University of Mannheim – Master Thesis

Weighted Splitting in Random Forests

Christian Hilscher [*]
University of Mannheim – Economics Faculty

Supervisor: Prof. Dr. Cathrine Aeckerle-Willems

September 3, 2021

[*]Enrolment Number: 1570550

Contents

1 Introduction

Regression trees and random forests are the cornerstones of most applied ML algorithms. They are comparatively quick, accurate, and very easy to interpret. The CART (Classification and Regression Tree) algorithm is the basis of regression trees and random forests. The theoretical properties of these methods have come into focus lately, especially given the increased use in applied work. Since it has proven difficult to formally analyze the CART algorithm's behavior, most approaches consider simplified versions of the algorithm. In addition, these analyses have focused on the infinite sample mechanisms of the estimator.

In the beginning, tree nodes contain many data points, making the asymptotic framework an accurate description. As the tree is grown deeper and the original data is split more often, the number of observations within the nodes decreases, rendering the infinite sample setting inadequate. We will show that the crucial point is the impact of error terms on the tree building process. As long as the infinite sample setting is an accurate description, the random forest can differentiate well between signal and noise. However, with fewer data points, the error terms start influencing the tree structure and, therefore, the predictions.

Bridging two, up to now distinct, phenomena of regression trees allows us to gain insights into this switch from infinite to finite sample properties. By considering the location of split points, we can improve on the original CART algorithm. A weighted splitting rule allows the algorithm to differentiate between signal and noise better, leading to more accurate predictions. The significant advantage of the proposed approach is that it allows the regression tree to retain its biggest strength: local adaptability. Previous attempts to curtail the influence of error terms on the tree structure all constitute hard constraints that affect the whole tree at each node in the same way. In contrast, weighted splitting adapts to the strength of the signal within a specific node, making it a more flexible approach. It allows nodes with a strong signal to exploit it further and quickly terminates nodes in a region with relatively low signal strength.

The consistency of the weighted splitting approach will be shown by generalizing the proof of Klusowski (2019). Moreover, introducing weights into the CART algorithm allows connecting two distinct random forest estimators. By setting the weights equal to zero, one gets the standard CART algorithm. On the other hand, imposing a high weight yields a perfectly randomized tree, proposed by Cutler and Zhao (2001). Thus, the proposed splitting rule generalizes previous approaches. In addition, the theoretical advantages also manifest themselves in an empirical setting. The weighted splitting estimator performs significantly better than the CART algorithm and other popular tuning parameters.

The remainder of the paper is structured as follows: Section 2 gives a detailed account of regression trees and random forests, while Section 3 introduces concepts that are helpful later on. Section 4 focuses on the transition from asymptotics to finite sample properties and its effects on the regression tree. Based on this, Section 5 introduces the weighted splitting approach, and Section 6 evaluates its performance.

2 Regression Trees and Random Forest

This section serves as groundwork for the upcoming analysis. Since there are multiple versions and descriptions of regression trees and random forests, the aim is to clearly state the tree-building process and further aggregation into random forests.

2.1 Preliminaries

We start with the training data $\mathcal{D} = \{(\mathbf{X}_1, Y_1), ..., (\mathbf{X}_n, Y_n)\}$ where $\mathbf{X}_i \in [0,1]^d$ and $Y_i \in \mathbb{R}$ is a continuous response variable for $1 \leq i \leq n$. The j^{th} coordinate of the input matrix $\mathbf{X}$ is denoted by X_j. We assume $Y_i = m(\mathbf{X}_i) + \epsilon_i$ with $m(\mathbf{x}) = \mathbb{E}[Y|\mathbf{X} = \mathbf{x}]$ being an unknown regression function and ϵ_i is an i.i.d. error. The main goal is to estimate $m(\mathbf{x})$ and make predictions $\hat{Y}(\mathbf{x})$. The accuracy of the predictions will be determined by the mean squared error $\mathbb{E}[(\hat{Y}(\mathbf{X}) - m(\mathbf{X}))^2]$ which we seek to minimize.
Regression trees are often used in setups with many input variables, of which only a few determine the outcome variable Y. Therefore, we assume that $m(\mathbf{x}) = \mathbb{E}[Y|\mathbf{X} = \mathbf{x}]$ depends only on a small subset $\mathcal{S} < d$ features, which we call strong features. On the other hand, all weak (noisy) variables $\{X_j : j \notin \mathcal{S}\}$ are independent of Y. The terms weak and noisy variables are used interchangeably.

2.2 Regression Tree

A regression tree partitions the original data into smaller, more homogeneous subgroups. It does so by repeatedly splitting the data. The very first node of the tree is a rectangular cell on $[0,1]^d$, containing all observations. At each step, one variable and one split point are chosen, determining how the data will be partitioned. Consider the case of node $\mathbf{t}$, where one splits on variable X_j, and s is the split point. The data will be separated into a left and right child node: $\mathbf{t}_L = \{\mathbf{X} \in \mathbf{t} : X_j \leq s\}$ and $\mathbf{t}_R = \{\mathbf{X} \in \mathbf{t} : X_j > s\}$. Then we take $\mathbf{t}_L$ as our new node, find the optimal X_j and s, and split it again. The same happens to $\mathbf{t}_R$ such that each node is being split over and over, and the tree depth k increases. This procedure is repeated until a specified stopping criterion. Each node of the tree is then a rectangular subset of $[0,1]^d$, and this partition becomes smaller with every split.
Figure 1 shows this process schematically.

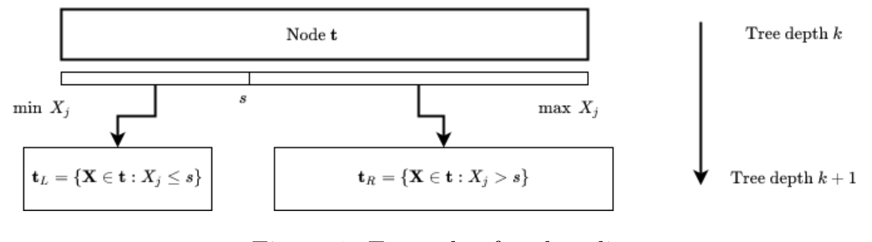
Node $\mathbf{t}$
Tree depth k
min X_j
s
max X_j
$\mathbf{t}_L = \{\mathbf{X} \in \mathbf{t} : X_j \leq s\}$
$\mathbf{t}_R = \{\mathbf{X} \in \mathbf{t} : X_j > s\}$
Tree depth $k+1$

Figure 1: Example of node split

The question arising now is how to choose the splitting dimension and the split point. Breiman et al. (1984) propose the CART algorithm, which became the standard way to construct regression trees.

The partition into homogeneous groups is achieved by minimizing the within-node variance at each step. Thus, the CART algorithm aims to reduce the variance of the response variable Y within a node by choosing the best splitting dimension and location.
The variance of a node $\mathbf{t}$ is given by

$$\Delta(\mathbf{t}) = Var[Y \mid \mathbf{X} \in \mathbf{t}].$$

To assess the quality of different split points as dimensions, we compare them using a measure called *decrease in impurity*. The decrease in impurity for any variable X_j and split point s is given by

$$\Delta(s, \mathbf{t}) = \Delta(\mathbf{t}) - [P(\mathbf{t}_L)\Delta(\mathbf{t}_L) + P(\mathbf{t}_R)\Delta(\mathbf{t}_R)]. \tag{1}$$

$\Delta(s, \mathbf{t})$ measures the reduction in variance from the original node by subtracting the variance of the two child nodes, weighted by their probability content. The conditional variances of the child nodes are

$$\Delta(\mathbf{t}_L) = Var[Y \mid \mathbf{X} \in \mathbf{t}, X \leq s], \quad \Delta(\mathbf{t}_R) = Var[Y \mid \mathbf{X} \in \mathbf{t}, X > s]$$

and the probabilities of an observation falling into the left and respectively right node are given by

$$P(\mathbf{t}_L) = \mathbb{P}[X \leq s \mid \mathbf{X} \in \mathbf{t}], \quad P(\mathbf{t}_R) = \mathbb{P}[X > s \mid \mathbf{X} \in \mathbf{t}].$$

$P(\mathbf{t}_L)$ also corresponds to the distribution function of $X \leq s \mid \mathbf{X} \in \mathbf{t}$ and the density function is given by $\frac{\partial P(\mathbf{t}_L)}{\partial s} = p(\mathbf{t}_L)$.
Maximizing (1) with respect to s then yields the split point, which in turn partitions the data such that the resulting child nodes are as homogeneous as possible. In other words, the more similar the observations within each node are, i.e., the lower the variance within each of the child nodes, the higher the decrease in impurity. Iterating over all input variables contained in $\mathbf{X}$ then allows to choose the splitting dimension X_j and the respective split point s with the largest impurity reduction for node $\mathbf{t}$.

The empirical counterpart to (1) is given by

$$\hat{\Delta}(\mathbf{t}) = \frac{1}{N(\mathbf{t})} \sum_{\mathbf{X}_i \in \mathbf{t}} (Y_i - \bar{Y}_{\mathbf{t}})^2$$

where $\bar{Y}_{\mathbf{t}}$ is the sample mean of the responses and $N(\mathbf{t})$ the number of observations in node $\mathbf{t}$ respectively. In the same way, the empirical decrease in impurity is defined as

$$\hat{\Delta}(s,\mathbf{t}) = \hat{\Delta}(\mathbf{t}) - [\hat{P}(\mathbf{t}_L)\hat{\Delta}(\mathbf{t}_L) + \hat{P}(\mathbf{t}_R)\hat{\Delta}(\mathbf{t}_R)] \tag{2}$$

with $\hat{P}(\mathbf{t}_L) = \frac{N(\mathbf{t}_L)}{N(\mathbf{t})}$ and $\hat{P}(\mathbf{t}_R) = \frac{N(\mathbf{t}_R)}{N(\mathbf{t})}$ being the fractions of observations falling into the left and right child node.

The CART algorithm then chooses the split dimension and a split point by finding

$$\arg\max_{j} \ \arg\max_{\hat{s}} \ \hat{\Delta}(j,\hat{s},\mathbf{t}).$$

It first calculates the highest decrease in impurity for every possible split point for a variable j. Then it takes the maximum over all splitting dimensions.
The tree continues splitting until a pre-specified stopping criterion, usually when fewer than v observations are left in a node. In the original CART algorithm $v = 1$, meaning that each node is split until it contains only one observation. If the threshold is reached, the node is designated a *terminal node* and is not split further. The prediction for a terminal node is given by the sample mean of all observations falling into that node: $\hat{Y} = \bar{Y}_{\mathbf{t}}$. The exact procedure on how a node is split is summarized in the following algorithm.

Algorithm 1: Node-Splits

```
Input: node $\mathbf{t}$, number of observations $n_{\mathbf{t}}$ in node $\mathbf{t}$, $mtry \in \{1,\dots,d\}$, $v \geq 1$
if $n_{\mathbf{t}} > v$ then
    Draw random sample of $mtry$ split dimensions
    Set $max\ decrease = 0$
    for $j = 1,\dots,mtry$ do
        $tmp\ decrease = \max_s \ \Delta(j,s,\mathbf{t})$
        if $tmp\ decrease > max\ decrease$ then
            $tmp\ decrease = max\ decrease$
            $split\ dim = j$
            $s = argmax_s \ \Delta(j,s,\mathbf{t})$
        end
    end
    Assign $\mathbf{t}_L = \{\mathbf{t} : X_j \leq s\}$ and $\mathbf{t}_R = \{\mathbf{t} : X_j > s\}$
    Set $n_{\mathbf{t}_L}, n_{\mathbf{t}_R}$ = number of observations in $\mathbf{t}_L, \mathbf{t}_R$
    Call the same algorithm recursively on $\mathbf{t}_L$ and $\mathbf{t}_R$
else
    Make $\mathbf{t}$ a terminal node
    Estimator for node given by $\hat{Y}_{\mathbf{t}} = \frac{1}{n_{\mathbf{t}}} \sum_{i:Y_i \in \mathbf{t}} Y_i$
end
```

While single regression trees have been empirically shown to have relatively low biases, they exhibit high variance. This behavior is especially pronounced when the stopping criterion is set such that each terminal node contains only one observation, i.e., $v = 1$. Splitting up to that point makes the tree very unstable, and when faced with different errors, the tree looks markedly different. Thus, while being a local predictor, the high variance of the estimator is one drawback of this approach. One possibility to mitigate this problem is to make the tree stop earlier via a pre-specified stopping criterion. Imposing that each terminal node has to contain at least $v > 1$ observations leads to multiple observations in the terminal nodes and reduces the variance of the tree. With an increasing v, the tree becomes more shallow. Naturally, one faces another trade-off with this imposition: The higher v, the more observations in a terminal node, making the estimator less local and resulting in a higher bias.

2.3 Random Forest

Another remedy to reduce the variance of a regression tree while trying to keep the favorable property of low bias was introduced in Breiman (2001). Building multiple trees and averaging their predictions can lower the variance significantly while keeping the bias relatively low. A random forest is a collection of n_tree, independently built regression trees. To achieve a lower variance, the individual trees must be sufficiently distinct from each other. If all trees would look precisely the same, then there is no benefit over a single regression tree because the impact of error terms cannot cancel itself out. To de-correlate the trees, only a subset of the original data is used for each tree, obtained by bootstrapping. Introducing other differences between trees is possible by influencing the selection of variables to split on. Considering only a randomly determined subset $mtry \leq d$ of all variables as possible split points at each node restricts the available choices, and therefore, each tree is distinct. This randomization process is described by a sequence $\{\Theta_l\}_{1 \leq l \leq n\_tree}$ which governs the tree building process. More specifically, it determines the draws from the bootstrapping and the splitting dimension if $mtry < d$. If $mtry = d$, all variables are considered as a possible splitting dimension.
The prediction for a single data point is then made by averaging over the predictions of all trees. Combining multiple, independent estimators makes the random forest an ensemble estimator, which constitutes the most popular class of ML algorithms. The procedure for arriving at a prediction for a generic point $\mathbf{x}$ is given by:

Algorithm 2: CART - Random Forest Prediction at point $\mathbf{x}$

```
Input: Training set $\mathcal{D}$, number of trees $n\_tree$, $mtry \in \{1, \dots, d\}$, $v \geq 1$, $\mathbf{x}$
for $t = 1, \dots, n\_tree$ do
    Assign randomization process $\Theta_t$
    $\mathcal{D}_t$ = bootstrapped data from $\mathcal{D}$
    Create node $\mathbf{t}_0 = \mathcal{D}_t$ as beginning of tree
    Split node $\mathbf{t}_0$ recursively according to Algorithm 1 and $\Theta_t$
    Predict $\hat{Y}_{\mathbf{t}}(\mathbf{x})$ for tree $t$
end
Prediction is given by average over trees: $\hat{Y}(\mathbf{x}) = \frac{1}{n\_tree} \sum_1^{n\_tree} \hat{Y}_{\mathbf{t}}(\mathbf{x})$
```

2.4 Randomization and Consistency

The theoretical properties of random forests have been studied quite extensively in the last couple of years. The challenges mainly revolve around the randomization process $\{\Theta_l\}_{1\leq l\leq n\_tree}$ and its data-dependency. Splitting points and dimensions are determined by underlying data, which in turn is influenced by Θ_l. Varying from tree to tree, the randomization process complicates the analysis. For this reason, the literature has made simplifications to the original CART algorithm and the way trees within a forest are constructed.
Conditions for consistency of data-independent partitioning estimators have been proposed by Stone (1977). Thus, the first approaches to prove consistency for regression trees all made adaptions such that the splitting algorithm is data-independent. Data-independence means that the mechanism determining the tree structure Θ_l does not depend on the underlying data. Cutler and Zhao (2001) achieve this by first randomly choosing the splitting variable and afterward randomly selecting the split point. Another way to abstract from the data was analyzed by Breiman (2004), where the splitting dimension is chosen randomly, and the split point is always the median observation. These alterations make the estimator independent of the data such that the probability of a split happening on a particular variable is not influenced by the underlying data. This way, the tree structure is again data-independent, and consistency can be shown relatively easily in both cases.
Biau (2012) was the first to show consistency results for an algorithm that more closely resembles the original random forest procedure and is data-dependent. The respective decrease in impurity determines the split points as in the original CART approach. However, in Biau's setup, the split points do not maximize the empirical decrease in impurity but rather the asymptotic one. A second dataset is needed to consistently estimate these asymptotic values since using the same data for estimation and locating the split points would lead to inconsistencies. The simplification of this approach lies in de-coupling the tree-building process Θ_l from the observations which serve as a predictor. Since one dataset is used for determining the tree structure and another one for making the predictions, the errors across the two datasets are independent and thus have no impact. Building on these results, Scornet et al. (2015) can prove consistency for a data-dependent random forest, albeit only for additive models. The restriction of allowing only an additive structure can be traced back to an influential paper by Lin and Jeon (2006), who reformulate the random forest as an adaptive nearest neighbor estimator. In an additive model, the influence of one variable only depends on that variable itself since there are no interactions terms, which render the analysis of the process more amenable.
Klusowski (2020) adds to the literature by proving consistency for a more general class of response surfaces beyond the additive structure used in Scornet et al. (2015). Overall the study of theoretical properties of the random forest algorithm has proved challenging, and all the adaptions to the original algorithm prevent simple comparisons of approaches.

2.5 Connection to kNN-Estimator

An essential contribution to the literature has been the reformulation mentioned above of a random forest as an adaptive nearest neighbor estimator by Lin and Jeon (2006). The neighborhood of a point x_0 can be thought about as all points in the same terminal node as x_0. The adaptive component comes from the fact that the number of neighbors can differ depending on the neighborhood. This feature allows the random forest to adapt to the regression function and exploit local changes. Looking at a regression tree through the lens of a k-nearest neighbor (kNN) estimator gives us the benefit of studying specific properties in a more familiar setting. The properties of kNN estimators are better understood and more straightforward to describe than those of a regression tree. Especially when trying to understand which variables the regression tree splits the data on, the perspective from the kNN estimator proves insightful. Lin and Jeon show that a regression tree splits more often on variables that are "important," meaning that they explain a large share of the variance of the outcome variable. To intuitively understand their argument, assume the following simple setup:

$$Y = g(\mathbf{X}) + \epsilon$$
$$g(\mathbf{X}) = \sum_{j \in \mathcal{S}} a_j X_j$$

Here, Y is a simple additive model of our strong variables and ϵ an i.i.d. error term. Assuming further that all $X_j \sim \mathrm{U}[0,1]$, the effect of a particular X_j on Y is given by $|a_j|$. Variables with a large $|a_j|$ influence the outcome variable Y more than those with a low value.

Consider the case where we want to estimate $\hat{Y}_0$ for a given $\mathbf{X}_0$ and the corresponding actual outcome Y_0. The goal then is to characterize the neighborhood, aiming to best approximate $g(\mathbf{X})$ around $\mathbf{X}_0$. This is done by choosing intervals for each X_j such that the area within all those intervals consists of n points and whose average constitutes our estimate $\hat{Y}_0$. Let us denote the interval lengths for each variable q_j. Lin and Jeon (2006) show that in equilibrium, $q_j|a_j| = C \ \forall j$ for any constant C. In other words, in the optimal case, all variables have the same importance within the neighborhood.
Turning to our example with $X_j \sim \mathrm{U}[0,1]$, this implies that variables with a high $|a_j|$ have shorter intervals q_j. Intuitively, this means that for variables that have a significant effect on Y, the interval around $\mathbf{X_0}$ should be relatively small, since being far away from $\mathbf{X_0}$ would push our estimate too far away from the actual outcome Y_0. One implication of the smaller intervals is that the estimator is less likely to underfit the data in that dimension. On the other hand, allowing for a wider interval for variables with low importance mitigates the possibility of overfitting the data.

We can now take these insights and apply them to our random forest setup. The neighborhood around $\mathbf{X}_0$ are all observations that are in the same terminal node as $\mathbf{X}_0$. The side lengths q_j correspond to the length of X_j, which defines the terminal node. One interesting aspect of this example is that the optimal q_j can vary from node to node, showcasing the local adaptivity of the random forest estimator.
The other, for this exposition more relevant insight, is that variables that have high importance have smaller side lengths. To get smaller side lengths, the algorithm must split more often on those particular variables. Put differently, the random forest will choose strong variables more often to split on than weak ones. This ties into Klusowski (2019), who finds a positive relationship between the selection frequency of X_j and the respective importance of variable j measured by its MDI (Mean

Decrease in Impurity).

The simple additive model above can explain intuitively why strong variables are more often chosen as splitting dimensions. In general, the relative contribution of the variable j to the total variance of Y plays a vital role in the number of times it is selected as a splitting dimension. It is important to stress that this finding holds only in an infinite sample setting. While this is the case, the regression tree can clearly differentiate between strong and weak variables and chooses strong variables as splitting dimensions.

2.6 Simulation Study

The subsequent sections will be supported by simulating artificial data, and highlighting certain properties of regression trees and random forests. One advantage of which we will make extensive use is keeping specific parameters constant and thereby examining the direct impact of subtle changes, for example increases in the error term variance. The regression function is taken from Friedman (1991)

$$m(\mathbf{X}) = 10\sin(\pi X_1 X_2) + 20(X_3 - \frac{1}{2})^2 + 10X_4 + 5X_5 \tag{3}$$

which is used throughout the literature to test the performance of nonparametric estimators. The structure of (3) lends itself to analysis due to its interaction effects between variables and mixture between additive and non-additive components. All regressors are drawn from an uniform distribution $U[0,1]$ and we have five strong variables for which $j \in \mathcal{S}$. The remaining dimensions of $\mathbf{X}$ are not used in the construction of the regression function. Since, in total, we have d regressors, the remaining $d-5$ are noisy dimensions. The outcome variable is given by $Y = m(\mathbf{X}) + \epsilon$ where the errors are drawn from a normal distribution with $N(0, \sigma_\epsilon^2)$.

The estimator used is a random forest built according to the original CART algorithm. Each node is split by choosing the dimension and split point yielding the highest decrease in impurity. Nodes are split until only one observation is left in each terminal node, whose Y-value then constitutes the estimate for that node. We will investigate the effects of changing two parameters in the next section: how changing the error term variance affects the location of split points and the impact of an increase in noisy dimensions. The former is achieved by increasing σ_ϵ^2 and the latter by raising the number of dimensions d. Since $m(\mathbf{X})$ is determined by five variables, a higher dimension of the regressor matrix means increasing the number of noisy variables.
The advantage of simulated data here is that we can be precise about which parameters we change and which ones we keep constant. This allows for better comparability throughout the paper and opens up the possibility to see in-depth how a random forest adapts depending on small changes in the underlying data.

3 Assumptions and Concepts

Before going to the results of the simulation study, the purpose of this section is to lay out the assumptions and concepts on which the theoretical analysis will build upon. Since the consistency proof of the adapted algorithm is done by generalizing Klusowski's (2019) work, the notation closely follows his. Two concepts are laid out, which allow us to understand how big the impact of one variable X_j on the outcome is. Additionally, two Lemmas are introduced which will be useful for proving consistency later on. Before that, the assumptions on the regression function are stated.

3.1 Assumptions

For the subsequent analysis to be valid, the regression function has to fulfill some assumptions. They are taken from Klusowski (2019), and not all assumptions need to hold for each of the following results, but they are presented here together for organizational purposes.

Assumption 1 *For each node* $\mathbf{t}$ *and variable* X_j*, the distribution function* $\mathbb{P}[X_j \leq x_j | \mathbf{X} \in \mathbf{t}]$ *is strictly increasing.*

Assumption 2 *For each node* $\mathbf{t}$*, variable* X_j *and split point* s*, it holds that*

$$\begin{aligned}\mathbb{P}[X_j \leq s \mid a_j(\mathbf{t}) \leq X_j \leq b_j(\mathbf{t})] &\leq \eta \mathbb{P}[X_j \leq s \mid \mathbf{X} \in \mathbf{t}] \text{ and} \\ \mathbb{P}[X_j > s \mid a_j(\mathbf{t}) \leq X_j \leq b_j(\mathbf{t})] &\leq \eta \mathbb{P}[X_j > s \mid \mathbf{X} \in \mathbf{t}]\end{aligned}$$

for some constant $\eta \in (0, 1]$.

Assumption 3 *Assume that there exists a finite integer* $R \geq 1$ *such that* $1 \leq r \leq R$ *and for all strong variables* $j \in \mathcal{S}$,

$$\sup_{x_j \in [0,1]} \inf_{r \geq 1} \{r : \frac{\partial^r}{\partial x_j^r} m(x_j, \mathbf{x}_{\setminus j}) \text{ is nonzero and continuous for all } \mathbf{x}_{\setminus j} \in [0,1]^{d-1}\} \tag{4}$$

is finite.

The first assumption holds as soon as the joint density of the regressors does not vanish. While the second assumption is a bit more restrictive, it is Assumption 3 which is crucial in restricting the form of $m(\mathbf{X})$.
With it we assume that there exists a partial derivative $\frac{\partial^r}{\partial x_j^r} m(x_j, \mathbf{x}_{\setminus j})$ up to a finite order R which is nonzero and continuous for all other input variables. Its primary purpose is to guarantee that the regression function is not "too flat" in the dimensions of our strong variables $j \in \mathcal{S}$. With the assumption holding, we know that each strong variable impacts the outcome variable. This allows to later differentiate between weak and strong variables. Because weak variables do not influence the outcome Y, the slope of the regression function is constant for weak variables as $n \to \infty$. In contrast, because of Assumption 3, it is ensured that strong variables imply that the slope of regression function within their nodes is not constant.

Klusowski (2019) shows that Assumption 3 holds for any linear combination of Gaussian radial functions in $\mathbb{R}^d$ as well as any one-dimensional polynomial or partial sum of a Fourier series. Therefore, the subsequent analysis is valid for a fairly large collection of functions and encompasses more than the result from Biau (2012), who only consider a linear additive model.

3.2 Partial Dependence Function

From the connection between regression trees and kNN estimators, and the results of Lin and Jeon (2006), we know that the importance of regressors is an essential trait. Now, we introduce the conditional partial dependence function to formalize and measure the importance a specific variable X_j has on the output Y. Recall that we assume $Y_i = m(\mathbf{X}_i) + \epsilon_i$. The idea behind the conditional partial dependence function is to look at the influence of one specific X_j while ignoring the others. For this let

$$\bar{F}_j(x_j, \mathbf{t}) = E[Y \mid \mathbf{X} \in \mathbf{t}, X_j = x_j] = \int m(x_j, \mathbf{x}_{\setminus j}) \mathbb{P}_{\mathbf{X}_{\setminus j} \mid \mathbf{X} \in \mathbf{t}}\ (d\mathbf{x}_{\setminus j}) \tag{5}$$

where $\bar{F}_j(x_j, \mathbf{t})$ is the partial dependence function for variable j at point x_j conditional on being in node $\mathbf{t}$. One can also think of (5) as a solution to a least squares approximation of $m(\mathbf{X})$ as a function of only X_j within node $\mathbf{t}$. For later purposes, it is beneficial to additionally define the mean-centered partial dependence function

$$\begin{aligned} \bar{G}_j(x_j, \mathbf{t}) &= \bar{F}_j(x_j, \mathbf{t}) - E[Y \mid \mathbf{X} \in \mathbf{t}] \\ &= E[Y \mid \mathbf{X} \in \mathbf{t}, X_j = x_j] - E[Y \mid \mathbf{X} \in \mathbf{t}] \end{aligned} \tag{6}$$

which is the partial dependence function demeaned by the average of all observations within the respective node.

This concept now allows us to reformulate our definition of strong and weak variables. Weak variables are independent of Y. Then, by the law of iterated expectations, (5) is equal to $E[Y \mid \mathbf{X} \in \mathbf{t}]$ for all weak variables $\{X_j : j \notin \mathcal{S}\}$. This in turn implies that the demeaned partial dependence function $\bar{G}_j(x_j, \mathbf{t})$ is zero for all weak variables. Strong variables, in contrast, affect Y, and thus the demeaned partial dependence function is not necessarily 0. Note, however, that here we are focusing on the population parameters. In a finite sample analysis, it is indeed the case that because of the error terms, we can have $\bar{G}_j(x_j, \mathbf{t}) \neq 0$ even for weak variables. This will play an essential part in the later analysis when constructing a decision tree.

3.3 MDI - Mean Decrease in Impurity

As the partial dependence function gives an account of how much a variable X_j influences the outcome Y, it is also helpful to have a way of comparing the influence of multiple variables. One such measure for ranking variables that has found broad appeal in the practical application of random forests is the mean decrease in impurity (MDI). The intuitive idea behind it is to determine the influence of each variable on the output. In (2), we called the associated decrease in variance $\hat{\Delta}(j, \hat{s}, \mathbf{t})$ which happens when splitting the node $\mathbf{t}$ on variable j with split point s.

The empirical MDI for a variable j is defined as the average of all decreases in variance when splitting on that variable within a tree T, such that

$$\widehat{MDI}(X_j, T) = \sum_{\substack{\mathbf{t}' \supset \mathbf{t} \\ j = j_{\mathbf{t}'}}} \frac{n_{\mathbf{t}'}}{n} \hat{\Delta}(j, \hat{s}, \mathbf{t}') \tag{7}$$

where $\mathbf{t}$ is a terminal node and $\mathbf{t}'$ are all ancestor nodes of $\mathbf{t}$. The associated decrease in impurity when split on j is weighted by the number of observations in the respective node relative to the total number of observations. To generalize the MDI, one can take the average of (7) over all trees and arrive at

$$\widehat{MDI}(X_j) = \frac{1}{n\_tree} \sum_{T} \widehat{MDI}(X_j, T).$$

Looking at the infinite sample version, we can re-write (7) as

$$MDI(X_j, \mathbf{t}) = \sum_{\substack{\mathbf{t}' \supset \mathbf{t} \\ j = j_{\mathbf{t}'}}} w(j, s^*, \mathbf{t}') \hat{\Delta}(j, s^*, \mathbf{t}'). \tag{8}$$

The sum is taken over all splits in which variable X_j was selected as splitting dimension and $w(j, s^*, \mathbf{t}')$ are non-negative weights.
Analyzing the infinite sample properties of the weights, one can express the weights as a function of the demeaned partial dependence function and the decrease in impurity

$$w(j, s^*, \mathbf{t}') = \frac{1}{|\bar{G}_j(s^*, \mathbf{t}')|^2 + \Delta(j, s^*, \mathbf{t}')} \tag{9}$$

where the empirical parameters $\hat{\Delta}$ and $\hat{s}$ are replaced by their population counterparts. The derivation of the weights is a consequence of the proof of Theorem 1 for the CART algorithm. Note that the weights are only clearly defined for strong variables for which $\bar{G}_j(s^*, \mathbf{t}') \neq 0$ and $\Delta(j, s^*, \mathbf{t}') \neq 0$.

3.4 Role of the MDI

Li et al. (2019) highlight the importance of the MDI for the CART algorithm in extremely randomized forests. The CART algorithm chooses the split point such that the resulting variances of Y within the child nodes are maximized. Weak variables, however, are independent of the outcome variable Y,

and therefore splitting does not lead to any decrease in impurity. Thus, Li et al. (2019) show that the MDI for weak variables is zero in the ideal case and the estimator never splits on weak variables. The weights (9) are not defined for weak variables in an asymptotic setting. In contrary, considering a finite sample framework, it can very well be that both terms are different from zero, and thus the MDI can be larger than zero for weak variables. Because of these finite sample properties, the algorithm may find the largest decrease in impurity on a noisy variable. Then, the algorithm would choose to split along a noisy dimension.
While Li et al. (2019) connect the MDI to the splitting on weak variables, Klusowski (2019) goes down the opposite route: relating the MDI of strong variables to the statistical properties of the estimator. He characterizes the bias as a function of the MDI of the strong variables. With that, one is then able to prove the consistency of the random forest for the most general class of functions yet. To get a clearer understanding, the two main theorems on the way to consistency are repeated here.

The first result sheds light on the node-length of a variable X_j in node $\mathbf{t}$ as a function of the MDI. Under some relatively mild assumptions on the predictor variables, it can be shown that the node length in direction j is negatively proportional to the MDI of that variable.

Theorem 1 *Assume* $[a_j(\mathbf{t}),\ b_j(\mathbf{t})]$ *is subnode in the* j^{th} *direction for the terminal node* $\mathbf{t} = \Pi_{j=1}^{d}[a_j(\mathbf{t}),\ b_j(\mathbf{t})]$ *and that Assumption 2 holds. Then,*

$$\mathbb{P}_{\mathbf{X}}[a_j(\mathbf{t}) \leq X_j \leq b_j(\mathbf{t})] \leq exp\left\{-\frac{\eta}{4}MDI(X_j, \mathbf{t})\right\} \tag{10}$$

with η being a constant and $MDI(X_j, \mathbf{t})$ defined by (8). This result says that the higher the MDI of the variable X_j, the smaller the length of the terminal node in the j^{th} direction. Two takeaways resemble the ones from the findings of Lin and Jeon (2006), who analyze the random forest through the lens of an adaptive nearest neighbor estimator. The first is that the side lengths of important variables (large MDI) are smaller. Hence the regression tree splits more often on strong variables. Second, the regression tree can exploit local changes and thus depends on the particular terminal node $\mathbf{t}$. The side lengths can differ even for the same variable, with regions exhibiting a strong signal being split more often and having smaller side lengths.

The most important implication of Theorem 1 is that it allows us to make the bridge to the conditions for the consistency of partitioning estimators as proposed by Stone (1977). One necessary condition for consistency is that the diameter of every terminal node converges to 0 for every strong variable, such that the estimate is genuinely local. This is the case in our setup if $MDI(X_j, \mathbf{t}) \to \infty\ \forall j \in S$. It would be an interesting extension to analyze whether one could exclude the nodes, in which the slope of the regression function is flat, from this condition and still have consistency. In other words, it could be that the restriction can be relaxed to include only those areas of strong variables which contain some signal.
Including all $j \in S$, one can lower bound $MDI(X_j, \mathbf{t})$ by the selection frequency of X_j and a measure of balancedness. For this, let

$$\lambda_j(\mathbf{t}) = 4P_j(\mathbf{t}_L^*)P_j(\mathbf{t}_R^*) = 1 - |P_j(\mathbf{t}_L^*) - P_j(\mathbf{t}_R^*)|^2$$

be a measure of node balancedness and an indicator for the amount of data in either child node. $P_j(\mathbf{t}_L^*)$ $[P_j(\mathbf{t}_R^*)]$ is the relative amount of observations that would fall into the left [right] child node in an infinite sample setting when splitting node $\mathbf{t}$ on variable j. If we were to split at the median, $\lambda_j(\mathbf{t})$ would be 1 and the further we go to either edge, the lower $\lambda_j(\mathbf{t})$ will be. To globalize this measure, we can take the infimum over all parent nodes and arrive at

$$\Lambda_j = \inf_{\mathbf{t}} \lambda_j(\mathbf{t}) \tag{11}$$

Being able to assign to each variable a measure of global balancedness Λ_j, we are in a position to state our second theorem.

Theorem 2 *Suppose Assumptions 1 and 3 hold, and that $\Lambda_j > 0$ $\forall j \in \mathcal{S}$. Then,*

$$MDI(X_j, \mathbf{t}) \geq \Lambda_j K_j(\mathbf{t}) \tag{12}$$

where K_j is the number of times variable X_j was selected as splitting dimension across all parent nodes of the terminal node $\mathbf{t}$.

Theorem 2 together with Theorem 1 imply that the diameter of each terminal node converges to zero in the j^{th} direction as $K_j(\mathbf{t}) \to \infty$ for all strong variables. Drawing on the insight of Lin and Jeon (2006), we know that more important variables are chosen more often as splitting dimensions. As $n \to \infty$, the tree size and with it the selection frequency of those strong variables increases. These two theorems then allow to show that the regression tree and the random forest are consistent estimators under the given assumptions.
In particular, Assumption 3 ensures that the slope of the regression function is not too flat. As a consequence, for strong variables, it holds that $\Delta(j, s^*, \mathbf{t}) > 0$, and thus we can differentiate them from weak variables. Since weak variables do not influence Y, splitting on them does not reduce the variance within the child nodes. This means that in an infinite sample, $\Delta(j, s^*, \mathbf{t}) = 0$ for all $j \notin \mathcal{S}$ which can be seen easily from (1).

Taken together, past research was able to connect measures of variable importance to the statistical properties of a regression tree. The most critical piece, which also serves as a building block for this paper, is the upper bound of the node diameter as a function of selection frequency K_j and node balancedness Λ_j. Following Stone (1977), the node diameter is directly related to the bias and thus links those concepts.

4 From Asymptotics to Finite Samples

A regression tree is a partitioning estimator that makes predictions by dividing the initial sample into smaller, more homogeneous subgroups. The CART algorithm, by default, keeps splitting every node until there is only one observation left, which is then used as a predictor. At the top of a regression tree, when there are still lots of data points within a node, the infinite sample framework accurately describes the inner workings of the CART algorithm. As the number of observations within a node decreases, the infinite sample setup becomes more and more inadequate to describe the procedure with which the estimator is developed. The gradual switch from infinite to finite-sample framework influences the choice of split points and dimensions and, consequently, the tree's structure and predictions. The deeper the tree is grown, the more we leave the realm where asymptotic analysis is permissible, and the more we need to pay attention to the finite sample properties.
This section first explores the effect of error terms on the location of split points in a one-dimensional setting. Extending the argument to a multi-variable framework, we will see how, in addition to the split points, also the choice of splitting dimension is influenced by the switch from infinite to finite sample behavior.

4.1 One dimension

To focus on this switch, we abstract for now from the multi-variable setting and consider the case of repeatedly splitting on a generic strong variable X_j. Fixing j allows us to focus solely on determining the split point s for a node $\mathbf{t}$. The subscript j is dropped in this sub-section in favor of readability.
As a starting point, we consider the case of a node $\mathbf{t}$ which has not been split yet and contains a large number of observations $n_{\mathbf{t}}$. The split point $\hat{s}$ is determined by maximizing the decrease in impurity associated with it and is given by (2). We can rewrite this as minimizing the sum of the resulting child nodes

$$\hat{s} = \arg\min_{s} \; \hat{P}(\mathbf{t}_L)\hat{\Delta}(\mathbf{t}_L) + \hat{P}(\mathbf{t}_R)\hat{\Delta}(\mathbf{t}_R)$$

where $\mathbf{t}_L$ ($\mathbf{t}_R$) is a node containing all observations which are smaller (larger) than the split point $\hat{s}$. Using the definition of $\hat{P}$ from (2), we can reformulate this as

$$\begin{aligned}
\hat{s} &= \arg\min_{s} \; \frac{n_{\mathbf{t}_L}}{n_{\mathbf{t}}}\hat{\Delta}(\mathbf{t}_L) + \frac{n_{\mathbf{t}_R}}{n_{\mathbf{t}}}\hat{\Delta}(\mathbf{t}_R) \\
&= \arg\min_{s} \; \frac{n_{\mathbf{t}_L}}{n_{\mathbf{t}}}\frac{1}{n_{\mathbf{t}_L}}\sum_{\mathbf{X}_i \in \mathbf{t}_L}(Y_i - \bar{Y}_{\mathbf{t}_L})^2 + \frac{n_{\mathbf{t}_R}}{n_{\mathbf{t}}}\frac{1}{n_{\mathbf{t}_R}}\sum_{\mathbf{X}_i \in \mathbf{t}_R}(Y_i - \bar{Y}_{\mathbf{t}_R})^2 \\
&= \arg\min_{s} \; \frac{1}{n_{\mathbf{t}}}\left[\sum_{X \le s}(Y_i - \bar{Y}_{\mathbf{t}_L})^2 + \sum_{X > s}(Y_i - \bar{Y}_{\mathbf{t}_R})^2\right]
\end{aligned}$$

with $\bar{Y}_{\mathbf{t}_L}$ being the average of all observation in node $\mathbf{t}_L$. We can drop the fraction at the beginning since it does not change the solution of minimization problem. Substituting in the model $Y = m(\mathbf{X}) + \epsilon$ where $\epsilon \sim N(0, \sigma_\epsilon^2)$ then yields

$$\hat{s} = \arg\min_s \sum_{X \leq s} \left[\left(m(\mathbf{X}_i) - \overline{m(\mathbf{X}_{\mathbf{t}_L})} \right)^2 + (\epsilon_i - \bar{\epsilon}_{\mathbf{t}_L})^2 + \left(m(\mathbf{X}_i) - \overline{m(\mathbf{X}_{\mathbf{t}_L})} \right) (\epsilon_i - \bar{\epsilon}_{\mathbf{t}_L}) \right] + \sum_{X > s} \left[\left(m(\mathbf{X}_i) - \overline{m(\mathbf{X}_{\mathbf{t}_R})} \right)^2 + (\epsilon_i - \bar{\epsilon}_{\mathbf{t}_R})^2 + \left(m(\mathbf{X}_i) - \overline{m(\mathbf{X}_{\mathbf{t}_R})} \right) (\epsilon_i - \bar{\epsilon}_{\mathbf{t}_R}) \right] \quad (13)$$

The bars represent the average of the terms within the respective child nodes. For a start, consider the case with a large number of observations $n_\mathbf{t}$ in the parent node $\mathbf{t}$ which we aim to split. With $n_{\mathbf{t}_L}$ and $n_{\mathbf{t}_R}$ in both child nodes being sufficiently large, we can replace (13) with its infinite sample counterparts. Using $\epsilon \sim N(0, \sigma_\epsilon^2)$ and the fact that the error term is uncorrelated with the regression function $m(\mathbf{X})$, we get

$$\begin{aligned} s^* &= \arg\min_s \sum_{X \leq s} \left[\left(m(\mathbf{X}_i) - \overline{m(\mathbf{X}_{\mathbf{t}_L})} \right)^2 + \sigma_\epsilon^2 \right] + \sum_{X > s} \left[\left(m(\mathbf{X}_i) - \overline{m(\mathbf{X}_{\mathbf{t}_R})} \right)^2 + \sigma_\epsilon^2 \right] \\ &= \arg\min_s \sum_{X \leq s} \left[\left(m(\mathbf{X}_i) - \overline{m(\mathbf{X}_{\mathbf{t}_L})} \right)^2 \right] + \sum_{X > s} \left[\left(m(\mathbf{X}_i) - \overline{m(\mathbf{X}_{\mathbf{t}_R})} \right)^2 \right] \end{aligned} \quad (14)$$

This equation is the infinite sample version of (13) where s^* denotes the optimal split point in the infinite sample. Since both sides contain a large number of observations, the error term variance is equal to σ_ϵ^2 in both child nodes and therefore is also irrelevant for the minimization problem which selects the split point. Thus, the regression function is the only object determining the split point. This setting allows the regression tree to partition the data according to the underlying signal given by the underlying regression function. Because of the large number of observations in both child nodes, the errors average themselves out and do not play any role.

Yet, with repeated splits the picture changes. The more often we split, the smaller the length of node $\mathbf{t}$ in direction j becomes and the node diameter shrinks. Repeated splits make the groups within a node more homogeneous, meaning that the slope of the regression function within that node becomes flatter and flatter. Because of the continuity of $m(\mathbf{X})$, there comes a point after which partitioning the data into smaller subsets makes the regression function within that node close to a constant. Thus, the variance of the regression function decreases with every split. Looking at (13), this means that the contribution of the regression function to the total decrease in impurity $\hat{\Delta}(\mathbf{t})$ is lower if the node has been split upon multiple times.
On the other hand, doing several splits also decreases the total number of observations within a node. Because of this, one cannot apply the infinite sample properties after a certain point. In the infinite sample setting, the error term variance is the same within both child nodes and cancels itself out. With fewer observations, the (by now small sample) variances are more likely to differ between the child nodes due to randomness induced by the error terms, which then influences where the split points are set. For the most extreme case, consider a node where the slope of the regression function

is entirely flat. Then, the only factor determining the split point in (13) is the error term variances between the child nodes. A low number of observations then makes it more likely that the variances differ because the convergence to the population parameter σ_ϵ^2 is not guaranteed anymore.
Thus, there are two developments taking place simultaneously while a regression tree grows. In the beginning, the within-node variances are influenced overwhelmingly by the signal because the error term variances cancel each other out due to the infinite sample properties as seen in (14). Continued splits decrease the node size and, with it, the strength of the signal. Because the slope of the regression function becomes flatter, its contribution to the total node variance decreases. At the same time, the falling number of observations leads to a breakdown of the asymptotics, and the error term variances start affecting the location of the split points. These processes do not happen necessarily with the same speed and depend heavily on the actual data being analyzed, but the direction of the effects holds for any regression tree.

If we start splitting on a strong variable, the signal is the driving force determining the split points. Since the strength of the signal can vary over the regression surface, in some regions the signal likely is exploited quicker than in others. Regions, where the slope of the regression function is steeper, need to be split more often to make the regression function within the nodes flatter. This way, the regression tree adapts locally to changes in the underlying regression function. As soon as the error terms govern the split points, the estimator does not group similar observations anymore. Instead, it randomly assigns them depending on the respective errors. With the original CART algorithm, this is bound to happen after a certain tree depth, once all of the signal is exhausted and observations are left to split. This is not a sudden event but rather a gradual process as the error term variance within the nodes starts to diverge from σ_ϵ^2. In practice, it is impossible to know at which point the signal is exhausted, and the error terms determine the split points.

4.2 Location of split points

Another interesting point worth highlighting is the connection between the decrease in impurity and the location of the split points. In the case of the CART algorithm, consider the following lemma:

Lemma 1 *Suppose Assumption 1 holds and that* $\Delta(s^*, \mathbf{t}) > 0$. *Then,*

$$\mathbb{P}[X \leq s^* \mid \mathbf{X} \in \mathbf{t}] = \frac{1}{2}\left(1 \pm \sqrt{\frac{\bar{G}(s^*, \mathbf{t})^2}{\bar{G}(s^*, \mathbf{t})^2 + \Delta(s^*, \mathbf{t})}}\right) \tag{15}$$

and by consequence

$$\lambda(\mathbf{t}) = \frac{\Delta(s^*, \mathbf{t})}{\bar{G}(s^*, \mathbf{t})^2 + \Delta(s^*, \mathbf{t})} \tag{16}$$

Since $\mathbb{P}[X \leq s^* \mid \mathbf{X} \in \mathbf{t}] = P(\mathbf{t}_L)$, equation (15) links the split point with the decrease in impurity $\Delta(s^*, \mathbf{t})$. It says that the deviation of the split point s^* from the median of the node is proportional to the decrease in impurity associated with that split point. We can not conclude whether it will be below or above the median, thus $\pm$ indicates that it could be any of the two. A more intuitive way of thinking about this link is through the lens of (16). It says that the node balancedness in node $\mathbf{t}$ depends positively on $\Delta(s^*, \mathbf{t})$. In the case of a very high decrease in impurity, $\lambda(\mathbf{t})$ will be close to 1, and the node is split very close to its median. Put differently, splits at the median suggest a high decrease in impurity.

4.3 End-Cut-Preference

In addition, (16) can be used to make the connection to another phenomenon called end-cut-preference (ECP). Discussed as early as Breiman et al. (1984), ECP describes the tendency of the CART algorithm to split on the edges of a node, if the splitting dimensions are noisy.
The CART algorithm looks for two subgroups with a low variance within the two child nodes while maximizing the difference in variance between them. In the event of infinite observations on both sides, a split along a noisy variable does not decrease impurity. For a noisy variable, the regression function $m(\mathbf{X})$ is simply flat, meaning it does not have any influence on the outcome. In this case, the variance of the error terms is the sole factor determining the split point. But, since there are enough observations in both child nodes, it is guaranteed that the variance in both child nodes is equal to σ_ϵ^2. In this case, the decrease in impurity is zero.
On the other hand, the fewer observations are left in a node, the likelier it is that the error term variance within the respective node is different from its population parameter σ_ϵ^2. Choosing a split point very close to one of the node edges creates two child nodes with significant differences in sample size. While one of the resulting nodes retains most of the observations and, therefore, also the large sample properties of the ancestor node, it becomes more probable that the variance in the smaller child node is different from its infinite sample equivalent. A regression tree tends to split closer to the edges if there is no signal to exploit and the biggest part of the node variance comes from the error terms. Theorem 11.1 in Breiman et al. (1984) formally describes this mechanism.
Looking at (16) confirms this behavior. A weak variable has a low decrease in impurity, making $\lambda(\mathbf{t})$ very low. This low node balance means that the split point is close to one of the node's edges.

The exciting aspect of the end-cut-preference is that it can indicate when the tree starts splitting on noise instead of signal. We argued before that it is impossible to know when the error terms determine the split points. However, using the ECP, we can at least get some ideas of when it is likely to happen. Results from the simulation study described earlier offer some insights into this mechanism. By raising the error term variance, we can automatically increase the contribution of the error terms to the total within-node variance. All else equal, this should increase the likelihood of splitting on noise, thus leading to lower node balancedness.
An illustration of the effects of such an increase can be seen in Figure 2. In the three plots, we use the same data generating process for the regression function. The only difference is the variance of the error terms. Each line plots the empirical cumulative distribution function of the node balances given the tree depth. The very first level of the tree is called *Depth* = 1 and the subsequent levels are numbered accordingly. With a comparatively low $\sigma_\epsilon^2 = 1$, all four levels of the tree have rela-

tively many balanced nodes. Especially the first level has a very few unbalanced nodes (low $\lambda(\mathbf{t})$). As we increase the error term variance to 3 and then to 8, two effects can be observed. First, even in the first level, the amount of unbalanced nodes increases, as evidenced by an earlier increase of the CDF. Second, the amount of very balanced nodes decreases rapidly for levels 3 and 4. In these levels, the previous splits already exploited some of the signal. With a low error term variance, the remaining signal was strong enough to overcome the influence of the error term variance. Meanwhile, with a larger error term variance, levels 3 and 4 cannot distinct the remaining signal from the noise anymore. Consequently, the location of the split points is determined to a more considerable extent by the error terms, and the splits happen more at the edges, leading to more imbalanced nodes.

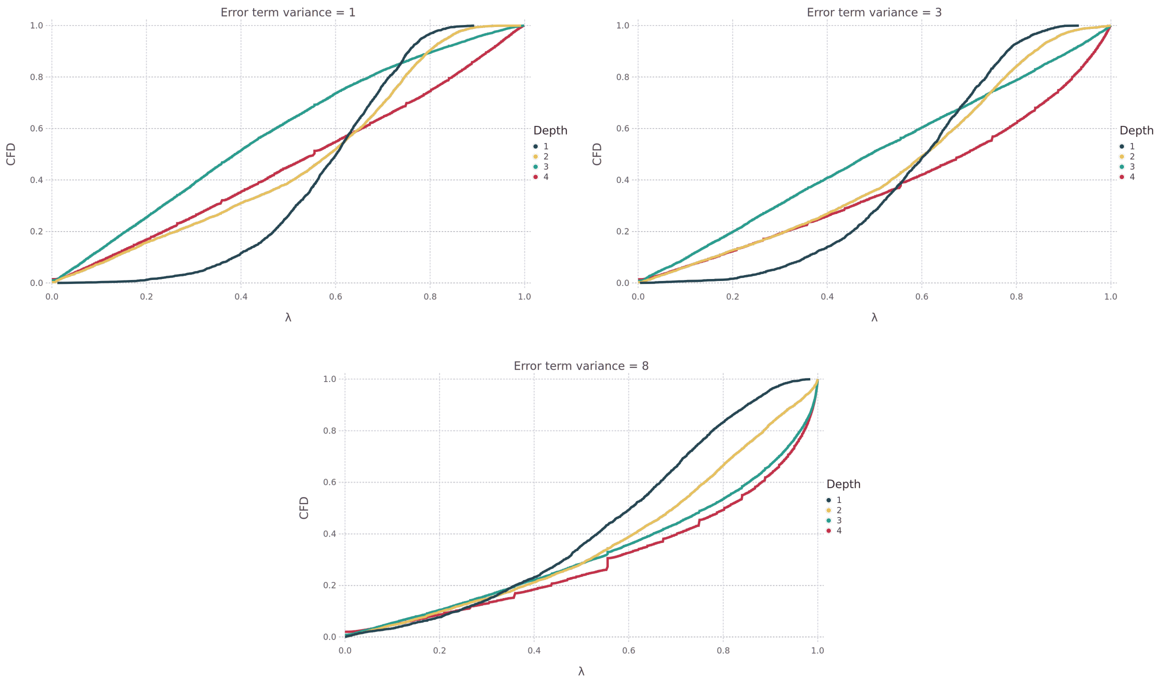

Figure 2: Effect of error term variance on $\lambda(\mathbf{t})$

4.4 Multiple dimensions

Having gained a clearer understanding of what happens to a repeatedly split node along direction j, we now turn to the more common case of considering multiple variables as splitting dimensions. Up to here, the only possibility for a regression tree to be influenced by noise was that its split points were affected, after continued splits sufficiently decreased the signal of a strong variable. Allowing for multiple variables also introduces a new way of noise affecting the regression tree's structure and, consequently, its estimates. The CART algorithm always chooses the dimension and split point with the highest decrease in impurity. The argument for which splitting dimension will be chosen is similar to the one before regarding the influence of the regression function and error terms on the split point.

Considering a split on weak variables, the node variance is equal to the error term variance. Due to the large sample size at the beginning, the error term variance is close to constant for every possible split point and $\hat{\Delta}(\hat{s}, j, \mathbf{t}) \approx 0$ for all weak variables $j \notin \mathcal{S}$. Thus, the choice of possible split dimensions is primarily among the strong variables. Since weak variables do not contain any signal, it is unlikely that any split point chosen in a weak dimension will yield a higher decrease in impurity than choosing a strong variable. As tree depth is increasing, however, we again have two effects working in the same direction. Continued splits on strong variables lead to a weakening of signal in the respective nodes. At the same time, the decrease in observations within each node makes it more probable that a split on a weak variable could cause a sufficiently high decrease in impurity for it to be chosen. The deeper the tree is grown, the more likely it is to choose a noisy variable as a splitting dimension.

Figure 3 illustrates this effect. Compared to the previous simulation, five noisy variables are added to our regressor matrix $\mathbf{X}$ additionally to the five strong variables. The Figure plots the fraction of splits that are made on noisy variables for a given tree depth. We run 100 simulations for each $\sigma_\epsilon^2 \in \{1, 3, 8\}$. The solid line indicates the mean fraction of splits along noisy dimensions for each σ_ϵ^2, and the shaded areas represent the minimum and maximum values.

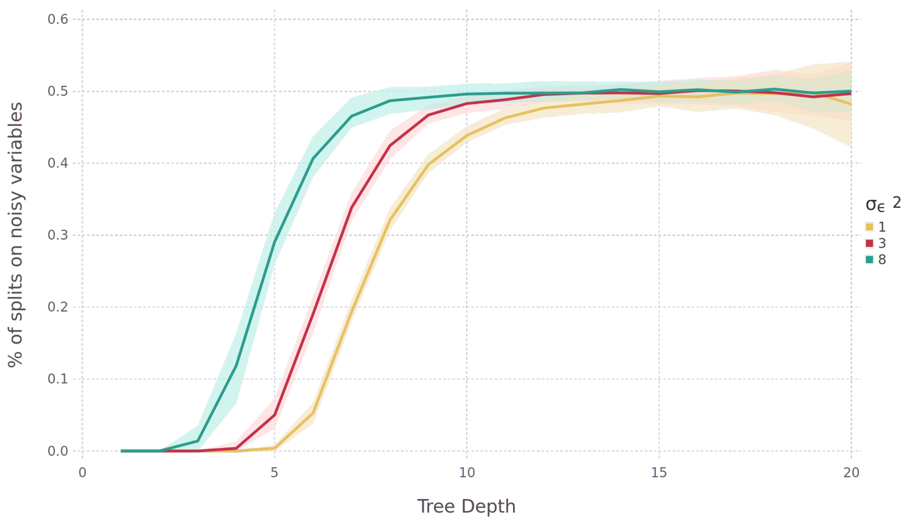

Figure 3: Impact of error term variance

The first effect for all three choices of the error term variance is that half of all splits are done along a noisy dimension after a certain tree depth. As tree depth increases, the nodes become smaller, and most of the variance within the nodes comes from the error terms. This means that after exploiting the signal within a node, everything left for the regression tree to split upon are the error term variances. At this point, the initially strong variables are also ”weak” because they do not contain any signal within their respective nodes anymore. Thus, their relative frequency gives the probability of splitting along an originally noisy or strong dimension. Since we have five strong variables in our simulation and now added five weak ones, after a certain tree depth, the ratio between noisy and weak variables is one half.

The second effect is that this convergence towards the final ratio happens more quickly if the error term variance is higher. A regression tree splits along a weak variable by cutting off the observations close to the edge because the resulting child node variance is different from the infinite sample error term variance in the larger node. By increasing σ_ϵ^2, the variance in the small child node deviates more from the variance in the large child node. This deviation is increasing in σ_ϵ^2, leading to a higher $\hat{\Delta}(\hat{s}, j, \mathbf{t})$ compared to a low error term variance. Choosing a noisy variable as splitting dimension therefore happens earlier when the error term variance is large. In other words, a higher σ_ϵ^2 relative to signal makes it harder for the algorithm to differentiate between strong and weak dimensions.

4.5 Splitting on noisy information

Having seen the impact of error term variances on the tree building process, the natural question is to see how splitting on noise affects the statistical properties of the regression tree and random forest estimator. Here, the literature begins to consider many alternatives to mitigate problems caused by deep trees and splitting on noise.
The very first effect of deeply grown trees is their tendency to overfit the data. While this gives a very good in-sample fit, the estimator fares relatively worse in an out-of-sample setting. This tendency towards low bias and high variance led to the introduction of random forests by Breiman (2001). By aggregating trees built on a subsample of the whole data, the estimator's variance can be reduced without incurring large increases in bias. Other ways to lower the variance were sought by Cutler and Zhao (2001), who propose perfectly randomized trees. Their approach is to choose the splitting dimension and split location at each node randomly. This way, they lower the estimator's dependence on the training data and achieve significant variance reductions.
A problem occurring in both regression trees and random forests is the impact of the end-cut-preference on consistency. One necessary condition for consistency of partitioning estimators is that the number of observations within each terminal node approaches infinity. With ECP, this can not be guaranteed. By splitting at the very end of the considered splitting dimension, the resulting child nodes are very unbalanced: a tiny amount of observations in one node and the large rest in the other. Thus, even if $n \to \infty$, the smaller node consists of very few observations, and consistency cannot be shown. Researchers have made several attempts to mitigate this problem. The most straightforward approach is the one adopted in Athey and Wager (2018). They impose a fraction α which constitutes a bound on the number of observations in each node. Considering a node with $n_\mathbf{t}$ observations, Athey and Wager lower bound the number of observations in the child nodes after splitting by $\alpha n_\mathbf{t}$. The resulting trees are then called α-regular and as n grows, so does the number of observations within each node. Scornet (2015) imposes that at least v observations have to be in each node, which prevents splits at the very extremes of the feature space and is crucial in his consistency proof. Denil, Matheson and De Freitas (2014) employ a slightly different procedure. For a given dimension j, they randomly draw five observations and then only consider those points between the smallest and the largest of the five observations as possible split points. By constraining the range of possible split points, the authors keep the CART algorithm from splitting too close to the edge of a node.

Additionally, another problem for consistency is the use of the same observations when building the tree and making predictions. According to the original algorithm, the same data is used for determining split dimensions and split points, and crucially, also predictors in the terminal node. Error terms impact both the structure and the estimates, thereby introducing a bias. To circumvent this behavior, Athey and Wager (2018) propose "honest trees." They use a second dataset such that observations used for constructing the tree are not used to estimate the terminal nodes. This way, splitting on noise does not have any impact since the error terms which determine the partitions are independent of the errors which form the estimate. It is possible to get consistent estimates by the independence of the errors and letting n approach infinity.
Interestingly, in the appendix to their paper, Athey and Wager (2018) find that the bias of a tree grown with the original CART algorithm is *increasing* in the number of observations. In light of the above discussion, this seems plausible. The original CART algorithm splits until there is only one observation left in each terminal node. Increasing the sample size leads, everything else equal, to deeper trees and thus an increased likelihood of splitting on noise. Without simultaneously raising the number of regression trees in the random forest, the increase in observations does not lead to better predictions. More trees offer more opportunities for the error terms to cancel each other out and thereby mitigating their impact on the final predictions. In a similar vein, Wager and Walther (2015) find that this behavior can be averted when stopping tree growth at a certain lower bound of observations per node v which grows with n. They effectively keep the regression tree from becoming too deep, making it less likely to be impacted by noise.

These settings have in common that they implicitly aim to reduce the likelihood of the tree structure to depend on errors. While assuming "honesty" has favorable theoretical properties, it requires double the data to achieve results similar to the original algorithm. On the other hand, having a lower bound on the observations within every terminal node does not allow the tree to adapt locally to changes in the underlying regression function and is thus a reasonably crude measure.
Based on the observations above, it seems beneficial to grow a tree as long as the signal is strong and discourage splits on weak variables. Once the strength of the signal declines, one should stop splitting. The challenges with this are that the optimal stopping point is not known in practice. Moreover, it varies between nodes depending on the regression function. The following sections will propose a weighted splitting scheme that addresses these problems.

5 Weighted Splitting

Breiman et al. (1984) propose a weighted splitting rule to mitigate the end-cut-preference of the standard CART algorithm. This idea can be used to prevent end-cut-preference and proves promising to remedy the problem of splitting on noise. Following the previous sections, one faces two problems: choosing a weak variable as splitting dimension and the split points being influenced by noise once the signal is weak. One can mitigate both by adopting the splitting rule such that it takes the node balancedness into account. First, we introduce the general idea of weighted splitting for a regression tree. Then we briefly compare it to existing approaches and work out the differences. Finally, we show that the weighted splitting approach is consistent by generalizing the current CART framework to account for the addition of weights.

5.1 Setup

To have a clear picture in mind, let's consider the case where the algorithm is confronted with choosing between a weak and strong variable as a splitting dimension. Because of the end-cut-property of regression trees, splits on weak variables will be made at the node's edges. To prevent the algorithm from doing so, we modify the splitting rule to take the location of the split points into account. Already Breiman et al. (1984) mention the idea of adapting the splitting rule of the CART algorithm and Klusowski (2019) then proposes using weights to increase the node balancedness. We adopt this approach and use weights to penalize splits towards the edges of the feature space. Instead of maximizing the standard decrease in impurity, one could choose

$$\begin{aligned}\Delta_\alpha(s,\mathbf{t}) &= [4P(\mathbf{t}_L)P(\mathbf{t}_R)]^\alpha \ \Delta(s,\mathbf{t}) \\ &= [\lambda(\mathbf{t})]^\alpha \ \Delta(s,\mathbf{t}) \\ &= \omega \ \Delta(s,\mathbf{t})\end{aligned} \tag{17}$$

where $\Delta(s, \mathbf{t})$ is the original decrease in impurity, which is maximized in the standard CART algorithm. $P(\mathbf{t}_L)$ and $P(\mathbf{t}_R)$ are the probabilities that an observation ends up in the left and right node and thus $\omega \leq 1$. These quantities naturally depend on the location of the split point. If a split is at the median of the observations, the probabilities are equal and the weight is maximized at $\omega = 1$. The more imbalanced the nodes become, the smaller the weight will be. The parameter α acts as a regularizer which governs how severely imbalances and thus split points at the edges are penalized. Setting $\alpha = 0$ leads to the unweighted case, which is the original CART algorithm. The higher α, the larger the penalty for splitting at the edges.
Interestingly, (17) reveals a trade-off between the decrease in impurity and the node balancedness for strong variables. Variables with a strong signal have a high decrease in impurity $\Delta(s, \mathbf{t})$. A split point at the edge can overcome the penalty term if the associated reduction in variance is sufficiently large. This way, non-linearities in the regression function are taken into account while overall node balancedness is still rewarded. For small decreases in impurity, on the other hand, the weighted splitting favors splits more towards the median of the node. Noisy variables generally have a very low decrease in impurity due only to spurious correlation, but because of the ECP, the algorithm would like to split on the edges. Exactly this is discouraged by (17).

The first mechanism through which noise affects the tree structure was choosing a weak variable as a splitting dimension. By favoring splitting points closer to the median, the weighted splitting rule implicitly penalizes weak variables and makes them less likely to be chosen as splitting dimensions. The other impact of noise was through influencing the location of split points. This generally happens in the deeper nodes of the regression tree, where the strength of the signal diminishes, and the decrease in the number of observations leads to error terms determining the location of split points. With a low signal, it is mainly the errors contributing to the decrease in impurity. The associated $\Delta(s, \mathbf{t})$ is then comparably low. As soon as the estimator starts splitting based only on noise, it increases its variance. At this point, it is best to terminate that particular node quickly. Since the CART algorithm keeps splitting until there is only one observation left in each node, the question is how to split quickly to have only one data point left. Repeatedly choosing the median of each resulting child node is the fastest way to terminate a node. Thus, weighted splitting ensures that once there is no signal within a node anymore, that node is terminated in the quickest manner possible.

5.2 Comparison to other approaches

There have been many adaptions of the standard CART algorithm to make it more efficient. The approaches which have been proposed up to now include limiting overall tree depth, imposing a minimum amount of observations in each node, or restricting the interval within a node in which the optimal split point is searched for.
The drawback of those approaches is that they are a binding constraint for the whole regression tree. Taking an overall limit on tree depth as an example, every tree branch is stopped after a pre-specified depth. This is advantageous for nodes that already exploited most of their signal since further splits along them would only lead to an increase in variance. On the other side, one gives up valuable information for nodes that still contain a large amount of signal. Continued splits within those nodes would lead to a more accurate partition and consequently to better predictions. Thus, the constraint imposed by limiting tree depth affects the whole tree irrespective of the differences across nodes regarding signal strength. The optimal limit is always determined by a trade-off between nodes that should be terminated quickly and those on which one could continue splitting. One of the significant advantages of regression trees and random forests over other estimators is their adaptability to local changes in the regression function. Optimally one could terminate a node quickly as soon as the signal is exploited but at the same time keep on splitting a different node where the signal is still strong. Setting one rigid boundary on any parameter - be it tree depth, a minimum amount of observations, or something else - keeps the random forest from employing its local adaptability.
Weighted splitting makes improvements on this front. By setting a weight, we penalize splits on the edges, which indicate strongly that a weak variable was chosen as a splitting dimension. In this way, splits along the middle of the node are rewarded and once all of the signal within a node is exploited, that node is split along the median to terminate it quickly. In another node where the signal is still strong, it can overcome the weight and the choice of split points is determined more by the regression function. With weighted splitting, the optimal weight is, of course, also binding for the whole tree, but it permits the CART algorithm to depend more on the data *within* the respective node compared to the overall data across all nodes. Put differently, we want the *correct* nodes, which contain lots of signal, to be split often while quickly terminating the others.

5.3 Consistency

While weighted splitting offers improvements over the currently used methods to increase the efficiency of regression trees and random forests, its statistical properties also need to be considered. One of the most fundamental aspect of any estimator is its consistency. Because error terms govern the structure of a tree, it is challenging to describe the tree-building process and its effects statistically. We build on the literature of the recent past, which has incrementally come closer to proving consistency for regression trees and random forests. After Biau (2012) shows consistency of random forests for an additive model, Klusowski (2019) extends it to any linear combination of Gaussian radial functions in $\mathbb{R}^d$ as well as any one-dimensional polynomial or partial sum of a Fourier series.
To start, it is worth mentioning that weighted splitting is a generalization of splitting rules. The most common rules up to now are either splitting according to the largest decrease in impurity or always on the median of the node. Equation (17) sets these rules into a broader context. The regularizer of our weighted splitting approach allows us to bridge the original CART algorithm and perfectly random tree ensembles (PERT). The latter is made up of trees that ignore the decrease in impurity and always split on the median. Within PERT trees, the location of the split point does not depend on the underlying data, and thus the tree is data-independent which significantly facilitates its consistency poof (Cutler and Zhao (2001)). On the other side, Klusowski (2019) shows consistency for a regression tree grown with the original CART algorithm for the broadest class of functions yet.

Based on Klusowski's approach, we show how weighted splitting fits into his setup and is reconcilable with consistency in the broadest way, which has been shown to hold for random forests. The theorems build on Scornet (2016), who analyzes non-adaptive trees in an infinite sample setting. To achieve this, the estimates in the terminal nodes are independent of the training set $\mathcal{D}_n$ used to determine the structure of the tree. The maximum amount of terminal nodes of any tree within the forest is given by $n_{\mathbf{t}}$, and the number of observations is n.

Theorem 3 *Let $\hat{Y}(x) = \hat{Y}(x, \mathcal{D}_n)$ be a non-adaptive random forest predictor where the weighted splitting rule (17) is applied and $\mathbf{t}$ is a terminal node. Assume that*

1. *$m(x)$ is continuous on $[0,1]^d$*
2. *$\frac{n}{n_{\mathbf{t}}} \to \infty$*
3. *$\min_{\mathbf{t}} MDI(X_j, \mathbf{t}) \to \infty \quad \forall j \in \mathcal{S}$*

Then, as $n \to \infty$,

$$\mathbb{E}_{\mathcal{D}_n}\left[\int |m(\mathbf{X}) - \hat{Y}(\mathbf{X})|^2 \; \mathbb{P}_{\mathbf{X}} \; d(\mathbf{x})\right] \to 0$$

The first assumption guarantees that the slope of the regression function within each node approaches zero as the node diameter becomes smaller. The second assumption ensures that the number of observations grows quicker than the number of terminal nodes of any tree. Thus, the number of observations within each terminal node approaches infinity. This way, one avoids all the particularities discussed in the previous section with the gradual switch from the infinite sample analysis to the finite sample difficulties. Finally, the last assumption describes the behavior of the MDI and, with it, the node diameters. Theorem 1 relates the MDI and node diameter inversely. The larger the MDI, the smaller the diameter of the respective node. By letting the MDI approach infinity, the node is made arbitrarily small to pick up local changes in the regression surface.

The main task for a consistent estimator is now to show that the MDI converges to infinity for all strong variables. From (2) it holds that $MDI(X_j, \mathbf{t}) \geq \Lambda_j K_j(\mathbf{t})$, where Λ_j is the global balancedness of j and K_j the number of times variable j was chosen as splitting dimension.
Focusing first on K_j, it is essential to note that the error terms do not influence the split points and dimensions since we are in an infinite sample setting, where the error term variances cancel each other out. Therefore, the algorithm never chooses weak variables as splitting dimensions. Abstracting from the errors, we can fall back on the results from Lin and Jeon (2006) that every strong variable has the same importance in the optimum. Thus, after a certain tree depth, all variables lead to the same decrease in impurity. The probability of choosing one of the strong variables as splitting dimension is $1/S \; \forall j \in \mathcal{S}$ where S is the number of strong variables. Scornet et al. (2015) describe this mechanism more formally. With $n \to \infty$, tree depth also increases accordingly and since the probability of selecting a strong variable is $1/S > 0$, the selection frequency $K_j \to \infty$.
Turning to the global balancedness Λ_j, we need to show that it is strictly larger than zero since K_j increases in n. For this, consider the following Lemma:

Lemma 2 *Assume that the conditions from Assumption 3 are met. Then,*

- $\bar{G}_j(x_j, \mathbf{t}) > 0$
- $\Delta(j, s^*, \mathbf{t}) > 0$

hold for all strong variables $j \in \mathcal{S}$.

The global balancedness is the minimum over all nodes $\Lambda_j = \inf_{\mathbf{t}} \lambda_j(\mathbf{t})$ which in turn is the ratio between two parts: $\Delta(s^*, \mathbf{t})$ and $\bar{G}(s^*, \mathbf{t})^2 + \Delta(s^*, \mathbf{t})$. Since both are strictly larger than zero, the ratio may be close to zero but never exactly zero. With this, the MDI for each strong variable converges to infinity and also the last assumption for the consistency of the weighted splitting estimator is guaranteed.
This section's focus is on how to extend the framework of Klusowski (2019) and prove consistency of the weighted splitting rule for a generic weight determined by α. Since the main benefit of the weighted splitting is derived from its performance in a small sample setting, the following section considers the impacts of using weights as trees are grown deeply.

6 Weighted Splitting in Practice

The main goal behind introducing weights into the splitting rule is to mitigate the impact of noise on the tree building process. While in an infinite sample they do not play any role, the error terms start affecting the regression tree's structure when the sample size within a node becomes small. Through the mechanisms described in the previous sections, this impact is increasing in tree depth.
A small regularizer α allows for splits near the edges while a larger one forces the splits points more towards the median of the nodes. Consequently, conditioning the size of the regularizer on tree depth can yield improvements over the original CART algorithm. While at the beginning the signal is strong, we permit with a small α that the algorithm chooses the highest decrease in impurity without focusing too much on the location of the split points. As tree depth is increasing and the error terms start to influence the tree's structure, a higher α forces it to split more along the median and terminate those nodes quickly. The main difference to before is that we do not hold the regularizer α fixed for the whole regression tree, instead we allow it to vary with tree depth.
An empirical comparison of the weighted splitting approach with the original CART algorithm and also other popular tuning parameters showcases the improvements achievable with the weighted splitting rule. Before that, a short description into how exactly these comparisons are made, is given.

6.1 Estimator Description

In the following, we compare the weighted splitting approach to the standard CART algorithm, including amendments like maximum tree depth. Since weighted splitting is introduced newly in this paper, there is no algorithm to fall back on. In addition, most software packages have numerous features and default options, and it is not easy to guarantee that one uses the original CART algorithm. To ensure that the estimator mirrors the setup described here, we implement a random forest estimator that allows for this and which can be accessed via Github[1].

The basis is a random forest consisting of n_tree regression trees, where splits are done according to the weighted splitting rule (17). By setting the default value of $\alpha = 0$, we have the standard CART algorithm. At each node, the optimal split point is chosen among $max\ features$ dimensions, usually $max\ features = d$, meaning we always consider every variable. To de-correlate the trees as suggested by Breiman (2001), every tree within the forest is built on a bootstrapped sample of the whole dataset. Since we also want to showcase the improvements of weighted splitting over currently used methods, certain limits and constraints can be imposed. First, it is possible to limit the depth of the tree by setting *max depth* to the desired value. In addition, one can control the minimum number of observations within a leaf by changing a parameter called $min\ samples\ leaf$.

To make sure that these comparisons are as objective as possible, we always compare approaches with their respective optimal parameters. These parameters are chosen through a three-fold, grid-search cross-validation procedure. For this, a vector of parameter values is supplied, and a training and

[1]The source files for the estimator and reproduction of the results are available under https://github.com/christianhilscher/emma

test set are generated independently of each other. Next, the parameter value which minimizes the means squared error of the training set is chosen. This optimal parameter is then used to predict the outcomes of the test set. A more detailed description can be found in the files on Github.

6.2 Impact of Tree Depth

Since we aim to partition our data based on signal instead of noise, the weights should be larger when noise is higher. Drawing on the previous discussion, weights that increase in the error term variance and tree depth would yield better predictions. Going beyond the simulation study into the real world, it is impossible to know the size of the error term variance. Thus, there is no point in making the weights depend on an unknown quantity. On the contrary, tree depth is observed and, therefore, amenable for the weights to depend on.
For this reason, let k denote tree depth and assume

$$\alpha_k = f(k) \qquad \text{with} \quad \frac{\partial f(k)}{\partial k} > 0 \tag{18}$$

such that the weights are a function of tree depth. Recall that the weighted splitting rule maximizes $[4P(\mathbf{t}_L)P(\mathbf{t}_R)]^{\alpha}\Delta(s, \mathbf{t})$. The first derivative being positive then forces the weights to rise as tree depth increases and unbalanced nodes are penalized more. A straightforward example for such a weighting function would be $\alpha_k = ak$ or $\alpha_k = k^a$. Depending on the particular form of $f(k)$, the weights respond more or less to tree depth. For the rest of the section, we choose $\alpha_k = k^a$ as our preferred specification, although there are no reasons why one couldn't pick any other increasing function. The optimal value for a can then be chosen via cross-validation.

The effect of the weights on general tree depth can be seen in Figure 4. As the number of observations n increases, so does the average tree depth in a forest.

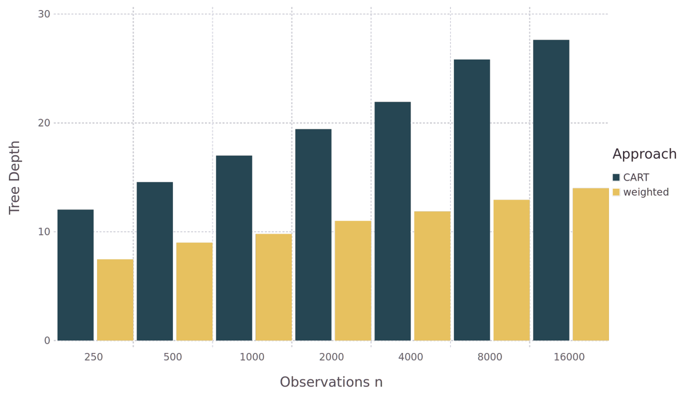

Figure 4: Average Tree Depth

Compared to the unweighted approach, the weighted splitting rule significantly curtails the depth of trees. This is not surprising given that higher weights favor splits along the median of nodes, which

is the quickest way to terminate a tree. Not allowing the regression tree to grow deep prohibits the regression tree from splitting on noise, once the signal is exploited.
The second channel through which noise impacts the predictions is through the algorithm selecting weak variables as splitting dimensions. Figure 5 compares the weighted splitting to the original CART algorithm for three different values of $\sigma_\epsilon^2 = \{1, 3, 8\}$. The yellow line describes the mean fraction of splits along noisy variables using weighted splitting, while the black line shows the results of the CART algorithm, which are identical to Figure 3. The mean is taken over 100 simulation runs, and the shaded areas indicate the minimum and maximum values. As the weighted splitting leads to more shallow trees, all trees in Figure 5 are terminated after a tree depth of 11. In contrast, the unweighted trees are grown significantly deeper. Throughout the first levels of the tree, both approaches can differentiate well between signal and noise and only split on strong variables. As tree depth increases, the strength of the signal declines and the algorithms start selecting weak variables as splitting dimensions. At first, the weighted splitting follows the line of the CART algorithm and also splits sometimes along noisy variables. This is the case while there is still some signal to be exploited, but the error terms already influence the tree building process. As soon as the nodes become sufficiently small and most of the signal is already used, the introduction of the weights starts to make a difference. With no more signal to split upon, the weights become binding and force splits along the median of the nodes. On the one hand, this leads to fewer noisy variables chosen as a splitting dimension because those would like to split at the extremes of the node. At the same time, splits along the median terminate the tree quickly since any further splits would only cause overfitting and an increase in variance.

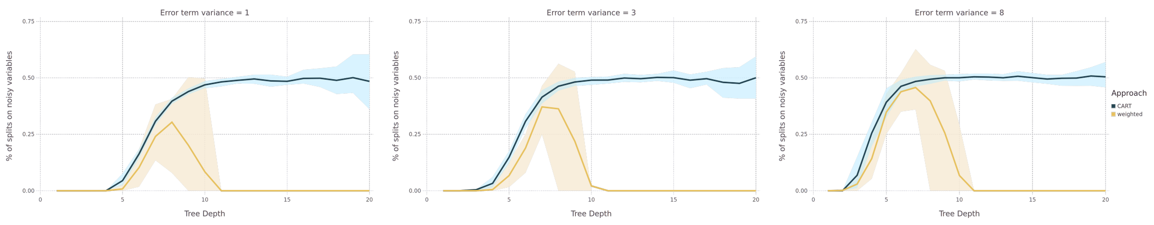

Figure 5: Comparison with $\sigma_\epsilon^2 = \{1, 3, 8\}$

For the CART algorithm, the effects are similar to Figure 3: an increase in the error term variance leads to a quicker convergence to the fraction of split points determined by the relative number of strong variables over weak variables. As the σ_ϵ^2 increases, it becomes more challenging to tell strong and weak variables apart. One manifestation of this is that splits along noisy dimensions happen earlier. Even though weighted splitting is not immune to this, its advantage lies in quickly terminating trees as it becomes impossible to distinguish between signal and noise.

6.3 Empirical Comparison

While weighted splitting has clear advantages over the original CART algorithm, the question of how it stacks up against other popular tuning parameters remains.
In theory, weighted splitting has multiple advantages over other tuning parameters. First, it implicitly

restricts the depth of the trees to avoid splitting on noise and thereby reduces overfitting. One can also achieve this effect with parameters such as *max depth*, which imposes a definite constraint on the tree depth k. Moreover, due to its easy interpretability, it is widely used in applied works. Imposing limits on tree depth, however, does not restrain the end-cut-preference of the CART algorithm.
A more subtle approach is setting a parameter $min\ samples\ leaf \geq v$. By forcing at least v observations to be in each node, it indirectly influences tree depth. This is also the instrument used by Scornet et al. (2015) in a simplified framework to prove consistency. Additionally, the restriction of at least v observations per node also discourages splits at the edges of the nodes. In this regard, both weighted splitting and setting a minimum number of observations achieve the same goal.
Yet, the advantage inherent to weighted splitting, which none of the other approaches share, is its local adaptability. While the alternative restrictions all impose constraints affecting every node in the tree equally, the weights allow for some flexibility depending on the underlying regression function. Setting *max depth* terminates a tree even though some gains could still be made from another split and $min\ samples\ leaf$ rule out any possibility of outliers. In contrast, weighted splitting terminates nodes where it cannot differentiate anymore between signal and noise, but keeps splitting where the signal is still strong. Contrary to setting $min\ samples\ leaf$, weighted splitting only penalizes splits towards the edges, but allows for splits if the associated decrease in impurity is sufficiently large.

For comparing the weighted splitting to the original CART algorithm and it's variations, we run 100 simulations for different sample sizes from 250 to 16,000 observations. The optimal tuning parameter was chosen via a three-fold grid-search procedure, ensuring that we compare the best respective models. Figure 6 shows the result of this comparison, decomposed into bias, variance and the mean squared error (MSE). The first sub-figure depicts the effect of increasing sample size on the bias. As the number of observations n increases, the bias decreases rapidly and eventually goes against zero for all approaches except the original CART algorithm. Using a different regression function, Athey and Wager (2018) also find that the bias of the CART algorithm is increasing in sample size. One possible explanation could be the fact that in both simulations the number of trees is kept constant. However, it is not entirely clear, whether this is a general finding or rather a particularity of the chosen regression functions. As soon as the original CART algorithm is adjusted either via *max depth* or $min\ samples\ leaf$, its behavior is identical to the weighted splitting rule. This corroborates the theoretical findings that weighted splitting is a generalization of the CART algorithm and thus unbiased under the same assumptions. At the same time, there is no apparent difference between the weighted splitting and setting $min\ samples\ leaf$ or imposing an empirically optimal tree depth via *max depth*.

Turning to the second and third sub-figure, the comparison of the variance and MSE of the predictions exposes some big differences. Since the bias goes against zero for all approaches, the variance is the determining part of the MSE. First, it is noteworthy that all tuning parameters improve on the original CART algorithm in terms of the mean squared error. Moreover, as the sample size increases, so does the variance and with it the MSE. Theoretically, due to the consistency arguments in the previous section, one would expect the mean squared error to decrease as the sample size rises. However, these arguments are made under the assumption of an infinite sample framework and a

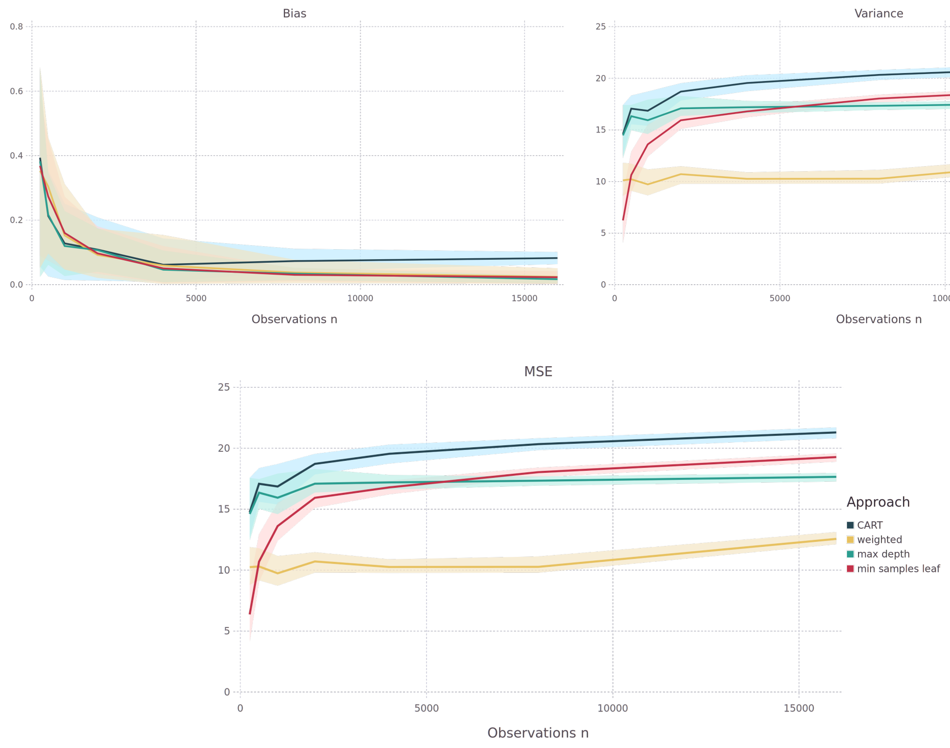

Figure 6: Decomposition of Comparison

non-adaptive regression tree. Since we are more interested in the comparison with the original CART and algorithms widely used in applied work, we choose to focus on those.

Except in the first case of 250 observations, weighted splitting consistently outperforms all alternative methods. While the tuning parameter *min samples leaf* looks promising in small sizes, it quickly loses its performance advantage over *max depth* with rising n. The weighted splitting rule has a significantly lower mean squared error over the whole range of observations than the original CART algorithm and when contrasted against the other two methods. Table 1 shows the reduction in MSE compared to the CART algorithm in percentages.

Obs. n	CART	weighted	max depth	min samples leaf
Panel A: Friedman Function				
250	0.0	-30.5%	-0.9%	-56.8%
500	0.0	-39.7%	-4.3%	-37.4%
1000	0.0	-42.2%	-5.5%	-19.3%
2000	0.0	-42.8%	-8.7%	-14.9%
4000	0.0	-47.5%	-12.0%	-14.1%
8000	0.0	-49.5%	-14.7%	-11.3%
16000	0.0	-41.0%	-17.1%	-9.5%

Table 1: MSE-Comparison to original CART algorithm

All comparisons up to now use the Friedman function (3). To ensure that the results presented here are not a one-off due to this specific function, we also consider three other regression functions. Their performance is again measured in terms of MSE reduction compared to the CART algorithm and given in Table 2 Their form and prior uses are given in more detail in the Appendix.

Obs. n	CART	weighted	max depth	min samples leaf
Panel B: DP-3 Function				
250	0.0	-34.9%	0.5%	-80.6%
500	0.0	-28.9%	-3.4%	-29.6%
1000	0.0	-38.3%	-7.2%	-12.6%
2000	0.0	-36.5%	-8.3%	-8.0%
4000	0.0	-41.7%	-9.5%	-9.5%
8000	0.0	-35.7%	-8.4%	-6.2%
Panel C: DP-8 Function				
250	0.0	-32.2%	-1.0%	-62.8%
500	0.0	-41.2%	-0.6%	-29.6%
1000	0.0	-35.1%	-3.9%	-17.0%
2000	0.0	-32.2%	-10.8%	-10.3%
4000	0.0	-36.7%	-6.1%	-5.3%
8000	0.0	-34.4%	-9.4%	-6.3%
16000	0.0	-37.7%	-9.8%	-3.4%
Panel D: Robot Function				
250	0.0	-48.0%	-12.2%	-84.2%
500	0.0	-40.3%	-9.6%	-46.1%
1000	0.0	-57.9%	-17.6%	-40.0%
2000	0.0	-60.2%	-21.0%	-30.6%
4000	0.0	-58.8%	-25.9%	-25.4%
8000	0.0	-65.2%	-28.1%	-23.4%
16000	0.0	-61.0%	-32.5%	-23.8%

Table 2: MSE-Comparison to original CART algorithm

Similar to the previous table, one pattern emerges: the consistent advantage of weighted splitting. Irrespective of regression function, the weighted splitting rule delivers better predictions than the original algorithm when measured against the other two alternatives. More specifically, the improvements over the CART algorithm are relatively constant and do not depend too much on the number of observations. Only in fairly low sample sizes, the weighted approach does not have the upper hand. One explanation for this could be the very way through which it achieves its superiority in larger samples. Since we let the weights depend positively on tree depth, very low sample sizes could impede the weights to be large enough to be binding. Since most ML algorithms are used in settings with thousands if not more observations, this is not too much of a concern from an applied point of view. As soon as the number of observations rises, the local adaptiveness can play out its strengths and thereby achieves significant improvements over the alternative methods.

These comparisons can be taken as an indicator that the theoretical advantages of the weighted splitting rule also carry over into the empirical realm. Moreover, its predictions are more accurate than the original CART algorithm and beat other popular tuning parameters when measured via the MSE.

6.4 Further Ideas

Introducing the concept of weighted splitting into the CART framework opens up the possibility to exploit the switch in a regression tree from an infinite sample setting to a finite sample framework. As this is the first attempt to generalize the original CART algorithm, it opens up multiple new avenues for further research. While there are multiple theoretical advantages of using weights in the splitting rule, and the respective improvements have been documented, more work is needed to shed some light on the exact source of these gains. Especially the effects of the local adaptability and which nodes get terminated while others keep splitting could be of interest.
Additionally, one could bring the weighted splitting rule even closer to reality and investigate the behavior in a setting where not all variables are chosen as potential splitting dimensions such that $mtry < d$. This widely used restriction aims to further de-correlate the trees as already done via bootstrapping. Here, the weighted splitting rule could prove more advantageous than in the just presented setting where $mtry = d$. Only considering a subset of dimensions makes it more critical to prevent end-cut splits and retain a balanced sample. In case of only weak variables being selected, it is beneficial if the resulting child nodes both have lots of observations such that the tree can recover from a split along a noisy dimension. This is achieved by splitting along the median, which is rewarded with the weighted splitting rule.
In addition, the generalization of the CART algorithm to include weights bridges two, up to now distant, approaches: the original CART algorithm and perfect random tree ensembles (PERT), proposed by Cutler and Zhao (2001). By choosing the weights prohibitively high, all splits within a tree are made along the median of the nodes, effectively mirroring the behavior of a PERT. The connection between the two approaches and especially the theoretical implications could be interesting to uncover. Another way of looking at this would be through the lens of data-dependency. The data-dependency of split points and splitting dimensions is one of the major obstacles for a thorough statistical analysis of regression trees and ensemble methods in general. With weights being sufficiently high and the splits taking place along the median of the nodes, one has an instrument to govern the data-dependency of the split points.
Leaving the realm of theoretical analysis, another obvious extension would be to see how the weighted approach stacks up with real-world data. The simulations presented here had the advantage of a controlled environment. It would be alluring to see how the theoretical advantages carry over into actual problems and how the weighted splitting fares in a setting with real-world data.

7 Conclusion

Regression trees and random forests are usually analyzed in an infinite sample setting. While it facilitates statistical analysis, it is not an accurate description of the process within a tree. With

decreasing observations in nodes, the error terms start impacting the tree-building process and lead to an increase in the estimator's variance. The introduction of weights into the CART splitting rule rewards splits towards the median of the nodes and mitigates the influence of the error terms. Error terms shape the tree structure through a phenomenon known in the literature as end-cut-preference, which the weighted splitting rule penalizes. As the sample size decreases and with it the strength of the signal, the weights automatically terminate nodes where the error terms impact the structure of the regression tree. This leads to less overfitting, and through reducing the variance, the predictions become more accurate. The main advantage of the weighted approach is its local adaptability. It can terminate nodes where all the signal has been exploited while allowing further splits in regions still exhibiting some signal. Thus, it adjusts locally to the underlying regression function and is a more precise instrument when used as a tuning parameter. By generalizing the standard CART framework, we show that the previous consistency results hold under the same assumptions.
The empirical performance of weighted splitting is significantly exceeding that of the original CART algorithm. Compared to limiting tree depth and the minimum observations within a node, two other widely used tuning parameters, the weighted splitting rule delivers superior predictions. While the first results from introducing weights into the CART framework are promising, they also open up new paths for further research, both theoretical and applied.

References

[1] An, J., and Owen, A. Quasi-regression. *Journal of Complexity 17*, 4 (2001), 588–607.

[2] Athey, S., and Wager, S. Estimation and Inference of Heterogeneous Treatment Effects using Random Forests. *Journal of the American Statistical Association 113*, 523 (2018), 1228–1242.

[3] Biau, G. Analysis of a Random Forests Model. *Journal of Machine Learning Research 13*, 38 (2012), 1063–1095.

[4] Breiman, L. Random Forests. *Machine Learning 45*, 1 (2001), 5–32.

[5] Breiman, L. Consistency for a simple model of random forests. Tech. rep., 2004.

[6] Breiman, L., Friedman, J., Stone, C., and Olshen, R. *Classification and Regression Trees.* Taylor & Francis, 1984.

[7] Cutler, A., and Zhao, G. PERT - Perfect Random Tree Ensembles. *Computing Science and Statistics* (2001), 497.

[8] Denil, M., Matheson, D., and de Freitas, N. Narrowing the Gap: Random Forests In Theory and In Practice. In *International Conference on Machine Learning* (2014).

[9] Dette, H., and Pepelyshev, A. Generalized Latin Hypercube Design for Computer Experiments. *Technometrics 52*, 4 (2010), 421–429.

[10] Friedman, J. H. Multivariate Adaptive Regression Splines. *The Annals of Statistics 19*, 1 (1991), 1–67.

[11] Ishwaran, H. The Effect of Splitting on Random Forests. *Machine Learning 99*, 1 (2015), 75–118.

[12] Klusowski, J. Analyzing CART. *arXiv e-prints* (June 2019), arXiv:1906.10086.

[13] Klusowski, J. Sparse Learning with CART. In *Advances in Neural Information Processing Systems* (2020), pp. 11612–11622.

[14] Li, X., Wang, Y., Basu, S., Kumbier, K., and Yu, B. A debiased MDI feature importance measure for random forests. *arXiv preprint arXiv:1906.10845* (2019).

[15] Lin, Y., and Jeon, Y. Random Forests and Adaptive Nearest Neighbors. *Journal of the American Statistical Association 101* (June 2006), 578–590.

[16] Scornet, E. On the Asymptotics of Random Forests. *Journal of Multivariate Analysis* (2016).

[17] Scornet, E., Biau, G., and Vert, J.-P. Consistency of Random Forests. *The Annals of Statistics 43*, 4 (Aug 2015).

[18] Stone, C. J. Consistent Nonparametric Regression. *The Annals of Statistics 5*, 4 (1977), 595 – 620.

[19] Wager, S., and Walther, G. Adaptive Concentration of Regression Trees, with Application to Random Forests, 2016.

Appendix

A Additional Regression Functions

In addition to the Friedman function (3), we use three other functions to compare the performance of the weighted splitting approach.
The first two are taken from Dette and Pepelyshev (2010) and are widely used for computer experiments. Their 3-dimensional function is called **DP-3** in Table 2 and is given by:

$$m(\mathbf{X}) = 4(X_1 - 2 + 8X_2 - 8X_2^2)^2 + (3 - 4X_2)^2 + 16(2X_3 - 1)^2\sqrt{X_3 + 1}$$

and the 8-dimensional counterpart **DP-8** by:

$$m(\mathbf{X}) = 4(X_1 - 2 + 8X_2 - 8X_2^2)^2 + (3 - 4X_2)^2 + 16(2X_3 - 1)^2\sqrt{X_3 + 1} + \sum_{i=4}^{8} i\ ln\left(1 + \sum_{l=3}^{i} X_l\right).$$

All regressors are drawn independently from U$[0, 1]$. These functions are highly curved in some dimensions while being relatively flat in others.

Finally, the function called **Robot** in Table 2 is taken from An and Owen (2001) and is a mathematical representation of the position of a robot arm with four segments. Over time, it became commonly used in papers evaluating neural networks. The origin of the arm is fixed, and the segments have lengths X_1, X_2, X_3, and X_4 respectively. Call the angles at which the segments are positioned θ_i for $i = 1, 2, 3, 4$. The line segments are independently drawn from U[0, 1] while the angles are drawn from U$[0, 2\pi]$. The response $m(\mathbf{X})$ is then the distance from the end of the fourth segment to its origin.

$$\begin{aligned} m(\mathbf{X}) &= \left(u^2 + v^2\right)^{0.5} \\ u &= \sum_{i=1}^{4} X_i\ sin\left(\sum_{l=1}^{i} \theta_l\right) \\ v &= \sum_{i=1}^{4} X_i\ cos\left(\sum_{l=1}^{i} \theta_l\right) \end{aligned}$$

B Proofs

Proof of Lemma 1

Here we present a generalized framework and allow for weighted splitting. The terms which depend on the weighted splitting rule are generally denoted with a subscript α.
One can re-write the decrease in impurity as

$$\Delta(s,\mathbf{t}) = P(\mathbf{t}_L)P(\mathbf{t}_R)\left[\mathbb{E}[Y|\mathbf{X}\in\mathbf{t}, X\le s] - \mathbb{E}[Y|\mathbf{X}\in\mathbf{t}, X > s]\right]^2$$

and

$$\Xi(s,\mathbf{t}) = P(\mathbf{t}_L)P(\mathbf{t}_R)\left[\mathbb{E}[Y|\mathbf{X}\in\mathbf{t}, X\le s] - \mathbb{E}[Y|\mathbf{X}\in\mathbf{t}, X > s]\right]$$

such that

$$\Delta(s,\mathbf{t}) = \frac{[\Xi(s,\mathbf{t})]^2}{P(\mathbf{t}_L)P(\mathbf{t}_R)}.$$

We consider the weighted splitting rule which is going to be maximized:

$$\Delta_\alpha(s,\mathbf{t}) = [4P(\mathbf{t}_L)P(\mathbf{t}_R)]^\alpha \frac{[\Xi(s,\mathbf{t})]^2}{P(\mathbf{t}_L)P(\mathbf{t}_R)} \tag{19}$$

The derivatives of the three terms are given by

$$\begin{aligned}
\frac{\partial[4P(\mathbf{t}_L)P(\mathbf{t}_R)]^\alpha}{\partial s} &= 4\alpha[4P(\mathbf{t}_L)P(\mathbf{t}_R)]^{\alpha-1}p(\mathbf{t}_L)(1-2P(\mathbf{t}_L)) \\
\frac{\partial\Xi(s,\mathbf{t})}{\partial s} &= p(\mathbf{t}_L)\left(\mathbb{E}[Y|\mathbf{X}\in\mathbf{t}, X = s] - \mathbb{E}[Y|\mathbf{X}\in\mathbf{t}]\right) \\
&= p(\mathbf{t}_L)\bar{G}(s,\mathbf{t}) \\
\frac{\partial[P(\mathbf{t}_L)P(\mathbf{t}_R)]^{-1}}{\partial s} &= (-1)\ p(\mathbf{t}_L)(1-2P(\mathbf{t}_L))
\end{aligned}$$

where $p(\mathbf{t}_L)$ is the density function $\frac{\partial P(\mathbf{t}_L)}{\partial s}$. Taking the derivative with respect to s and substituting the above yields

$$\begin{aligned}
\frac{\partial \Delta_\alpha(s,\mathbf{t})}{\partial s} &= \frac{\Xi(s,\mathbf{t})^2}{P(\mathbf{t}_L)P(\mathbf{t}_R)}\ \frac{\partial [4P(\mathbf{t}_L)P(\mathbf{t}_R)]^\alpha}{\partial s} + \frac{[4P(\mathbf{t}_L)P(\mathbf{t}_R)]^\alpha}{P(\mathbf{t}_L)P(\mathbf{t}_R)}\ \frac{\partial [\Xi(s,\mathbf{t})]^2}{\partial s} \\
&\qquad + [4P(\mathbf{t}_L)P(\mathbf{t}_R)]^\alpha \Xi(s,\mathbf{t})^2\ \frac{\partial [P(\mathbf{t}_L)P(\mathbf{t}_R)]^{-1}}{\partial s} \\
&= \frac{[4P(\mathbf{t}_L)P(\mathbf{t}_R)]^\alpha \Xi(s,\mathbf{t}) p(\mathbf{t}_L)}{(P(\mathbf{t}_L)P(\mathbf{t}_R))^2} \left[\Xi(s,\mathbf{t})\alpha(1-2P(\mathbf{t}_L)) + 2\bar{G}(s,\mathbf{t})P(\mathbf{t}_L)P(\mathbf{t}_R) - \Xi(s,\mathbf{t})(1-2P(\mathbf{t}_L))\right]
\end{aligned}$$

To find the maximum decrease in impurity, we set $\frac{\partial \Delta_\alpha(s,\mathbf{t})}{\partial s} = 0$. Using (19) then results in

$$\begin{aligned}
2\ \bar{G}(s^*_\alpha,\mathbf{t})\ \Xi(s^*_\alpha,\mathbf{t})\ P(\mathbf{t}_L)P(\mathbf{t}_R) &= (1-\alpha)\ \Xi(s^*_\alpha,\mathbf{t})^2(1-2P(\mathbf{t}_L)) \\
2\ \bar{G}(s^*_\alpha,\mathbf{t})\ \Xi(s^*_\alpha,\mathbf{t}) &= (1-\alpha)\ \Delta(s^*_\alpha,\mathbf{t})\ \frac{(1-2P(\mathbf{t}_L))}{[4P(\mathbf{t}_L)P(\mathbf{t}_R)]^\alpha} \\
1-2P(\mathbf{t}_L) &= \frac{2\ \bar{G}(s^*_\alpha,\mathbf{t})\ \Xi(s^*_\alpha,\mathbf{t})}{\Delta(s^*_\alpha,\mathbf{t})}\ \frac{[4P(\mathbf{t}_L)P(\mathbf{t}_R)]^\alpha}{(1-\alpha)} \\
P(\mathbf{t}_L) &= \frac{1}{2} - \frac{\bar{G}(s^*_\alpha,\mathbf{t})\ \Xi(s^*_\alpha,\mathbf{t})}{(1-\alpha)\ \Delta(s^*_\alpha,\mathbf{t})}\ [4P(\mathbf{t}_L)P(\mathbf{t}_R)]^\alpha \\
&= \frac{1}{2} - \frac{\bar{G}(s^*_\alpha,\mathbf{t})\ \sqrt{\Delta(s^*_\alpha,\mathbf{t})P(\mathbf{t}_L)P(\mathbf{t}_R}}{(1-\alpha)\ \Delta(s^*_\alpha,\mathbf{t})}\ \frac{[4P(\mathbf{t}_L)P(\mathbf{t}_R)]^\alpha}{[4P(\mathbf{t}_L)P(\mathbf{t}_R)]^{\alpha/2}} \\
&= \frac{1}{2} - \frac{\bar{G}(s^*_\alpha,\mathbf{t})}{2(1-\alpha)\ \sqrt{\Delta(s^*_\alpha,\mathbf{t})}}\ [4P(\mathbf{t}_L)P(\mathbf{t}_R)]^{\frac{\alpha+1}{2}} \qquad (20)
\end{aligned}$$

Most expressions above like the decrease in impurity and the demeaned partial dependence function depend on the regularizer α. With a different weight, the maximization problem would yield a different split point and, consequently, also different decreases in impurity and partial dependence functions.

We are looking for an expression of node balancedness $\lambda = 4P(\mathbf{t}_L)P(\mathbf{t}_R)$. To arrive at a lower bound for λ, we substitute in the previous result:

$$\begin{aligned}
4P(\mathbf{t}_L)P(\mathbf{t}_R) &= 4P(\mathbf{t}_L)(1-P(\mathbf{t}_L)) \\
&= 4\left(\frac{1}{2} - \frac{\bar{G}(s^*_\alpha,\mathbf{t})}{2(1-\alpha)\ \sqrt{\Delta(s^*_\alpha,\mathbf{t})}}\ [4P(\mathbf{t}_L)P(\mathbf{t}_R)]^{\frac{\alpha+1}{2}}\right)\ \left(\frac{1}{2} + \frac{\bar{G}(s^*_\alpha,\mathbf{t})}{2(1-\alpha)\ \sqrt{\Delta(s^*_\alpha,\mathbf{t})}}\ [4P(\mathbf{t}_L)P(\mathbf{t}_R)]^{\frac{\alpha+1}{2}}\right) \\
&= 4\left(\frac{1}{4} - \frac{\bar{G}(s^*_\alpha,\mathbf{t})^2}{4(1-\alpha)^2\Delta(s^*_\alpha,\mathbf{t})}\ [4P(\mathbf{t}_L)P(\mathbf{t}_R)]^{\alpha+1}\right) \\
\lambda &= 1 - \frac{\bar{G}(s^*_\alpha,\mathbf{t})^2}{(1-\alpha)^2\Delta(s^*_\alpha,\mathbf{t})}\ \lambda^{\alpha+1} \qquad (21)
\end{aligned}$$

From now on, we denote by λ_α the node balancedness for a node split by the weighted splitting rule. Now define $\nu = \frac{(1-\alpha)^2\Delta(s^*_\alpha,\mathbf{t})}{\bar{G}(s^*_\alpha,\mathbf{t})^2}$. Then one can find a lower bound for λ_α by

$$\begin{aligned}
\lambda_\alpha &= 1 - \frac{1}{\nu}\lambda_\alpha^{\alpha+1} \\
\lambda_\alpha^{\alpha+1} &= \nu(1-\lambda_\alpha) \\
\lambda_\alpha &= \nu^{\frac{1}{\alpha+1}}(1-\lambda_\alpha)^{\frac{1}{\alpha+1}} \\
\lambda_\alpha &\geq \nu^{\frac{1}{\alpha+1}}(1-\lambda_\alpha) \\
\lambda_\alpha &\geq \frac{\nu^{\frac{1}{\alpha+1}}}{1+\nu^{\frac{1}{\alpha+1}}}
\end{aligned}$$

Plugging the original terms back finally yields the result

$$\begin{aligned}
\lambda_\alpha &\geq \frac{\left(\frac{(1-\alpha)^2\Delta(s^*_\alpha,\mathbf{t})}{\bar{G}(s^*_\alpha,\mathbf{t})^2}\right)^{\frac{1}{\alpha+1}}}{1+\left(\frac{(1-\alpha)^2\Delta(s^*_\alpha,\mathbf{t})}{\bar{G}(s^*_\alpha,\mathbf{t})^2}\right)^{\frac{1}{\alpha+1}}} \\
\lambda_\alpha &\geq \frac{\left(\frac{(1-\alpha)^2\Delta(s^*_\alpha,\mathbf{t})}{\bar{G}(s^*_\alpha,\mathbf{t})^2}\right)^{\frac{1}{\alpha+1}}}{\left(\frac{\bar{G}(s^*_\alpha,\mathbf{t})^2+(1-\alpha)^2\Delta(s^*_\alpha,\mathbf{t})}{\bar{G}(s^*_\alpha,\mathbf{t})^2}\right)^{\frac{1}{\alpha+1}}} \\
\lambda_\alpha &\geq \left(\frac{(1-\alpha)^2\Delta(s^*_\alpha,\mathbf{t})}{\bar{G}(s^*_\alpha,\mathbf{t})^2+(1-\alpha)^2\Delta(s^*_\alpha,\mathbf{t})}\right)^{\frac{1}{\alpha+1}} \qquad (22)
\end{aligned}$$

The expressions in Lemma 1 are a simplified version of the just presented result. By setting $\alpha = 0$, one gets the original CART algorithm, and the expressions are identical to the ones found in Klusowski (2019). To see this, consider (20) and denote by s^* the optimal split point if $\alpha = 0$. Then expression (20) simplifies to

$$P(\mathbf{t}_L) = \frac{1}{2} - \frac{\bar{G}(s^*,\mathbf{t})}{\sqrt{\Delta(s^*,\mathbf{t})}}\,[P(\mathbf{t}_L)P(\mathbf{t}_R)]^{1/2}$$

The solution to a quadratic expression $x = 1/2 - c\sqrt{x(1-x)}$ is given by $x = \frac{c^2 \pm \sqrt{c^4-c^2}+1}{2(c^2+1)}$. Moreover, denote by λ the node balancedness in the original CART algorithm. Then,

$$
\begin{aligned}
P(\mathbf{t}_L) &= \frac{\frac{\bar{G}(s^*,\mathbf{t})^2}{\Delta(s^*,\mathbf{t})} + 1 \pm \sqrt{\frac{\bar{G}(s^*,\mathbf{t})^4}{\Delta(s^*,\mathbf{t})^2} - \frac{\bar{G}(s^*,\mathbf{t})^2}{\Delta(s^*,\mathbf{t})}}}{2\left(\frac{\bar{G}(s^*,\mathbf{t})^2}{\Delta(s^*,\mathbf{t})} + 1\right)} \\
&= \frac{1}{2}\left(1 \pm \frac{\sqrt{\frac{\bar{G}(s^*,\mathbf{t})^4 + \bar{G}(s^*,\mathbf{t})^2 + \Delta(s^*,\mathbf{t})}{\Delta(s^*,\mathbf{t})^2}}}{\frac{\bar{G}(s^*,\mathbf{t})^2}{\Delta(s^*,\mathbf{t})} + 1}\right) \\
&= \frac{1}{2}\left(1 \pm \frac{\bar{G}(s^*,\mathbf{t})^2}{\Delta(s^*,\mathbf{t})} \; \frac{\Delta(s^*,\mathbf{t})}{\bar{G}(s^*,\mathbf{t})^2 + \Delta(s^*,\mathbf{t})}\right) \\
&= \frac{1}{2}\left(1 \pm \frac{\bar{G}(s^*,\mathbf{t})^2}{\bar{G}(s^*,\mathbf{t})^2 + \Delta(s^*,\mathbf{t})}\right)
\end{aligned}
$$

which proves (15). The expression (16) describing the node balancedness for the CART algorithm follows from (21) and by setting $\alpha = 0$:

$$
\begin{aligned}
\lambda &= 1 - \frac{\bar{G}(s^*,\mathbf{t})^2}{\Delta(s^*,\mathbf{t})} \; \lambda \\
&= \frac{\Delta(s^*,\mathbf{t})}{\bar{G}(s^*,\mathbf{t})^2 + \Delta(s^*,\mathbf{t})}
\end{aligned}
$$

Proof Theorem 1

Denote with $\mathbf{t}'$ the ancestor node of $\mathbf{t}$. Suppose that we split along the dimension X and that our split point is s^*. Then, $\mathbb{P}[\mathbf{X} \in [a(\mathbf{t}'), b(\mathbf{t}')]]$ is less or equal to

$$
\begin{aligned}
&\max\{\mathbb{P}[\mathbf{X} \in \mathbf{t}', X \le s^*],\ \mathbb{P}[\mathbf{X} \in \mathbf{t}', X > s^*]\} \\
&\quad = \mathbb{P}[\mathbf{X} \in [a(\mathbf{t}'), b(\mathbf{t}')]]\ \max\{\mathbb{P}[X \le s^* \mid \mathbf{X} \in [a(\mathbf{t}'), b(\mathbf{t}')]],\ \mathbb{P}[X > s^* \mid \mathbf{X} \in [a(\mathbf{t}'), b(\mathbf{t}')]]\} \\
&\quad \le \mathbb{P}[\mathbf{X} \in [a(\mathbf{t}'), b(\mathbf{t}')]]\ \exp\{-\eta P^*(\mathbf{t}_L)(1 - P^*(\mathbf{t}_L)\} \\
&\quad = \mathbb{P}[\mathbf{X} \in [a(\mathbf{t}'), b(\mathbf{t}')]]\ \exp\{-\frac{\eta}{4}\ \lambda_j\} \\
&\quad \le \mathbb{P}[\mathbf{X} \in [a(\mathbf{t}'), b(\mathbf{t}')]]\ \exp\left\{-\frac{\eta}{4}\left(\frac{(1-\alpha)^2\Delta(s^*_\alpha,\mathbf{t}')}{\bar{G}(s^*_\alpha,\mathbf{t}')^2 + (1-\alpha)^2\Delta(s^*_\alpha,\mathbf{t}')}\right)^{\frac{1}{\alpha+1}}\right\} \\
&\quad = \mathbb{P}[\mathbf{X} \in [a(\mathbf{t}'), b(\mathbf{t}')]]\ \exp\left\{-\frac{\eta}{4}\left(\frac{(1-\alpha)^2\Delta(s^*_\alpha,\mathbf{t}')^{-\alpha}}{\bar{G}(s^*_\alpha,\mathbf{t}')^2 + (1-\alpha)^2\Delta(s^*_\alpha,\mathbf{t}')}\right)^{\frac{1}{\alpha+1}}\ \Delta(s^*_\alpha,\mathbf{t}')\right\}
\end{aligned}
$$

The first inequality is the result of Assumption 2 and the fact that $\max\{p, 1-p\} \le e^{-p(1-p)}$ for $p \in [0,1]$. The second inequality is due to (22) from Lemma 1.

Consider now the case where $\mathbf{t} = \Pi_{j=1}^{d}[a_j(\mathbf{t}), b_j(\mathbf{t})]$ is a terminal node. To arrive at the terminal node, one repeatedly splits the ancestor nodes. Thus, by induction and because $\mathbb{P}[\mathbf{X} \in [0,1]] = 1$, it holds that:

$$\mathbb{P}[a_j(\mathbf{t}) \leq X \leq b_j(\mathbf{t})] \leq \exp\left\{ -\frac{\eta}{4} \sum_{\substack{\mathbf{t}' \supset \mathbf{t} \\ j = j_{\mathbf{t}'}}} \left(\frac{(1-\alpha)^2 \Delta(j, s^*_\alpha, \mathbf{t}')^{-\alpha}}{\bar{G}_j(s^*_\alpha, \mathbf{t}')^2 + (1-\alpha)^2 \Delta(j, s^*_\alpha, \mathbf{t}')} \right)^{\frac{1}{\alpha+1}} \Delta(j, s^*_\alpha, \mathbf{t}') \right\}$$

$$= \exp\left\{ -\frac{\eta}{4} \sum_{\substack{\mathbf{t}' \supset \mathbf{t}' \\ j = j_{\mathbf{t}'}}} w(j, s^*_\alpha, \mathbf{t}') \, \Delta(j, s^*_\alpha, \mathbf{t}') \right\}$$

$$= \exp\left\{ -\frac{\eta}{4} \, MDI(X_j, \mathbf{t}) \right\}$$

which proves Theorem 1 with the weights $w(j, s^*_\alpha, \mathbf{t}') = \left(\frac{(1-\alpha)^2 \Delta(j, s^*_\alpha, \mathbf{t}')^{-\alpha}}{\bar{G}_j(s^*_\alpha, \mathbf{t}')^2 + (1-\alpha)^2 \Delta(j, s^*_\alpha, \mathbf{t}')} \right)^{\frac{1}{\alpha+1}}$.
When looking at the particular case of $\alpha = 0$ (CART algorithm), the weights simplify to

$$w(j, s^*_{\alpha=0}, \mathbf{t}') = \frac{1}{\bar{G}_j(s^*_{\alpha=0}, \mathbf{t}')^2 + \Delta(j, s^*_{\alpha=0}, \mathbf{t}')}.$$

Proof Lemma 2

We want to show that $\bar{G}_j(s, \mathbf{t}) > 0$ and $\Delta(j, s^*, \mathbf{t}) > 0$ for all $j \in \mathcal{S}$. For this we look at a generic X_j and the regressor matrix without variable j is given by $\mathbf{X}_{\setminus j} \in [0,1]^{d-1}$.
By definition of the partial dependence function and the Leibniz integral rule we can re-write the derivative of $\bar{F}_j(x_j, \mathbf{t})$:

$$\frac{\partial^r \bar{F}_j(x_j, \mathbf{t})}{\partial x_j^r} = \int_0^1 \frac{\partial^r m(x_j, \mathbf{x}_{\setminus j})}{\partial x_j^r} \, \mathbb{P}_{\mathbf{X}_{\setminus j} | \mathbf{X} \in \mathbf{t}} d(\mathbf{x}_{\setminus j})$$

Applying the mean value theorem for integrals, there exists a $\mathbf{x}'_{\setminus j} \in \mathbf{t}_{\setminus j}$ for which

$$\frac{\partial^r m(x_j, \mathbf{x}'_{\setminus j})}{\partial x_j^r} = \frac{\partial^r \bar{F}_j(x_j, \mathbf{t})}{\partial x_j^r}.$$

Assumption 3 states that there is a finite integer R such that for $1 \geq r \geq R$ and each $x_j \in [a_j(\mathbf{t}), b_j(\mathbf{t})]$, it holds that $\frac{\partial^r m(x_j, \mathbf{x}_{\setminus j})}{\partial x_j^r} \neq 0$ for all $\mathbf{x}_{\setminus j} \in [0,1]^{d-1}$. Since this is true for all $\mathbf{x}_{\setminus j}$, it also holds for $\mathbf{x}'_{\setminus j}$.
Consequently,

$$\frac{\partial^r m(x_j, \mathbf{x}'_{\setminus j})}{\partial x_j^r} = \frac{\partial^r \bar{F}_j(x_j, \mathbf{t})}{\partial x_j^r} \neq 0.$$

Thus, there exists a derivative of $\bar{F}_j(x_j, \mathbf{t})$ of finite order r which is not zero. Because the boundaries of $[a_j(\mathbf{t}), b_j(\mathbf{t})]$ are arbitrarily set, it follows that $\bar{F}_j(x_j, \mathbf{t})$ is not constant on $[a_j(\mathbf{t}), b_j(\mathbf{t})]$.

We have $\bar{G}_j(x_j, \mathbf{t}) = 0$ if and only if

$$\bar{F}_j(x_j, \mathbf{t}) = \mathbb{E}[Y \mid \mathbf{X} \in \mathbf{t}, X_j = x_j] = \mathbb{E}[Y \mid \mathbf{X} \in \mathbf{t}]$$

which is not possible since there exists an interval $[a_j(\mathbf{t}), b_j(\mathbf{t})]$ within which $\bar{F}_j(x_j, \mathbf{t})$ is not constant. This assumption effectively guarantees that all strong variables $j \in \mathcal{S}$ have a sufficient effect on the outcome variable Y. This is ensured by forcing the slope of the regression function not to be too "flat".

In addition, Assumption 3 also guarantees that $\Delta(j, s^*, \mathbf{t}) > 0$ for all $j \in \mathcal{S}$. $\Delta(j, s^*, \mathbf{t}) = 0$ is the maximum of $\Delta(j, s, \mathbf{t})$ over all s. This maximum is zero only if the variances of the child nodes $Var[Y \mid \mathbf{X} \in \mathbf{t}, X_j \leq s] = Var[Y \mid \mathbf{X} \in \mathbf{t}, X_j > s]$ are the same for every possible split point s. The variances being equal over the whole range of X_j would again mean that the conditional dependence function $\bar{F}_j(x_j, \mathbf{t})$ is zero, which we ruled out before. Therefore, $\Delta(j, s^*, \mathbf{t}) > 0$ for all strong variables $j \in \mathcal{S}$.

Proof Theorem 2

Since in our setup $K_j(\mathbf{t}) \to \infty$ for all strong variables according to Scornet et al. (2015), the goal is to show that $\Lambda_j > 0$.
The global node balancedness $\Lambda_j = \inf_{\mathbf{t}'} \lambda_j$ where the infimum is taken over all ancestor nodes $\mathbf{t}'$.

$$\begin{aligned}\Lambda_j &= \inf_{\mathbf{t}'} \lambda_j \\ &\geq \inf_{\mathbf{t}'} \left(\frac{(1-\alpha)^2 \Delta(s^*_\alpha, \mathbf{t}')}{\bar{G}(s^*_\alpha, \mathbf{t}')^2 + (1-\alpha)^2 \Delta(s^*_\alpha, \mathbf{t}')} \right)^{\frac{1}{\alpha+1}}\end{aligned}$$

where we use (22) from Lemma 1. In Lemma 2 we showed that both $\Delta(s^*_\alpha, \mathbf{t}) > 0$ and $\bar{G}(s^*_\alpha, \mathbf{t}) > 0$ for any node $\mathbf{t}$. While the global node balancedness can be arbitrarily small, $\Lambda_j > 0$ strictly holds.

Proof Theorem 3

Here we follow the approach laid out in Scornet (2016) and Klusowski (2019). The goal is to show that the node diameter $diam_{\mathcal{S}}(\mathbf{t}(\mathbf{X}, \Theta)) \to \infty$ for all strong variables $j \in \mathcal{S}$. If this is achieved, our setup is identical to the assumptions of Theorem 4.1 from the consistency proof of Scornet (2016). We include the randomization process Θ_l to emphasize that these results depend on the underlying data.
From now on let us consider all strong variables and a split along dimension j and split point s. Then, let $H = \{\mathbf{X} : X_j = s\}$ be a collection of all observations for which the j^{th} coordinate is equal to s. Further, denote with $D = \{A : A \cap H \neq \emptyset\}$ a set of nodes which contain observations for which $X_j = s$

holds. Also, let $j_k(\Theta)$ be the splitting dimension at the k^{th} level of the tree. Suppose that a node $\mathbf{t}_k(\mathbf{X},\Theta) \in D$ and that we want to split this node. In this case, we are interested in the upper bound of $\mathbb{P}[\mathbf{t}_{k+1}(\mathbf{X},\Theta) \in D|\Theta]$.

Splitting the original node $\mathbf{t}_k(\mathbf{X},\Theta)$ can happen in two ways:

1. Dimension j is chosen as splitting dimension for $\mathbf{t}_k(\mathbf{X},\Theta)$. In this case, the node is split exactly along s and there will be one child node containing all observations for which $X_j = s$ and the other one will have an empty intersection with H.
2. Any other dimension but j is chosen as splitting dimension for $\mathbf{t}_k(\mathbf{X},\Theta)$. Now, since the regressors are independent, both child nodes have a nonempty intersection with H.

The probability of the next node $\mathbf{t}_{k+1}(\mathbf{X},\Theta)$ containing observations for which $X_j = s$ is then given by:

$$\begin{aligned}
&\mathbb{P}[\mathbf{t}_{k+1}(\mathbf{X},\Theta) \in D|\Theta] \\
&\quad = \mathbb{E}\,[\underbrace{\mathbb{1}\{j_{k+1}(\Theta)=j\}\ \mathbb{1}\{\mathbf{t}_{k+1}(\mathbf{X},\Theta)\in D\}}_{\text{Case 1}} + \underbrace{\mathbb{1}\{j_{k+1}(\Theta)\neq j\}\ \mathbb{1}\{\mathbf{t}_{k+1}(\mathbf{X},\Theta)\in D\}}_{\text{Case 2}}] \\
&\quad \le \mathbb{1}\{j_{k+1}(\Theta)=j\}\ \mathbb{P}[\mathbf{t}_{k+1}(\mathbf{X},\Theta)\in D|\Theta] + \mathbb{1}\{j_{k+1}(\Theta)\neq j\}\ \mathbb{P}[\mathbf{t}_{k+1}(\mathbf{X},\Theta)\in D|\Theta] \\
&\quad \le \mathbb{1}\{j_{k+1}(\Theta)=j\}\ (1-P(\mathbf{t}_{k_L})P(\mathbf{t}_{k_R}))\ \mathbb{P}[\mathbf{t}_{k}(\mathbf{X},\Theta)\in D|\Theta] + \mathbb{1}\{j_{k+1}(\Theta)\neq j\}\ \mathbb{P}[\mathbf{t}_{k}(\mathbf{X},\Theta)\in D|\Theta] \\
&\quad = \mathbb{1}\{j_{k+1}(\Theta)=j\}\ (1-\frac{1}{4}\lambda_j(\mathbf{t}_k(\mathbf{X},\Theta)))\ \mathbb{P}[\mathbf{t}_{k}(\mathbf{X},\Theta)\in D|\Theta] + (1-\mathbb{1}\{j_{k+1}(\Theta)=j\})\ \mathbb{P}[\mathbf{t}_{k}(\mathbf{X},\Theta)\in D|\Theta] \\
&\quad = \left(1-\frac{1}{4}\lambda_j(\mathbf{t}_k(\mathbf{X},\Theta))\ \mathbb{1}\{j_{k+1}(\Theta)=j\}\right)\ \mathbb{P}[\mathbf{t}_{k}(\mathbf{X},\Theta)\in D|\Theta] \\
&\quad \le \exp\left\{-\frac{1}{4}\lambda_j(\mathbf{t}_k(\mathbf{X},\Theta))\ \mathbb{1}\{j_{k+1}(\Theta)=j\}\ \mathbb{P}[\mathbf{t}_{k}(\mathbf{X},\Theta)\in D|\Theta]\right\}. \qquad (23)
\end{aligned}$$

Here, the inequalities follow from the fact that in Case 1, the probability that one of the resulting child nodes has a nonempty intersection with H can be upper-bounded with $\max(P(\mathbf{t}_{k_L}), P(\mathbf{t}_{k_R})) \le 1-P(\mathbf{t}_{k_L})P(\mathbf{t}_{k_R}))$. The last inequality is the result of $1+p \le \exp\{p\}$.
Since a node is the result of recursive partitions, the probability of $\mathbf{t}_{k+1}(\mathbf{X},\Theta) \in D$ is a function of the ancestor nodes. Thus, (23) is at most:

$$\begin{aligned}
\exp\left\{-\frac{1}{4}\ \sum_{l=1}^{k}\lambda_j(\mathbf{t}_l(\mathbf{X},\Theta))\ \mathbb{1}\{j_{l+1}(\Theta)=j\}\right\} &\le \exp\left\{-\frac{1}{4}\ \inf_{\mathbf{t}}\ \lambda_j(\mathbf{t}(\mathbf{X},\Theta))K_j(\mathbf{t})\right\} \\
&\le \exp\left\{-\frac{1}{4}\ \min_{\mathbf{t}}\ MDI(X_j,\mathbf{t})\right\}
\end{aligned}$$

where K_j is the number of times that dimension j was chosen as splitting dimension. The last inequality is a direct result of Theorem 2. Then, $\mathbb{P}[\mathbf{t}_{k+1}(\mathbf{X},\Theta) \in D|\Theta] \to 0$ as $MDI(X_j,\mathbf{t}) \to \infty$ since we assumed that $\min_{\mathbf{t}} MDI(X_j,\mathbf{t}) \to \infty\ \forall j \in \mathcal{S}$.

Finally, to make the connection to the diameter of the nodes, recall that the regression tree forms partitions of $[0,1]^{\mathcal{S}}$. Here we only consider the set of strong variables $j \in \mathcal{S}$ because in the infinite sample setting, the algorithm never selects a weak variable as splitting dimension. Consider now the partition of $[0,1]^{\mathcal{S}}$ into hypercubes with a side length ϵ and sides given by hyperplanes $\mathbf{x} : x_{j'} = \ell\epsilon$ for $j' \in \mathcal{S}$ and $\ell = \{0, \frac{1}{\epsilon}, \frac{2}{\epsilon}, \ldots, 1\}$. The diameter of these hypercubes is given by $\sqrt{S}\epsilon$ where S denotes the number of strong variables. If a node $\mathbf{t}(\mathbf{X}, \Theta)$ is part of only one of these hypercubes, it's node diameter $diam_{\mathcal{S}}(\mathbf{t}(\mathbf{X}, \Theta)) \leq \sqrt{S}\epsilon$. If D encompasses all hyperplanes, the probability that any node diameter is larger than $\sqrt{S}\epsilon$ can be upper-bounded by:

$$\mathbb{P}\left[diam_{\mathcal{S}}(\mathbf{t}(\mathbf{X}, \Theta)) > \sqrt{S}\epsilon\right] \leq \mathbb{P}\left[\bigcup_{D}\{\mathbf{t}(\mathbf{X}, \Theta) \in D\}\right] \to 0$$

if $\min_{\mathbf{t}} MDI(X_j, \mathbf{t}) \to \infty$. Thus, in our setting the node diameters tend to zero if the MDI of all strong variables approaches infinity. With this, all conditions for the proof of Scornet (2016) hold and Theorem 4.1 from their paper applies.

Affidavit

I affirm that this Master thesis was written by myself without any unauthorized third-party support. All used references and resources are clearly indicated. All quotes and citations are properly referenced. This thesis was never presented in the past in the same or similar form to any examination board. I agree that my thesis may be subject to electronic plagiarism check. For this purpose an anonymous copy may be distributed and uploaded to servers within and outside the University of Mannheim.

Mannheim, 03.09.2021 Christian Hilscher