accuracy. And the relative performance between them depends on the datasets. Evolutionary trees perform better on relatively larger datasets and born-again trees perform better on relatively smaller datasets. However, these single tree methods still show a huge gap between BART, especially when applied to large datasets.

References

Breiman, Leo. 2001. "Random forests." *Machine Learning* 45 (1): 5–32.

Breiman, Leo, Jerome H Friedman, Richard A Olshen, and Charles J Stone. 1984. *Classification and regression trees.* Belmont, CA: Wadsworth International Group.

Breiman, Leo, and Nong Shang. 1996. "Born again trees." *University of California, Berkeley, Berkeley, CA, Technical Report* 1 (2): 4.

Chen, Tianqi, and Carlos Guestrin. 2016. "Xgboost: A scalable tree boosting system." In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining,* 785–794.

Chipman, Hugh A, Edward I George, and Robert E McCulloch. 2010. "BART: Bayesian additive regression trees." *The Annals of Applied Statistics* 4 (1): 266–298.

Dorie, Vincent. 2021. *dbarts: Discrete Bayesian Additive Regression Trees Sampler.* R package version 0.9-20. https://CRAN.R-project.org/package=dbarts.

Dunnett, Charles W. 1955. "A Multiple Comparison Procedure for Comparing Several Treatments with a Control." *Journal of the American Statistical Association* 50 (272): 1096–1121.

Efron, Bradley, and Robert Tibshirani. 1997. "Improvements on cross-validation: the 632+ bootstrap method." *Journal of the American Statistical Association* 92 (438): 548–560.

Grubinger, Thomas, Achim Zeileis, and Karl-Peter Pfeiffer. 2014. "evtree: Evolutionary learning of globally optimal classification and regression trees in R." *Journal of statistical software* 61:1–29.

Hastie, Trevor, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. 2009. *The elements of statistical learning: data mining, inference, and prediction.* Vol. 2. Springer.

Hothorn, Torsten, Friedrich Leisch, Achim Zeileis, and Kurt Hornik. 2005. "The design and analysis of benchmark experiments." *Journal of Computational and Graphical Statistics* 14 (3): 675–699.

Leisch, Friedrich, and Evgenia Dimitriadou. 2021. *mlbench: Machine Learning Benchmark Problems.* R package version 2.1-3.

Therneau, Terry, and Beth Atkinson. 2022. *rpart: Recursive Partitioning and Regression Trees.* R package version 4.1.16. https://CRAN.R-project.org/package=rpart.

Venables, W. N., and B. D. Ripley. 2002. *Modern Applied Statistics with S.* Fourth. ISBN 0-387-95457-0. New York: Springer. https://www.stats.ox.ac.uk/pub/MASS4/.

Weiqiang, Hang, and Xia Yingcun. 2021. *MAVE: Methods for Dimension Reduction.* R package version 1.3.11. https://CRAN.R-project.org/package=MAVE.

Appendix

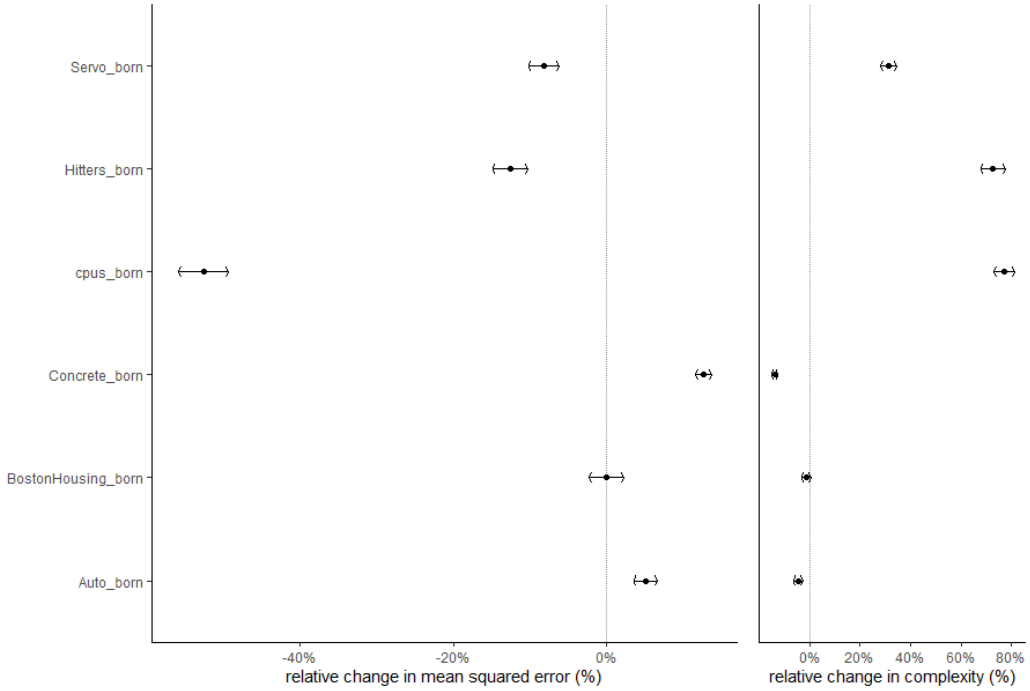

Figure 10: Performance comparison of born-again tree vs. evtree relative changes in mean squared error (left panel) and relative changes in terminal nodes number (right panel) tree depth 4

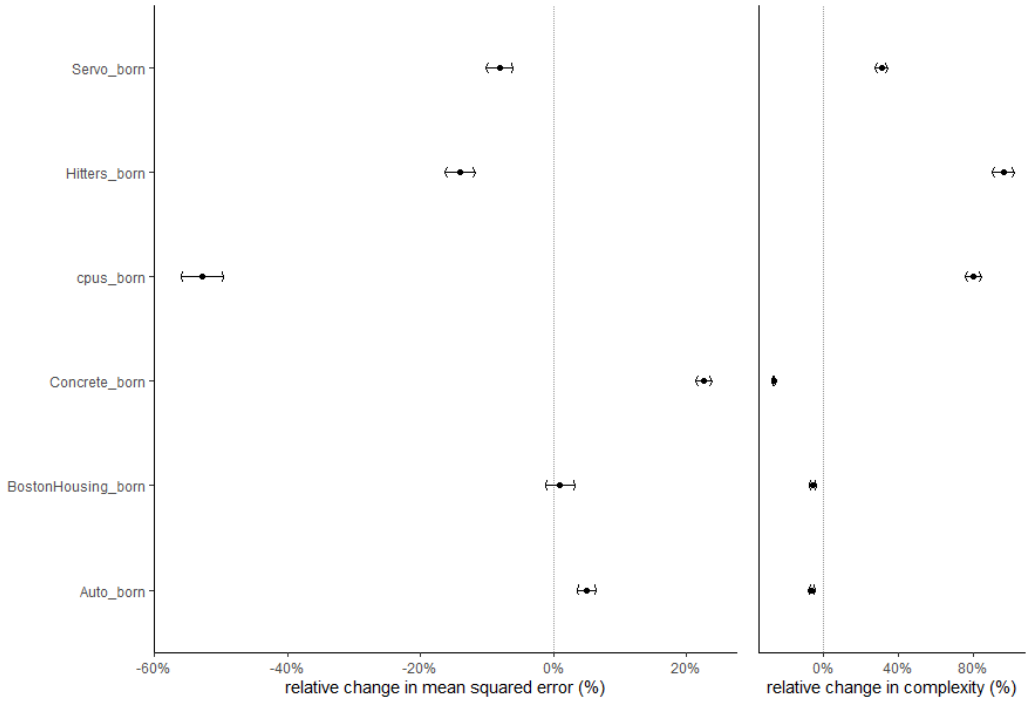

Figure 11: Performance comparison of born-again tree vs. evtree relative changes in mean squared error (left panel) and relative changes in terminal nodes number (right panel) tree depth 5

Codes can be found via the following link.
https://github.com/marjoleinF/Tree-thesis-project/tree/main/Project%20Xintong