INSTITUTE OF FINANCE AND STATISTICS

University of Bonn


**Seminar Paper in Academic Practice**

# Regression Trees

Timothy Currie

| | |
|---|---|
| Supervisor: | Dr. Elias Wolf |
| Semester: | Summer Term 2024 |
| Author: | Timothy Jakob Currie |
| Matriculation-Nrs.: | 50074426 |
| Email: | s69tcurr@uni-bonn.de |
| Subject: | Bachelor Economics |
| Submission: | 25. August 2024 |

# Contents



# 1 Introduction

A large part of modern economic research is built upon linear regression. To conduct and understand economic research, it is particularly crucial to identify where it performs well and where it's limitations lie and other methods can improve upon it. Regression Trees are a powerful machine-learning technique, that can be useful in many situations where linear regression falls short. Particularly for data that includes non-linear relationships and interaction effects Linear regression will often perfor poorly. And Regression Trees can be a important alternative. Regression Trees also have numerous extensions that greatly improve their usefulness and aplicabillity that it is worthwhile to study.

The CART algorithm or variations on it, that most tree based methods use was first introduced in (Breiman et al., 1984). Since then many extensions have been developed such as Random Forets, a very popular Tree based ML method. (Biau and Scornet, 2016) serves as a good introduction to that method. (James et al., 2021) serves as an excellent modern introduction to Regression trees and gives a gentile introduction to the topic and many extensions. (Tan and Roy, 2019) give a deeper dive on BART, one powerful ensemble method.

In this paper I will explain the theory behind regression trees and, compare them to linear regression using a series of simple simulations and on a dataset of student test performance. Simulations can allow for comparison under ideal circumstances, allowing one to focus on the issue

one is interested in, while real world data can give a more realistic perspective on a methods actual performance. I will focus especially on the regression Trees ability to naturally capture interaction effects, and how pruning can combat overfitting. I will also highlight how ensemble methods, a kind of extension to regression trees, can enhance their performance substantially.

In section 2 I will give a brief overview of the theory of regression trees, and where and where not we might expect them to perform well. And will also explain the bias-variance trade-off and explain how pruning helps and ensemble methods that are extensions to trees.

In section 3 I will run a number of simulations comparing the performance of Linear Regression and Regression Trees under ideal conditions.

In section 4 I will apply these methods to a dataset of student test performance. Highlighting pruning and the BART ensemble method. Also showing some of the shortcomings of Regression Trees.

Section 5 is the conclusion and will sum up what I have found and further interesting areas where research could be done.

This dataset used in this paper is publicly available at Kaggle: Student Alcohol Consumption Dataset. All code used for the simulations and data analysis is available at my GitHub.

## 2 Regression Trees

Regression Trees are a type of supervised machine-learning algorithm that recursively splits the predictor space into smaller rectangular subregions. This process approximates an unknown function $f$ by minimizing a measure of loss at each split. Unlike linear regression, Regression Trees don't make assumptions about linearity or non-interaction between different dimensions, making them particularly useful for complex, non-linear relationships in data.

The core mechanism of Regression Trees involves splitting the predictor space into regions that minimize the residual sum of squares, given by:

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \tag{1}$$

In each region, $\hat{y}$ typically takes on the mean of all observations in that region. Each branching of the tree divides the predictor space into one extra reason, for numerical variables typically of the form $\{x \leq c\}$ or $\{x > c\}$. This process continues until a specified threshold is reached, such as a

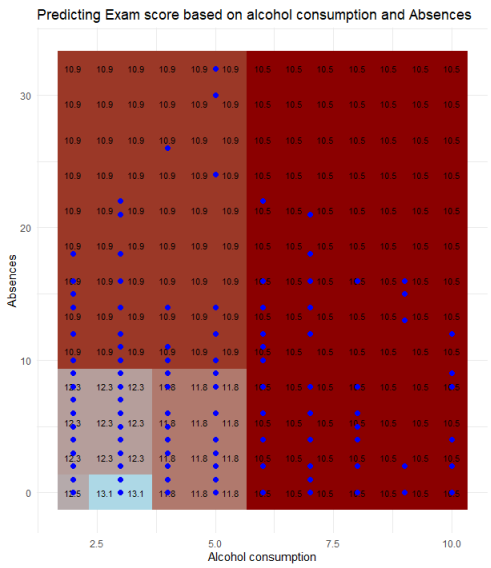


Figure 1: Visualization of Regression Tree's predictions

minimum number of observations in each leaf node or a maximum tree depth. Figure 1 shows one way to visualize the way a Regression Tree will carve up the predictor space and make predictions for each reagion.

The algorithm employs a greedy approach called Recursive binary splitting to find the optimal split at each stage to minimize prediction error.

Regression Trees offer several advantages over traditional linear regression methods. Furthermore, they naturally account for interaction effects. For instance, if, in ones dataset, being tall increases income but only for men, a Regression Tree will naturally partition the data along gender and only along hight for the male half. Wheras for a linear regression to capture this interaction effect would require one to explecitly include this interaction on one of ones regressors. This ability to model complex interactions without manual specification is a significant strength of the method.

One major weakness of Regression Trees however is their tendency to overfit on the data. To capture interesting relationships one must allow ones Tree to grow deep with multple levels of splits. Their flexibility allowing them to create very specific rules also allows them to capture noise in the training data rather than true underlying patterns. Figure 2, illustrates how a Regression Tree can overfit a dataset.

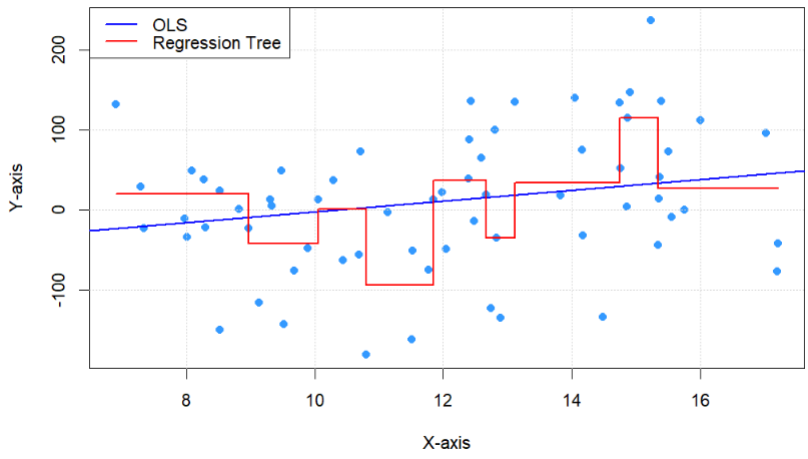


Figure 2: Regression Tree overfitting

## 2.1 Pruning

The main way to adress the overfitting problem is to, in keeping with the arborial theme, "prune" one's regression tree. In cost complexity pruning one starts by growing a large tree is grown that is bound to overfit the training data. Then, a complexity parameter $\alpha$ is introduced to penalize the tree's size. The pruning process is guided by the following formula:

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_j})^2 + \alpha |T| \tag{2}$$

Where $|T|$ is the number of terminal nodes in the tree. The tree shrinks in a predictable fashin as the cost complexity parameter $\alpha$, is increased and it becomes harder for each split to justify its existence. As one prunes a large tree by increasing $\alpha$, the tree naturally shrinks in a well-behaved fashion, allowing one to select the subtree with the lowest cross-validation error. Evaluating a model on different data then it was trained on allows us to see it's real prformance and select a subtree that achieves a good trade-off between bias and variance. In the `rpart` library cross-validation is performed automatically each time one grows a tree.

## 2.2 Ensemble Methods

While pruning can improve the performance of regression trees, they often still underperform compared to other machine-learning methods. This has lead to the development of a number

of ensemble methods, which combine many regression trees to create more robust and accurate predictions. As the well know jury theorem shows, if predictors are (sufficiently) uncorrelated and a predictor's answer is better than pure chance adding more predictors will increase the quality of the average of all answers (de Condorcet, 1785). Ensemble methods leverage this "wisdom of crowds" effect.

Two ways this can be done is by Growing many independent trees and averaging them. Random Forests are a very popular ML method that does this. Or by iterativly growing trees on the residuals of the current model, improving the overall model over multiple generations, as in boosting or Bayesian Additive Regression Trees (BART).

Random Forests create an ensemble by growing many decorrelated trees. Each tree is trained on a random subset of the training data and is only allowed to consider a random subset of features at each split. The final prediction is then formed by averaging the predictions of all trees, typically around 500. This approach reduces overfitting while maintaining the ability to capture complex patterns in the data (Biau and Scornet, 2016).

I will go into slightly more detail for the BART method, the method I used on the dataset. BART model the data as a sum of many Trees plus noise:

$$Y_i = \sum_{j=1}^{m} g(X_i; T_j, M_j) + \epsilon_i \tag{3}$$

Where $m$ is the number of trees, typically around 200. And $g(X_i; T_j, M_j)$ represents the contribution of the $j$-th tree. $T_j$ and $M_j$ represent the tree's structure and predictions respectively.

BART emyloys a Bayesian approach, incorporating a prior that discourages trees from growing too large. Specifically, the probability that a node at depth $d$ is non-terminal is given by $\alpha(1+d)^{-\beta}$. This prior on shallow trees, restricts the trees to be weak learners that only explain a small portion of the overall data and are better suited to being combined through summation and averaging. In doing this BART avoids overfitting and leverages the strengths of ensemble learning. Additionally, BART employs other priors to guide the estimation of terminal node parameters, though a detailed discussion of these is beyond the scope of this paper. This Bayesian approach enables the model to provide both accurate predictions and quantative measures of uncertainty.

Default values for these hyperparameters of $\alpha = 0.95$ and $\beta = 2$ are recommended by (Chipman et al., 2010). And generally provide good performance.

The BART algorithm proceeds by first generating e.g., 200 trees, without any splits. For each

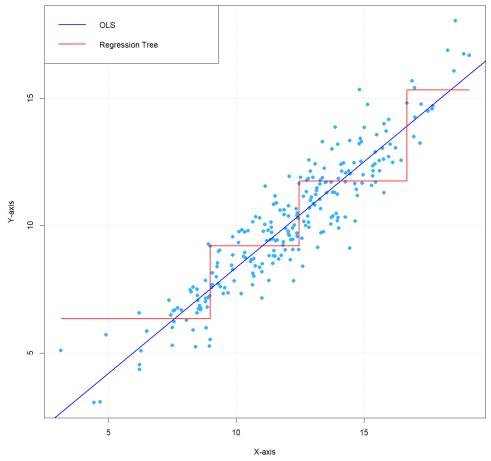


Figure 3: Linear Relation between Variables

Tree $j$, considering all the current trees except $j$, the residual error $R_{-j}$ is calculated:

$$R_{-j} = Y - \sum_{t \neq j} g(X, T_j, M_j). \tag{4}$$

Then a change to that tree's structure is proposed, such as pruning or growing a node, or changing a splitting rule, and is accepted or rejected based on its posterior probability. Meaning the split has to be in accordance with the model parameters as well as fit the data well enough. This process is repeated for all $m$ trees. That constitutes one iteration of the algorithm. Typically, BART uses 1000 burn-in iterations followed by 1000 sampling iterations. An estimate for $f(x)$ is obtained by taking the average over all sample iterations.

A more complete exposition can be found in (Tan and Roy, 2019) or (Chipman et al., 2010).

Ensemble methods have demonstrated remarkable success across various applications, often surpassing individual regression trees and many other machine learning techniques.

# 3 Simulations

Simulations can serve as a testing ground for statistical methods, allowing for easier repeatability and eliminating many of the complications that arise when using datasets. Simulations are especially useful in comparing two different methods. In this section I will compare Linear regression and regression trees on linear and non-linear simulated data, each simulation having been run 400 times to average out noise. The regression Trees were limited to four terminal nodes for both simulations.

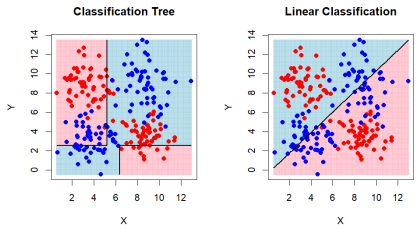


Figure 4: Complex, overfitting Tree

For the first simulation the data follow a simple linear relationship with $Y = \beta_0 + \beta_1 X + \epsilon$, with $\epsilon$ normally distributed. For this simulation the mean squared error (MSE) for the Linear regression is `0.992` and `1.494` for the Tree. Figure 3 shows quite clearly that this is the kind of data where it will be hard to beat linear regression.

The linear regression also wins out on intuitiveness. The derivative of the line of best fit can be interpreted as the expected increase in variable $y$ as the $x$ variable increases. While the Tree's step function lacks this.

In this second simulation, the data have a non-linear relationship. The data was generated using 4 normal distributions with varying variance, in the four corners of a two dimensional space. With the datapoints of the North-West and South-East distributions coloured in red and the North-Eastern and South-Western points in blue.

Using classifiers instead of regressors for this simulation the error rates of the tree and linear classifier are `37.57%` and `51.03%` respectively. The interaction effect between the two dimensions is not captured by the linear model, while the regression trees naturally captures it.

# 4 Predicting grades

Next, I will apply Tree-based methods and also linear regression to a dataset to showcase differences in a more realistic situation. This dataset of student performance includes 649 students and looks at 33 variables.

## 4.1 Pruning

For all experiments on the dataset, I split the data into training (70%) and validation (30%) sets.

To prevent overfitting and ensure meaningful insights, it is crucial to use cross-validation and prune the regression tree. Without pruning, the model risks fitting to noise in the data, leading

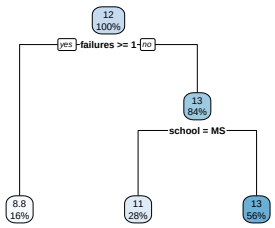


Figure 5: Smaller, better Tree

to misleading conclusions. While one will generally want to use the in built automatic pruning, manually pruning trees can provide valuable intuition about how regression trees tend to overfit. Instead of relying on the built-in 10-fold cross-validation to automatically select the optimal tree, I first experimented with manual pruning.

By growing a large tree with a complexity parameter of cp $= 0.01$, I obtained a tree with a training MSE of 4.521 and a validation MSE of 9.027, clearly indicating overfitting with the large tree. A smaller tree grown with cp $= 0.025$ resulted in a training MSE of 7.841 and a only slightly worse validation MSE of 8.651.

While it may seem counterintuitive that this simpler tree (Figure 5) outperforms the larger one, the larger tree fits the noise rather than the underlying signal, which explains its poorer generalization.

To find the best possible tree, one should employ proper automatic pruning. For this experiment I grew and pruned a large tree on the training data.

One challenge in pruning is that, even when using separate training and validation sets, searching for the optimal complexity parameter cp to minimize test error can lead to a situation where a particular subtree coincidentally performs very well on the test set by chance. Figure 6 shows how some subtrees can appear especially good for a particular value of cp. However, selecting this tree might still result in overfitting. To mitigate this, it's important to use a seperate validation and test sets, not allowing oneself to experiment around with the test set until one finds one tree that happens to perform especially on it by chance.

Sometimes it's so extreme that the optimal pruned tree using cross-validation, once it's evaluated

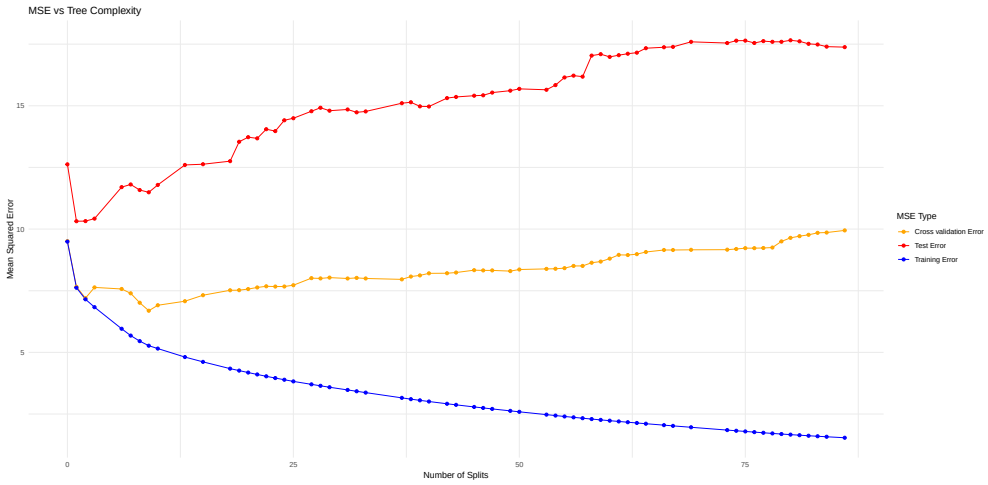


Figure 6: Pruning finds the lowest Test error

on the test set, gets an MSE on the training data of 5.272, but on the Test set MSE of 11.4923, clearly overfitting and just lucky during cross-validation.

Here one can see very well how the Cross-validation error helps against overfitting, it doesn't decrease monotonically, but still overfits to a large extent. The optimal tree using cross validation has 9 splits while the optimal one on the test error only has 1.

A different problem is that the results can vary quite widely. Different splits in training and test data might lead to MSEs on the test set for the pruned tree between 6.9 and 9.

An optimal value for textttcp which in this case is 0.028 and 0.035.

The pruned tree performs just one single split on failures, splitting the 15% of students who have previously failed the exam into one region (Estimate = 8.4) and the 85% of students with 0 failures into a second region (13 = estimate). This seems like something a human could have also done by hand.

The fact that the test error is lower than the training error should not be surprising; with only 1 split, the tree couldn't overfit even if it tried. It will always make almost the exact same split anyway. It's just chance if it will get a lower or higher error on the test set.

In extreme cases, the pruned tree selected via cross-validation might overfit, achieving a training MSE of 5.272 but a much higher test MSE of 11.492, indicating that it was merely lucky during cross-validation.

This highlights how cross-validation helps combat overfitting. As seen, the cross-validation error doesn't decrease monotonically, and the model can still overfit substantially. In this case, the optimal tree using cross-validation had 9 splits, while the tree minimizing test error had only 1 split.

Another issue is the variability in results. Different splits of the training and test sets can lead to test set MSEs for the pruned tree ranging from 6.9 to 9. The optimal cp values in these cases varied between 0.028 and 0.035.

Typically, the results look like this:

The pruned tree ends up making just one split based on the number of previous failures. It separates the 15% of students who had failed the exam before (with an estimated score of 8.4) from the remaining 85% who had no failures (with an estimated score of 13). This simple tree is something a human could likely deduce manually.

| Model | Training MSE | Valid or Test MSE |
|---|---|---|
| Complex Tree | 1.715 | 10.813 |
| Pruned Tree | 8.527 | 8.189 |

Figure 7: MSE values for different models

It is not surprising that the test error is lower than the training error; with only one split, the tree couldn't overfit even if it tried. Since it always makes almost the same split, any difference in error between the training and test sets is due to random chance.

## 4.2 Comparing Linear Regression, Regression Trees, and BART

After examining the tree's performance, it's informative to compare it to the results obtained from linear regression. These are the typical results one gets for Regression Trees and multivariate linear regression. The linear regression is not optimized in any way to counter overfitting. Even without any pruning, linear regression outperforms the tree by a wide margin:

Invoking Condorcet to assist us, we can use ensemble methods. Using BART with the default hyperparameters, that is, a burn-in and sampling period of 2000 iterations each and 200 Trees per iteration, results in the following MSEs for BART:

Depending on how the data is split into training and test sets, and other random factors, the numbers for all three models can vary quite widely. However, the single Tree is almost always outperformed by linear regression, and BART outperforms them both.

| Model | Training MSE | Validation MSE |
|---|---|---|
| Tree | 8.527 | 8.189 |
| Regr | 6.787 | 6.865 |
| BART | 5.777 | 6.523 |

Figure 8: MSE values for different models

In return, BART is far less easy to interpret than the other two methods. We did not attempt to improve the performance of the regression model.

Presumably, its performance can be increased in some way. But there are, of course, also other datasets where complicated ensemble methods significantly outperform linear methods.

# 5 Conclusion

Regression trees are powerful tools for handling non-linear and interactive effects, often outperforming linear regression in these scenarios. They are also very easy to interpret. However, trees require pruning to combat overfitting. Ensemble methods, by averaging independent trees or fitting trees on the residuals, can significantly improve results. BART, in particular, is a sophisticated method offering good results in many scenarios. While regression trees have their strengths, it's important to choose the right tool for each specific data analysis task. Clearly, there is a reason that there are "about 188,000 results" on Google Scholar when one searches for "Regression Tree" versus "about 4,320,000 results" when searching for "Linear Regression". However, especially random forests are quite popular.

The most obvious problem with simple regression trees is that they overfit immediately without doing anything useful. It seems unimaginable to be able to do only one split on a dataset this large, with that many variables. If students who drink more alcohol score slightly less in the Training set, a linear regression could incorporate this slight factor, and it might well be that this also applies in the Test set and improves model performance. If it's not a real effect and just noise, this slight coefficient won't increase Test error by a lot. Whereas the Regression Tree, after having made its initial useful split, can't help find the best optimal split, apparently hyper-focusing on some seemingly useful area that mostly turns out to be just noise.

While BART beats linear regression, it gives up on interpretability for the most part, while linear regression is very interpretable, in a similar way to regression trees. Arguably more so.

In conclusion, regression trees have their applications. As the simulations show, on very non-linear data, even a simple tree can outperform linear regression by a wide margin. However on real-life data, linear regression will often perform better and be more interpretable and useful. Perhaps trees shine only on data that has very strong interaction effects, more than the single dataset I considered.

Of course, one can always find datasets more suited for trees to work better than my data, and better methods to determine which is the best method to use.

This is a very important area of research, and clearly, there are still many open questions.

# References


Gérard Biau and Erwan Scornet. A random forest guided tour. *TEST*, 25:197–227, 2016.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Chapman & Hall/CRC, Boca Raton, 1984.

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

Marquis de Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. De l'Impr. royale, Paris, 1785.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R (Second Edition)*. Springer, 2021.

Y. V. Tan and J. Roy. Bayesian additive regression trees and the general bart model. *Statistics in Medicine*, 38(25):5048–5069, 2019.