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

Modern economic research heavily relies on linear regression. To conduct effective economic research, it is crucial to understand the strengths and weaknesses of linear regression and where it outperforms or is outperformed by other methods. This paper explores Regression Trees as an alternative to linear regression. In it, I will explain the theory behind regression trees and compare them to linear regression using a series of simple simulations, focusing on interaction effects and pruning. I will also highlight how ensemble methods, a kind of extension to regression trees, can enhance their performance. Subsequently, I will compare the different methods on a dataset of student performance, emphasizing the strengths and weaknesses of linear regression and different tree-based methods. While linear regression is arguably the most frequently used method by empirical economists and can be valuable in many situations, it is essential to examine its limitations and areas where alternative methods excel. Regression trees have emerged as a powerful machine-learning technique for predictive modelling, with numerous extensions greatly improving their utility. Regression trees offer an alternative approach that can be beneficial in various scenarios. This paper will discuss their advantages over traditional linear regression methods, cover the fundamentals of regression trees, compare them with linear regression, address the issue of

overfitting, and introduce advanced ensemble methods such as Bayesian Additive Regression Trees (BART).

In this paper, I will first give a brief exposition on Regression Trees, their theory, extensions, and applications and contrast them with linear regression. I will perform easily interpretable simulations comparing Regression Trees and Linear Regression. Show how Regression Trees can be used on a Dataset of information about students and their test performance.

Given the widespread use of linear regression, it is particularly important to identify where it performs well and where it falls short, as well as where other methods can improve upon it. Due to their interpretability, trees may also serve well in educational roles.

Linear regression performs poorly on many types of data, particularly those with non-linear relationships and interaction effects. Regression Trees can be useful in many situations where linear regression falls short.

Since (Breiman et al., 1984) presented the CART algorithm there has been a lot of work on various Tree based methods.

(James et al., 2021) serves as an excellent modern introduction to Regression trees, while while (Tan and Roy, 2019) give a deeper dive on BART a powerful ensemble method. Random forests, a very popular Tree based ML method are explained very well in(Biau and Scornet, 2016).

Simulations are an excellent way of investigating different aspects of one statistical method, as they allow one to create a dataset with just the kind of noise one wants. They allow creating an ideal setting.

The simulations show how regression trees outperform linear regression on datasets with non-linear relationships. And how pruning can help find the optimal trade-off between overfitting and not capturing enough of the signal. While applying the methods to real-world data can showcase how trees capture interaction effects and how pruning works I discovered. Also the complications of working with real-world data, as opposed to simulations.

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, $\hat{y}$ simply takes on the mean of all observations in that region. Each branching of the tree divides the predictor space into two regions, for numerical variables typically of the form $\{x \leq c\}$ or $\{x > c\}$. In each resulting region, the predicted value $\hat{y}$ is simply the mean of all observations within that region.

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

Regression Trees offer several advantages over traditional linear regression methods. They can capture non-linear relationships between predictors and the response variable without requiring explicit specification of these relationships. Furthermore, they naturally account for interaction effects. For instance, if being blonde typically increases pay, but only for women, a Regression Tree will often automatically split along gender and then, for women only, along hair colour. This ability to model complex interactions without manual specification is a significant strength of the method.

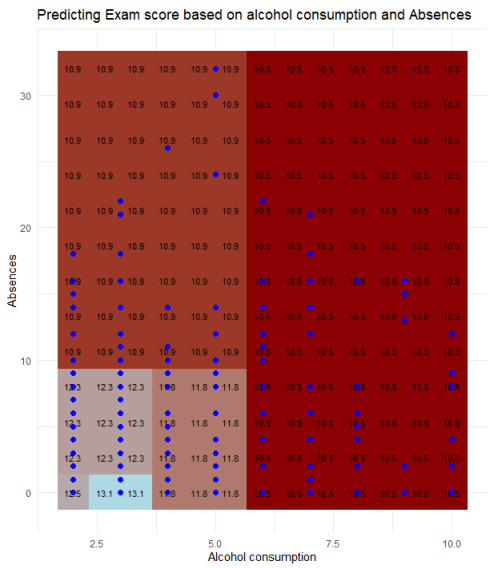

Figure 1: Visualization of Regression Tree's predictions

One other major benefit of Trees over Linear Regression is that they capture interaction effects naturally. If for example being blonde typically increases pay, but only for women, a Regression Tree will often naturally split along gender and then, for women only, along hair colour. Whereas in a linear Regression the researcher would have to manually include terms for all interaction effects they want to study. However, Regression Trees are not without challenges. One of the primary issues is their tendency to overfit the data, especially when allowed to grow deep. Unlike linear regression, where overfitting primarily occurs with high-dimensional data, regression trees can overfit even with low-dimensional data. This is because they have the flexibility to create very specific rules that may capture noise in the training data rather than true underlying patterns.

2.1 Pruning

To address the issue of overfitting and improve overall performance, researchers have developed methods to "prune" regression trees. One of the most effective techniques is cost complexity pruning. This process involves several steps:

First, a large tree is grown that is likely 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:

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

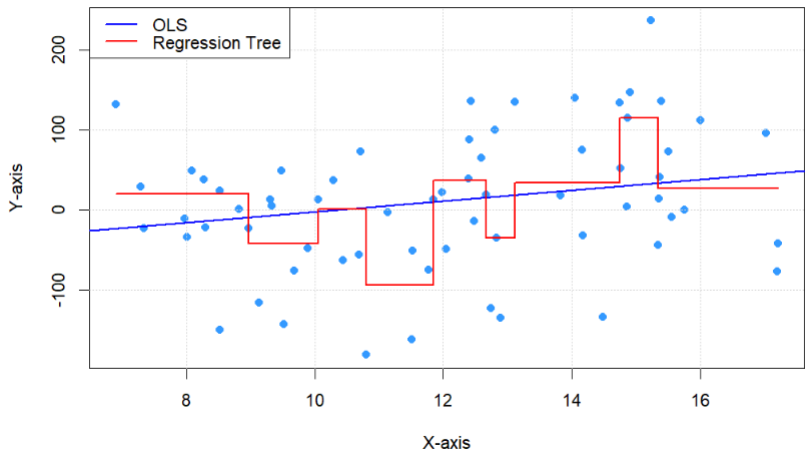

Figure 2: Regression Tree overfitting

Where $|T|$ is the number of terminal nodes in the tree. For each value of α, the algorithm finds the subtree that minimizes this cost function. The optimal α is then selected using cross-validation, which helps achieve a good trade-off between bias and variance. The larger the cost complexity parameter α, the better each split has to be to justify its existence. Recursively the least important split is removed. As one prunes a large tree by increasing α, the tree is naturally pruned in a predictable and well-behaved fashion, allowing one to select the preferred tree.

Instead of evaluating a model on the data we trained it on, we evaluate it on a separate set. In the `rpart` library cross-validation is performed, and the error for all different trees is produced right away, allowing one to select the tree that performs best on the cross-validation error. This lets one achieve a good trade-off between bias and variance. A large α results in very small trees, while a small α results in larger trees.

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 (de Condorcet, 1785).

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 (Biau and Scornet, 2016).

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.

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

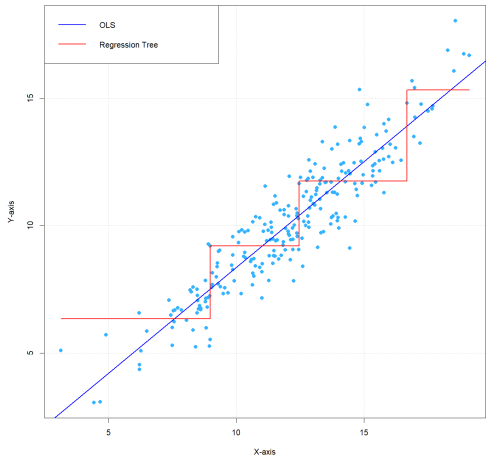

Figure 3: Linear Relation between Variables

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.

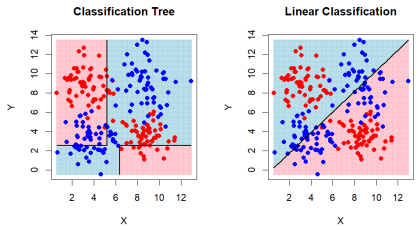

Figure 4: Complex, overfitting Tree

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

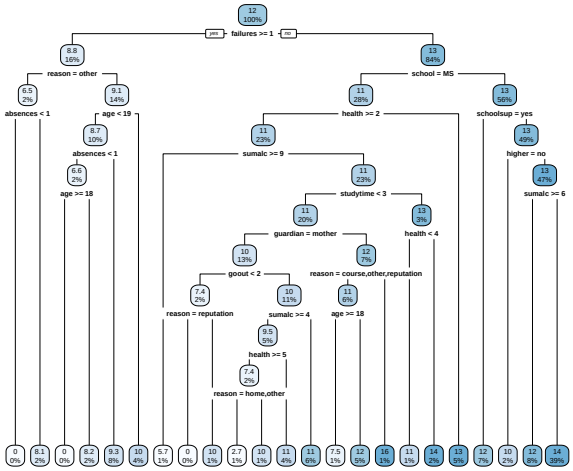

Figure 5: Complex, large Tree

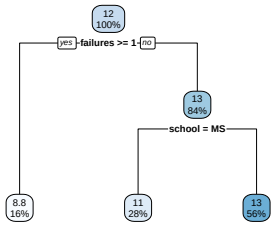

Figure 6: Smaller, better Tree

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,

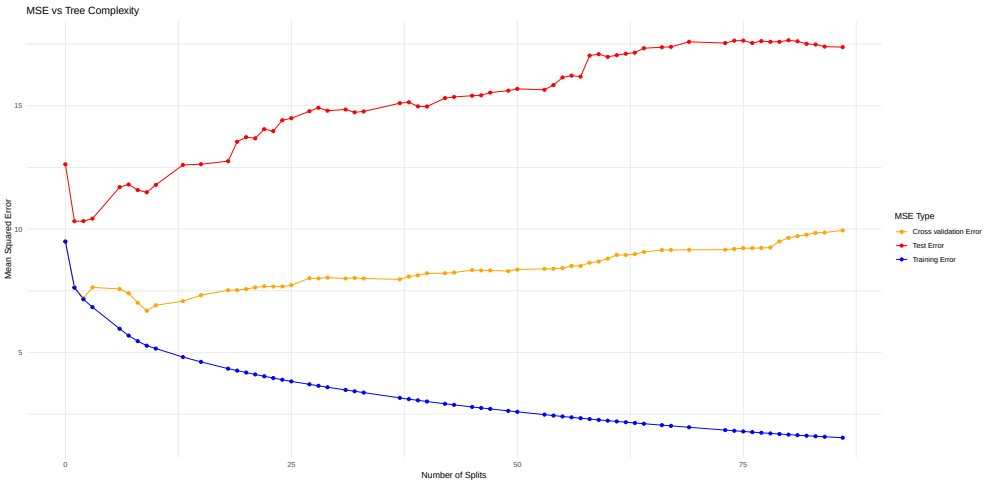

Figure 7: Pruning finds the lowest Test error

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 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

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

Figure 8: MSE values for different models

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.

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 9: 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.