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 (**?**). Since then many extensions have been developed such as Random Forets, a very popular Tree based ML method. (**?**) serves as a good introduction to that method. (**?**) serves as an excellent modern introduction to Regression trees and gives a gentile introduction to the topic and many extensions. (**?**) 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 minimum number of observations in each leaf node or a maximum tree depth. Figure **??** 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

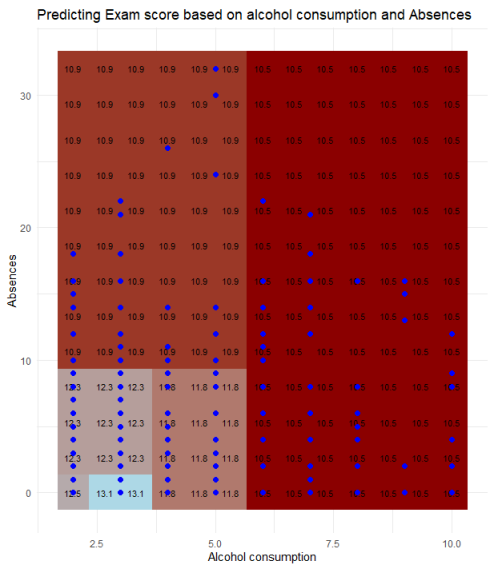

Figure 1: Visualization of Regression Tree's predictions

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 **??**, illustrates how a Regression Tree can overfit a dataset.

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 α 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 α, is increased and it becomes harder for each split to justify its existence. As one prunes a large tree by increasing α, 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

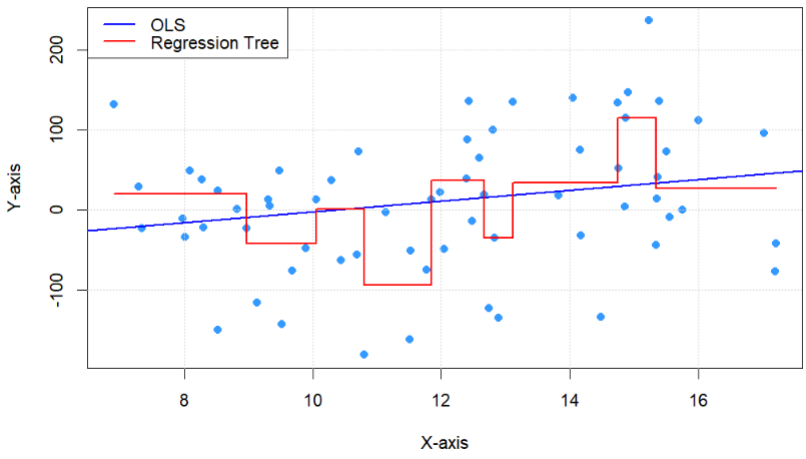

Figure 2: Regression Tree overfitting

performed automatically each time one grows a tree.

2.2 Ensemble Methods

While pruning can significantly improve the performance of individual regression trees, they often still underperform compared to other advanced machine learning methods. This limitation led to the development of ensemble methods, which combine many regression trees to create more robust and accurate predictions. As Condorcet's jury theorem shows, if predictors are uncorrelated and each predictor's answer is better than pure chance adding more predictors will increase the quality of the average of all answers (**?**).

Ensemble methods improve results by leveraging the "wisdom of crowds" effect. If each tree has, for example, a 70% chance of being correct, and the errors are uncorrelated, combining many trees will tend to improve overall accuracy. The two main approaches to creating ensembles are:

1. Growing many independent trees and averaging them. Random Forests are a very popular ML method that does this. 2. Growing trees on the residuals of the current tree model, as in boosting methods 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 (**?**).

I will go into slightly more detail for the BART method, as that is the extension 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. And ϵ_i is the noise term.

The BART algorithm proceeds by first generating e.g., 200 trees, without any splits. For each Tree j, it calculates the residual error R_{-j} by taking in all the current trees except j:

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

Then it proposes a change to that tree's structure, such as pruning a node, growing a node, or changing some splitting rule. It then accepts or rejects this modification based on its posterior probability. Which variable will be used is randomly selected to serve as the splitting variable, similar to random forests. If one averages many trees but does not ensure they are uncorrelated, most of them might make the same split, thereby eliminating their wisdom of crowds. 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) can be obtained not by simply taking the final sum of trees the last iteration produced, but by taking the average over all sample iterations, discarding the burn-in iterations.

A more complete exposition can be found in Chipman et al. (2010). of the BART method and how it's Bayesian.

The algorithm then uses Markov Chain Monte Carlo (MCMC) sampling to draw from the posterior distribution of the model parameters.

BART uses a Bayesian approach, specifying priors on the tree structures, terminal node parameters, and error variance. Nodes at depth d are nonterminal with probability $\alpha(1+d)^{-\beta}$. This prior on shallow trees makes them weak learners, more suited to being average. Each tree explains a small different part of f. It also has other priors that inform how it assigns estimates to the terminal nodes, but that is beyond the scope of this paper.

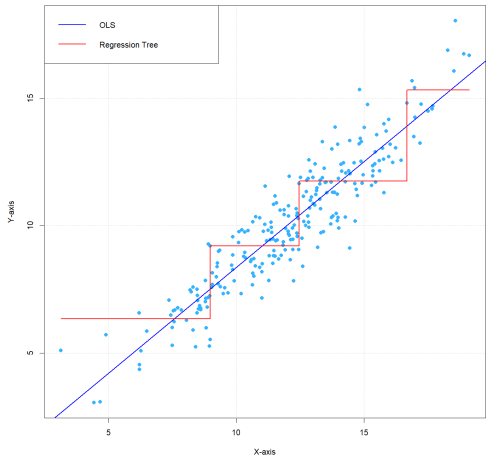

Figure 3: Linear Relation between Variables

Default values for these hyperparameters of $\alpha = 0.95$ and $\beta = 2$ are recommended by (**?**). Default values generally provide good performance, but optimal tuning can be achieved via cross-validation.

Ensemble methods like Random Forests and BART have shown remarkable success in various applications, often outperforming individual regression trees and even many other machine-learning techniques. They offer a powerful toolset for modelling complex relationships in data, providing both predictive accuracy and, in the case of BART, measures of uncertainty.

While individual regression trees offer interpretability and the ability to capture non-linear relationships, their tendency to overfit limits their effectiveness. Pruning helps mitigate this, but ensemble methods take the concept further, leveraging the strengths of multiple trees to create highly accurate and robust models.

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. Here, I will compare Linear regression and regression trees on linear and non-linear data.

This first simulation shows where linear regression excels and trees aren't especially useful. The data follows a simple linear relationship, exactly as the normal least squares regression assumes it does. The data was generated as: $Y = \beta_0 + \beta_1 X + \epsilon$, with epsilon normally distributed. I ran

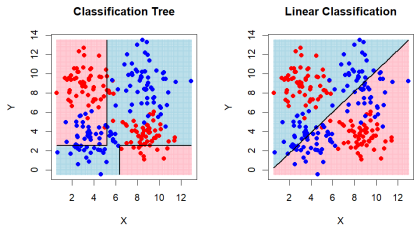

Figure 4: Complex, overfitting Tree

the simulations 400 times with regression trees, having four terminal nodes. And compared the Mean Squared Error (MSE) for both methods. The mean squared error for the Linear regression is texttt0.992 and for the tree it is: texttt1.494. The attached figure quite clearly shows how linear regression is more suited to this kind of data.

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 step function of the regression tree doesn't serve any obvious function. It's just a worse regression.

In this second simulation, the data have a non-linear relationship. In this case, I used classification trees instead of regression trees, as this allows using colour as our third dimension instead of having to use a three-dimensional plot.

The data was generated using 4 normal distributions with varying variance, in the four corners of the plot. With the North-West and SE being red and the NE and SW distribution producing blue data points.

In this case, the error rates of the tree and linear classifier are texttt37.57% and texttt51.03% respectively. If this weren't a classification but a regression problem the results would be much the same. The interaction effect here is not captured naturally by the linear model, while the regression trees can naturally capture interaction effects.

This simulation also shows how Trees capture interaction effects naturally; the fact that a large X value is only indicative of the observation being Red if the Y value is small is naturally captured by the tree.

One could try to showcase more differences between the two methods now, especially on a simulation with more dimensions, seeing how an optimally pruned tree compares against multiple linear regression. But it is more interesting to do this on a real dataset, so we will leave the simulations behind us for now.

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.

One can easily train a large tree on the dataset and find the variable importance, to then be better at further analyses. For example, if one does not remove the variables representing previous test scores, practically all the model does is predict the final test score based on the previous test score, and all other variables are practically irrelevant.

4.1 Pruning

The most important thing to do is to use cross-validation and prune your regression tree; otherwise, nothing of interest will be learned as one will have just overfitted on the dataset.

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

It can be useful to prune some trees manually to get an intuitive feeling for how Regression Trees overfit. Instead of using the sophisticated built-in option to automatically perform 10-fold cross-validation and select the best-performing tree, I did it by hand first.

Growing a large tree with the complexity parameter (which only allows splits if they cross a certain threshold in error reduction) cp = 0.01, we get a large tree with a Training MSE of 4.521407 and a Validation MSE of 9.026646. So clearly, this tree is overfitting to a significant degree.

Growing a small tree with cp = 0.025 and minsplit = 5 will result in a tree with a Training MSE of 7.840784 and a slightly higher Validation MSE of 8.651296.

It's somewhat disappointing that this super simple tree outperforms the larger one, but the larger tree is just fitting to the noise of the data, not the real signal.

To find the best possible tree, one should, of course, employ real pruning, automatically evaluated by the machine. This time, I first grew a massive tree that would totally overfit the training data.

One difficulty here is that even if one is using two datasets to combat overfitting, trying to find the optimal textttcp value that will minimize the error on the test set may result in a situation where, by chance, one of the many different stages of pruning happens to produce a very good result.

This plot nicely shows how pruning can find the optimal Tree, that doesn't overfit too much but

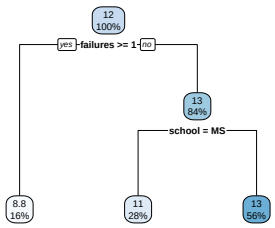

Figure 5: Smaller, better Tree

still captures some of the data's information.

In the plot one can nicely see how it could happen by chance that one tree is especially good for some number and if one then chooses that as the tree one is going to work with one overfitting still. To combat this, I first trained the complex tree on the training set and found the optimal value for textttcp using cross-validation, also on the training set. Only then did I evaluate its performance on the test set.

Sometimes it's so extreme that the optimal pruned tree using cross-validation, once it's evaluated on the test set, gets an MSE on the training data of 5.27211417154088, but on the Test set MSE of 11.4923047228304, 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 most typical results will look something like this:

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

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

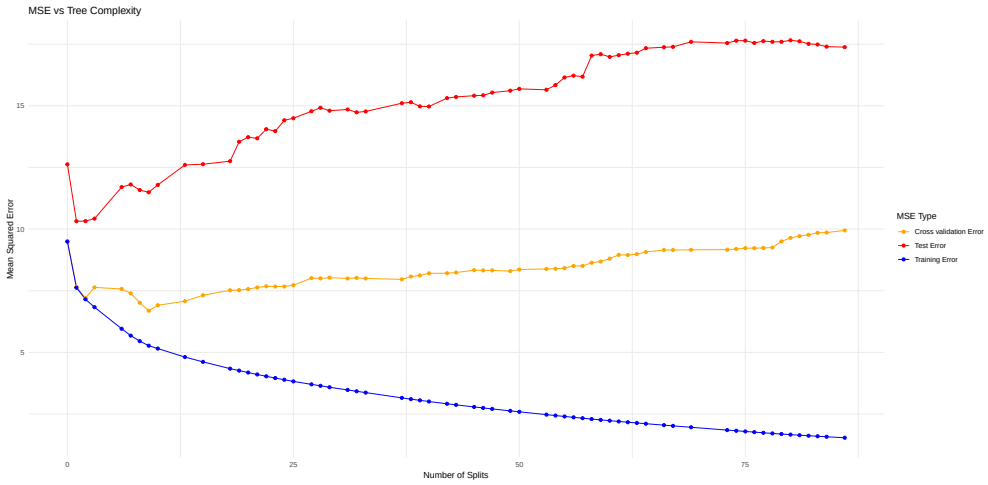

Figure 6: Pruning finds the lowest Test error

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.

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

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

BART outperforms them both.

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.