



Article

# Smooth Sigmoid Surrogate (SSS): An Alternative to Greedy Search in Decision Trees

Xiaogang Su <sup>1</sup>, George Ekow Quaye <sup>2,\*</sup>, Yishu Wei <sup>3</sup>, Joseph Kang <sup>4</sup>, Lei Liu <sup>5</sup>, Qiong Yang <sup>6</sup> and Juanjuan Fan <sup>7</sup> and Richard A. Levine <sup>7</sup><sup>1</sup> Department of Mathematical Science, University of Texas at El Paso, El Paso, TX 79968, USA; xsu@utep.edu<sup>2</sup> Division of Health Services and Outcomes Research, Children's Mercy Kansas City, Kansas City, MO 64108, USA<sup>3</sup> Reddit Inc., San Francisco, CA 94102, USA; yishuwei2019@u.northwestern.edu<sup>4</sup> US Census Bureau, Washington, DC 20233, USA; joseph.kang@census.gov<sup>5</sup> Division of Biostatistics, Washington University in St. Louis, St. Louis, MO 63110, USA; lei.liu@wustl.edu<sup>6</sup> Department of Biostatistics, School of Public Health, Boston University, Boston, MA 02118, USA; qyang@bu.edu<sup>7</sup> Department of Mathematics and Statistics, San Diego State University, San Diego, CA 92182, USA; jjfan@sdsu.edu (J.F.); rlevine@sdsu.edu (R.A.L.)

\* Correspondence: gequaye@cmh.edu

**Abstract:** Greedy search (GS) or exhaustive search plays a crucial role in decision trees and their various extensions. We introduce an alternative splitting method called smooth sigmoid surrogate (SSS) in which the indicator threshold function used in GS is approximated by a smooth sigmoid function. This approach allows for parametric smoothing or regularization of the erratic and discrete GS process, making it more effective in identifying the true cutoff point, particularly in the presence of weak signals, as well as less prone to the inherent end-cut preference problem. Additionally, SSS provides a convenient means of evaluating the best split by referencing a parametric nonlinear model. Moreover, in many variants of recursive partitioning, SSS can be reformulated as a one-dimensional smooth optimization problem, rendering it computationally more efficient than GS. Extensive simulation studies and real data examples are provided to evaluate and demonstrate its effectiveness.

**Keywords:** CART; decision trees; end-cut preference; greedy search; recursive partitioning; sigmoid function

**MSC:** 62G08; 62N01; 62J02



**Citation:** Su, X.; Quaye, G.E.; Wei, Y.; Kang, J.; Liu, L.; Yang, Q.; Fan, J.; Levine, R.A. Smooth Sigmoid Surrogate (SSS): An Alternative to Greedy Search in Decision Trees. *Mathematics* **2024**, *12*, 3190. <https://doi.org/10.3390/math12203190>

Academic Editor: Danilo Costarelli

Received: 30 August 2024

Revised: 1 October 2024

Accepted: 10 October 2024

Published: 11 October 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Recursive partitioning or tree-structured methods [1,2], together with their various extensions such as multivariate adaptive regression splines MARS; [3], bagging [4], boosting [5], and random forests [6], have formed an important class of learning and modeling tools for pattern recognition, data mining, and statistical analysis. Greedy search (GS, also called exhaustive search or brute-force search) is the dominating method for identifying the best split or knot at each step of recursive partitioning. In this discrete process, all permissible splits are compared via some criterion and the best one is selected.

Consider data that consist of i.i.d. observations  $\{(y_i, \mathbf{x}_i) : i = 1, \dots, n\}$ , where  $y_i$  is the  $i$ -th observed value on the response or target variable  $Y$  and  $\mathbf{x}_i \in \mathbb{R}^p$  is the associated input vector on predictor variables  $(X_1, \dots, X_p)$ . The response  $Y$  can be either continuous, as in regression trees, or categorical, as in classification trees, while the predictor variables can be of mixed types. A split of data is induced by a binary question such as “is  $X_j$  greater than  $c$ ?” for a continuous  $X_j$  or “does  $X_j$  belong to  $C_0$ ?” for a categorical  $X_j$ , where  $c$  is a cutoff point (or cutpoint for short) and  $C_0$  is a subset of categories or levels that  $X_j$  can assume. A greedy search for the best split can be viewed as a two-step process. In the first step,

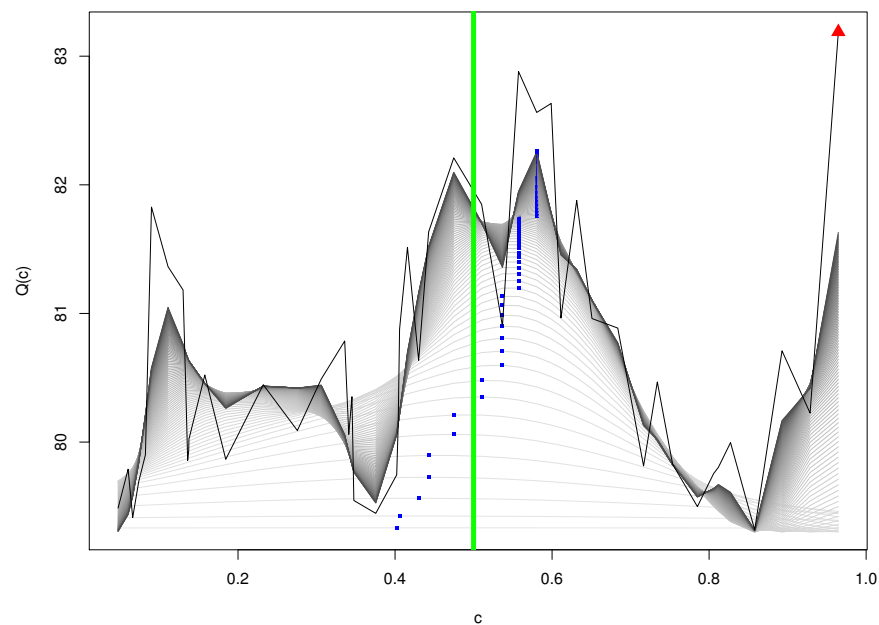
the best way to split each predictor, i.e., the best cutoff point or the best subset, is sought; then, in the second step, a comparison is made across all  $p$  predictors.

Consider the first step, namely, how to determine the best cutoff point for a given predictor  $X_j$ . There are two general approaches of splitting data in the literature: either by minimizing within-node impurity, or by maximizing between-node difference. The conventional approach, as in CART ([1], pp. 230–232), splits data by minimizing the within-node impurity. Let  $i(t)$  denote the impurity measure for node  $t$  and let  $\Delta i = i(t) - \{p(t_L)i(t_L) + p(t_R)i(t_R)\}$  denote the reduction in impurity due to a split of the data, where  $t_L$  and  $t_R$  denote the left and right child nodes, while  $p(t_L)$  is the proportion of observations falling into  $t_L$  and analogously for  $p(t_R)$ . For splits based on variable  $X$ , the best cutoff point  $c_j^*$  achieves the greatest reduction in the sense that  $c_j^* = \arg \max_c \Delta i$ . The other splitting approach is to maximize the between-node heterogeneity, as examined by LeBlanc and Crowley [7]. This approach is somehow more appealing from the statistical perspective and allows for the extension of tree methods to other data types, as any appropriately defined two-sample statistic may be used to bisect the data. In many scenarios, these two approaches lead to equivalent splitting criteria.

Although GS is the standard method for splitting data, it has a few deficiencies. First of all, the splitting statistic in GS is a random process that shows erratic patterns with substantial variations, as mentioned by [8] (Figure 1 on p. 8) in a recent **rpart** documentation, for example. As a result, it often occurs that local spikes overpower the overall peak associated with the true cutpoint, especially when the signal strength is relatively weak. Second, some commonly used splitting statistics in GS tend to favor unbalanced cuts that leave a child node with very small sample sizes, a problem referred to as end-of-cut preference (see Section 11.8 of CART [1]). This would eventually lead to tree structures that are hard to interpret. Figure 1 provides an illustration of the problems with simulated data. The goodness-of-split measures shows an overall pattern climaxing at the true cutpoint of 0.5, yet with high variation and many high local spikes; in particular, one prevailing local peak at the end is erroneously selected by GS. Third, variable selection bias has been considered inherent with GS. This refers to the phenomenon of a predictor with more values or levels being more likely to be selected as the splitting variable than a predictor with fewer values or levels. Loh and colleagues have pioneered excellent efforts in addressing this issue with a series of papers, as exemplified in Loh [9]. They also view GS as a two-step process, yet in a different way. Their main idea is to first select the most important variable and then evaluate splits based on the selected variable only. Another suggestion regarding this issue is to derive the distribution of maximally selected test statistics. This approach is exemplified by Miller and Siegmund [10], Shih and Tsai [11], and Hothorn and Zeileis [12]. Finally, searching over all permissible cutoff points for each predictor can be time-consuming. One common way for speeding up GS is to apply an updating formula to compute the splitting statistic for two consecutive cutoff points on a predictor. Examples can be seen in (Section 2.2, [13]), LeBlanc and Crowley [7], and Su, Tsai, and Wang [14] (Appendix A). Other strategies include splitting on percentiles, sub-sampling, and online incremental learning (see, e.g., [15]), which nevertheless reduces the search space and consequently provides no guarantee of success in finding the optimal cutoff point. Despite all of these efforts, greedy search can be time-consuming, especially when dealing with continuous variables or categorical variables that have many levels in large datasets.

In this article, we propose an alternative splitting method to greedy search, termed smooth sigmoid surrogate (SSS). The main idea is to replace the step functions involved in splitting with smooth sigmoid functions. The best cutoff point for each predictor can be estimated as a parameter in a nonlinear model. Very often, the estimation can be appropriately reformulated into a one-dimensional optimization problem that can be quickly solved. We demonstrate that this new method provides superior performance to GS along with correction or amelioration of its common deficiencies. First of all, SSS facilitates a parametric smoothing or regularization of the erratic splitting statistics in GS, yielding improved stability in the objective function to be optimized. As a result, SSS is more capable of identifying weak signals. As shown in Figure 1, smoothing preserves

the overall parabola pattern while flattening the local spikes and reducing variations. A cutpoint close to the true one of 0.5 is successfully recovered for a range of smoothing parameter values. The smoothing effect also leads to substantial amelioration of the end-cut preference problem in practice. Furthermore, because the search for the best cutoff is cast in a nonlinear regression framework, conventional statistical inference can be exploited to facilitate convenient comparisons across different predictors for finding the best split of data. We show that this alternative splitting method helps to alleviate the variable selection bias problem. In addition, SSS potentially reduces computational cost without sacrificing performance. Lastly, SSS can be flexibly extended to many other recursive partitioning methods designated for different analytic purposes. In most scenarios, the optimization problem remains one-dimensional with appropriate reformulation.



**Figure 1.** Plot of the goodness-of-split measure  $\Delta_l(c)$  and its approximation  $\tilde{\Delta}_l(c)$  versus the permissible cutoff point  $c$ . Data of sample size  $n = 50$  were generated from Model A in (7) with  $\beta = 0.2$ . The dark line corresponds to the goodness-of-split measures or splitting statistics  $\Delta_l(c)$  in greedy search, while the lines in gray scale are approximated  $\tilde{\Delta}_l(c)$  values in SSS with  $a = 1, 2, \dots, 100$ . The red solid triangle point corresponds to the best cutpoint found by GS, while the blue disc points are the best cutpoints obtained from SSS with different  $a$  values. The solid green line corresponds to the true cutoff point  $c_0 = 0.5$ .

We note that replacing step functions with smooth sigmoid functions is not an entirely new idea. Notably, this tactic has impacted the development of artificial neural networks (ANN). One well-known initial ANN proposal by Rosenblatt [16] consisted of single-layer networks called perceptrons, which contain threshold activation functions. In later developments such as multilayer perceptrons (MLPs) the step function is replaced by the differentiable logistic function, which has advanced ANNs tremendously. In the recursive partitioning context, the hierarchical mixture of experts (HME) architecture studied by Jordan and Jacobs [17] is essentially a mixture model where mixing probabilities are hierarchically modeled via multivariate sigmoid or softmax functions. Along similar lines, Ciampi, Couturier, and Li [18] proposed a ‘soft tree’ procedure at each node in which an observation goes to the left or right child node with a certain probability that is estimated via soft threshold functions or logistic models. To the best of our knowledge, however, our current proposal is the first attempt to use the sigmoid function as an alternative to GS for partitioning data. Therneau [8] also noted the potential of nonparametric smoothing in stabilizing the erratic pattern shown on the splitting statistic in GS; however, he applied nonparametric smoothing splines directly to the computed goodness-of-split measures, which could only add to

the computational cost of GS. Comparatively, SSS facilitates a parametric way of smoothing that is embedded in the process that generates the splitting statistic.

The remainder of this paper is organized in the following manner: Section 2 introduces the SSS method for regression trees, where the response variable is continuous, and compares it to GS using various numerical studies in many aspects; classification trees with binary outcomes are treated in Section 3; Section 4 outlines the usage of SSS in several other recursive partitioning-based methods; finally, Section 4 ends the article with a brief discussion.

## 2. SSS for Regression Trees

In regression trees, the node impurity  $\iota(t)$  is commonly measured by the mean squared error  $\iota(t) = \sum_{i \in t} (y_i - \bar{y}_t)^2 / n_t$ . In this case, minimizing the within-node impurity is equivalent to maximizing a two-sample  $F$  test statistic that compares the two child nodes. For the time being, we consider continuous predictors only. The scenario for categorical predictors as well as binary variables will be discussed later. For simplicity, we denote  $X_j$  as  $X$ . The splitting statistic can be cast into the following linear model:

$$y_i = \beta_0 + \beta_1 \delta(x_i; c) + \varepsilon_i \tag{1}$$

where  $\delta(x_i; c) = I\{x_i \geq c\} \triangleq \delta_i$  is the indicator function corresponding to the split and  $\varepsilon_i \stackrel{\text{iid}}{\sim} N(0, \sigma^2)$ . The two-sample  $F$  test is the same as the squared  $t$  test for testing  $H_0 : \beta_1 = 0$  vs.  $H_a : \beta_1 \neq 0$ . The indicator function in (1) can be written as  $\delta(x; c) = I\{x - c \geq 0\} = H(x - c)$ , where  $H(x) = I\{x \geq 0\}$  denotes the Heaviside (unit) step function. Often used in integration, the value of  $H(x)$  at  $x = 0$  does not really matter, whereas one natural choice is  $H(0) = 1/2$  for the sake of rotational symmetry.

Given that this search must be carried out over all variables, this can lead to a significant computational strain in a big data context whenever many features are present, with each containing several potential splits; it also suffers from other problems, as discussed earlier. This motivates us to consider a new way of splitting data. Our idea is to approximate the threshold indicator function with a smooth sigmoid surrogate (SSS) function.

### 2.1. Sigmoid Functions

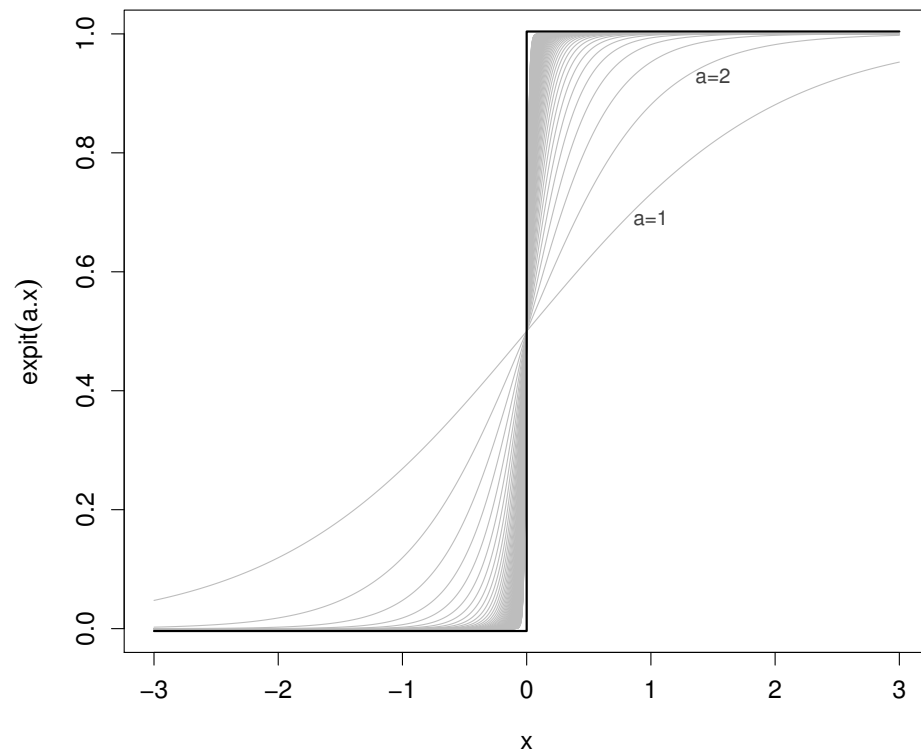
A sigmoid function is a mathematical function with an ‘S’ shape. Very often, the sigmoid function refers to the special case of the logistic function (also called the expit function):

$$\pi(x) = \pi(x; a) = \{1 + \exp(-ax)\}^{-1} = \frac{\exp(ax)}{1 + \exp(ax)}. \tag{2}$$

While there are various types of sigmoid functions, we primarily choose the logistic function due to its widespread use in statistics, such as in logistic regression, and the extensive research on its properties, e.g.,  $d\pi(x; a)/dx = a \pi(x; a) \{1 - \pi(x; a)\}$ . Readers are referred to Balakrishnan [19], Johnson, Kotz, and Balakrishnan [20] and Rosenblatt [16] (Section 23) for a book-length account of the logistic function and the logistic distribution as well as their properties and applications.

Figure 2 plots the logistic function with different values of the shape or scale parameter  $a = 1, 2, \dots, 100$ . Note that  $\pi(x - c; -a) = 1 - \pi(x - c; a)$ , hence;  $\pi(x - c; a)$  with  $a < 0$  approximates  $I\{x < c\}$ . Either  $\pi(x - c; a)$  or  $\pi(x - c; -a)$  can be used to represent a split in the recursive partitioning setting. Therefore, without loss of generality (WLOG), our consideration is restricted to  $a > 0$ . As  $a$  increases,  $\pi(x)$  provides a better approximation of the indicator function  $I\{x \geq 0\}$ . Similarly,  $\pi(x - c; a)$  approximates  $I\{x \geq c\}$ . On the other hand,  $\pi(x)$  becomes linear for small  $a$ . Because tree methods model data with threshold effects, we are primarily interested in seeking an approximation to the indicator function, where a relatively large  $a$  is desirable; hence, leaving  $a$  as a free parameter for estimation is not advisable. To fix  $a$ , we first note that the spread or scale of the predictor  $X$  on which the best cutoff is sought would affect the choice of  $a$  values. This concern leads

to the specification of  $a$  as  $a := a/s_x$ , where  $s_x$  denotes the sample standard deviation of the predictor. This is equivalent to working with the standardized predictor. To make this more clear, standardization is a necessary first step in SSS splitting. In terms of approximation, note that  $\lim_{a \rightarrow \infty} \pi(x; a) = H(x)$  for any  $x \neq 0$  with exponential rate, i.e.,  $|\pi(x; a) - H(x)| = O\{\exp(-a)\}$ . Thus, a reasonably large constant  $a$  would provide a good approximation. Another possibility is to have  $a$  adaptive of the sample size  $n$  such that  $a \equiv a_n$  satisfies  $\lim_{n \rightarrow \infty} a_n = \infty$ . Many rates can be used for  $a_n$  because of the exponential convergence rate. The choice of  $a$  will be further explored and discussed in the subsequent sections.



**Figure 2.** Plot of the logistic function  $\pi\{x; a\}$  for different values of  $a = 1, 2, \dots, 100$ . The dark line corresponds to the Heaviside step function  $H(x) = I\{x \leq 0\}$ .

### 2.2. Estimating the Best Cutoff Point

To continue, a natural approach is to replace  $\delta(x_i; c)$  in (1) with a sigmoid function. The model then becomes

$$y_i = \beta_0 + \beta_1 s(c; x_i) + \varepsilon_i, \tag{3}$$

where  $s(c; x_i) = \pi(x_i - c; a)$ . Note that we have suppressed the parameter  $a$  in the notation, as it is fixed a priori. By choosing a large  $a$  that corresponds to a sharper ‘S’ shape and estimating the  $c$  corresponding to the best fit, we are essentially finding a sharp step function-type change in the data. Model (3) is a nonlinear parametric model involving four parameters  $\{\beta_0, \beta_1, c, \sigma^2\}$ . Its estimation and related inference can be accomplished via nonlinear least squares [21]. In fact, the related optimization is a separable least squares problem (Section 9.4.1, [22]).

Estimating Model (3) involves multivariate optimization. However, we only need to estimate  $c$  for splitting purposes. This motivates us to approximate the splitting statistic directly and reduce the problem to a one-dimensional optimization with decision variable  $c$

only. We first note that the splitting statistic  $\Delta_l$  used in CART can be treated as an objective function for  $c$  and rewritten as follows:

$$\begin{aligned} \Delta_l(c) &= \sum_{i=1}^n (y_i - \bar{y})^2 - \left\{ \sum_{i \in t_L} (y_i - \bar{y}_L)^2 + \sum_{i \in t_R} (y_i - \bar{y}_R)^2 \right\} \\ &\hat{=} \frac{1}{n_L} \cdot \left( \sum_{i \in t_L} y_i \right)^2 + \frac{1}{n_R} \cdot \left( \sum_{i \in t_R} y_i \right)^2 \\ &= \frac{1}{n_L} \cdot \left( \sum_{i \in t_L} y_i \right)^2 + \frac{1}{n - n_L} \cdot \left( \sum_{i=1}^n y_i - \sum_{i \in t_L} y_i \right)^2, \end{aligned} \tag{4}$$

where  $\{n_L, \bar{y}_L\}$  denote the sample size and the average response in the left child node, respectively; similarly,  $\{n_R, \bar{y}_R\}$  for the right child node. Throughout this article, we use the notation  $\hat{=}$  to indicate ‘equivalence or correspondence up to some irrelevant constant’. In particular, we have ignored irrelevant terms that do not involve  $c$  in rewriting  $\Delta_l(c)$  in (4).

To approximate  $\Delta_l(c)$  in (4), it is only necessary to approximate  $n_L$  and  $\sum_{i \in t_L} y_i$  by replacing  $\delta(x_i; c)$  with  $s(c; x_i)$  such that

$$n_L = \sum_{i=1}^n \delta(x_i; c) \approx \sum_{i=1}^n s(c; x_i)$$

and

$$\sum_{i \in t_L} y_i = \sum_{i=1}^n y_i \delta(x_i; c) \approx \sum_{i=1}^n y_i s(c; x_i).$$

The approximated objective function, denoted as  $\tilde{\Delta}_l(c)$ , becomes

$$\tilde{\Delta}_l(c) = \frac{\mathbf{s}^T (\mathbf{y}\mathbf{y}^T) \mathbf{s}}{\mathbf{j}^T \mathbf{s}} + \frac{(\mathbf{j} - \mathbf{s})^T (\mathbf{y}\mathbf{y}^T) (\mathbf{j} - \mathbf{s})}{\mathbf{j}^T (\mathbf{j} - \mathbf{s})}, \tag{5}$$

where  $\mathbf{y} = (y_i)$ ,  $\mathbf{s} = \{s_i\}$  with  $s_i = (s(c; x_i))$ , and  $\mathbf{j} = (1)$  are  $n$ -dimensional vectors.

Suppose further that, WLOG, the response has been centered  $\mathbf{y} := (\mathbf{I}_n - \mathbf{j}\mathbf{j}^T/n)\mathbf{y}$  such that  $\sum_{i=1}^n y_i = 0$ . It follows that  $\sum_{i \in t_L} y_i = -\sum_{i \in t_R} y_i$ ; hence,  $\Delta_l(c)$  can be further simplified as  $(\sum_{i \in t_L} y_i)^2 / \{n_L(n - n_L)\}$  up to some irrelevant constant. Its approximation  $\tilde{\Delta}_l(c)$  reduces to

$$\tilde{\Delta}_l(c) = \frac{\mathbf{s}^T (\mathbf{y}\mathbf{y}^T) \mathbf{s}}{\mathbf{s}^T (\mathbf{j}\mathbf{j}^T) (\mathbf{j} - \mathbf{s})}. \tag{6}$$

The objective Function (5) or (6) involves a scale parameter  $a$ . Whereas leaving  $a$  for estimation is important in HME [17] and soft trees [18], it is not a good idea for our purpose. If  $a$  is treated as a free parameter, a small  $a$  estimate that corresponds to a nearly linear relationship may provide a better fit, but may not be well suited for identifying the best cutoff point in ordinary tree procedures. This is because tree models account for the relationships between response and predictors via hard threshold functions. Second, if the best cutoff point is estimated for each predictor with a different scale parameter  $a$ , comparison across predictors becomes groundless. Specific recommendations on the choice of  $a$  will be provided in the subsequent sections. As discussed earlier, in order to fix  $a$  it is crucial to standardize the predictor  $x_{ij} := (x_{ij} - \bar{x}_j) / \hat{\sigma}_j$ , where  $(\bar{x}_j, \hat{\sigma}_j)$  denote the sample mean and standard deviation of variable  $X_j$ , respectively. Alternatively, the range of the data may be normalized into the unit interval  $[0, 1]$  by setting  $x_{ij} := (x_{ij} - x_{(1)j}) / (x_{(n)j} - x_{(1)j})$ , where  $x_{(1)j}$  and  $x_{(n)j}$  are the minimum and maximum of  $X_j$ , respectively.

With fixed  $a$ , the best cutoff point  $\hat{c}$  can be obtained by maximizing  $\tilde{\Delta}_l(c)$  in (5) or (6) with respect to  $c$ , then transformed it back to the original data scale for interpretability. This is a one-dimensional optimization problem. It can be conveniently solved by many

standard algorithms. In this paper, we use the method in [23] as implemented in the R [24] function `optimize`. Brent's derivative-free method is very popular because it combines the bisection method, the secant method, and inverse quadratic interpolation by taking advantage of their best aspects.

One issue concerns non-concavity. The objective function in (5) or (6), although smooth, is not concave, as demonstrated in Figure 1. As a result, the algorithm may become stuck at a local optimum. In this case, global optimization procedures (see, e.g., [25,26]) can be helpful. Again thanks to the one-dimensional nature of the problem, one simple solution is to divide the search range into several intervals, find the maximum within each, and then identify the highest maximum among them. We have included this step as an option in our implementation, which may occasionally improve performance yet slow down the splitting procedure. This option is not used in any of the results reported throughout this paper. As shown in our numerical studies later, Brent's method seems quite practically effective in locating the true cutpoint.

An immediate advantage of SSS over GS is computational efficiency. The following proposition provides an asymptotic quantification of the computational complexity involved in both GS and SSS splitting.

**Proposition 1.** *Consider a typical dataset of size  $n$  in the regression tree setting. Both GS and SSS are used to find the best cutoff point of a continuous predictor  $X$  which has  $O(n)$  distinct values. In terms of computation complexity, GS is at best  $O\{\ln(n)n\}$  with the updating scheme and  $O(n^2)$  without the updating scheme, while SSS is  $O(mn)$ , where  $m$  is the number of iterations in Brent's method.*

The proof of Proposition 1 is provided in Appendix B. It is a common wrong impression that LS splitting with updating is only  $O(n)$ . This is because updating entails sorting the response values according to the  $X$  values. It turns out that this step would dominate the algorithm asymptotically in terms of complexity. Comparatively, SSS depends on the number of iterations in Brent's method, denoted as  $m$ . Although the number of iterations is affected by the convergence criterion and the desired accuracy,  $m$  is generally small, as Brent's method has guaranteed convergence at a superlinear rate. Based on our limited numerical experience,  $m$  rarely reaches over 12 even for large  $n$ . In other words, the  $O(mn)$  rate for SSS essentially amounts to the linear rate  $O(n)$ . Therefore, SSS can be more computationally efficient in least squares splitting. Additionally, SSS can be even more advantageous in other recursive partitioning scenarios where an updating formula is not readily available, especially when evaluating each individual split point involves iterative model estimation procedures.

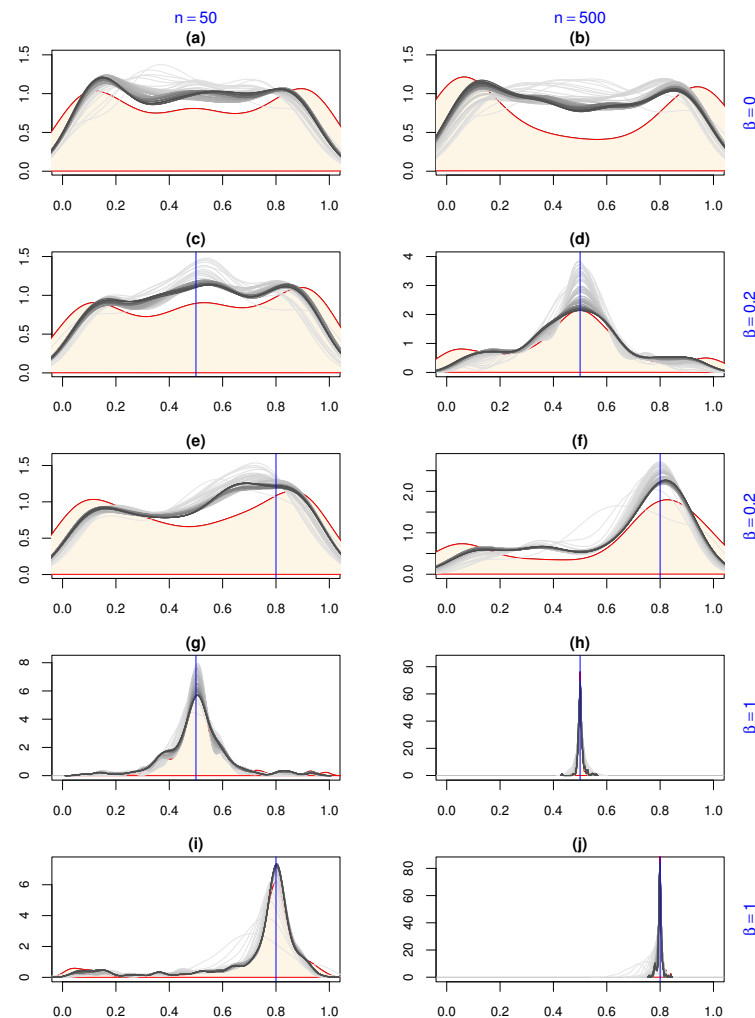
To speed up GS, subsampling or splitting on percentiles are two common strategies. Clearly, subsampling can be combined with SSS as well. Nevertheless, reduced samples could weaken performance and leave weaker signals undetectable. For percentile splitting, the number  $g$  of percentiles used is an issue. In a single tree analysis, the investigator is keenly interested in the 'best' cutoff point for the sake of interpretability and relevance to the scientific or business question at hand. Using percentiles reduces the fineness level to 1% at best in identification of the best cutpoint, which is undesirable, especially when dealing with big data. Moreover, computing all  $g$  percentiles of  $X$  is not computationally thrifty. In terms of complexity, this step is only  $O(gn)$ , and  $g$  could be much larger than  $m$ , i.e., the number of iterative steps in SSS.

### 2.3. Simulation Studies on Single Splits

To evaluate SSS and investigate several relevant issues, we simulated data from the following model:

$$\text{Model A: } y = 1 + \beta \cdot I\{x \leq c_0\} + \varepsilon \text{ with } \varepsilon \stackrel{IID}{\sim} \mathcal{N}(0, 1), \quad (7)$$

where the predictor  $X$  is generated from a uniform  $[0, 1]$  distribution. Two sample sizes  $n \in \{50, 500\}$  were investigated, as recursive partitioning always involves both small sample and large sample problems. We considered two true cutoff points  $c_0 \in \{0.50, 0.80\}$ , corresponding to the balanced and unbalanced scenarios, and three values for  $\beta \in \{0.0, 0.2, 1.0\}$ , corresponding to null, weak, and strong signals, respectively. These considerations form ten combinations in total, as the choice of  $c_0$  no longer matters in the null case of  $\beta_1 = 0$ . For each model configuration, 200 simulation runs were used. For each simulated dataset, both GS and SSS were used to identify the best cutoff point  $\hat{c}$ . For SSS, a spread of choices were considered for  $a \in \{1, 2, \dots, 100\}$ . With the identified cutpoints from all simulation runs, we obtained the empirical density curves of  $\hat{c}$ , as presented in Figure 3.



**Figure 3.** Empirical density of estimated cutpoint  $\hat{c}$ : SSS vs. GS (a–j). Ten scenarios were considered by combining two sample sizes  $n \in \{50, 500\}$ , signal strength  $\beta_1 \in \{0, 0.2, 1\}$ , and two true cutoff points  $c_0 \in \{0.5, 0.8\}$  (indicated via vertical blue lines). Each scenario was examined with 200 simulation runs. In each plot, the density curve of  $\hat{c}$  from GS is shaded and outlined in red. The  $a$  value in SSS falls within the range  $\{1, 2, \dots, 100\}$ , corresponding to the curves in grayscale, where darker colors represent larger  $a$  values.

Figure 3a,b corresponds to the null model case. The results of GS show extraordinarily high peaks at both ends. This phenomenon can be explained by the ‘end-cut preference’ (ECP) problem (see Section 11.8 of CART; [1]), which is known for the fact that the least squares criterion tends to favor unbalanced splits, i.e., splits in which one child node has only a very small proportion of observations. Comparatively, the smoothing effect of SSS substantially ameliorates the ECP problem. Although SSS is intended to approximate GS, weighting introduced by the SSS smoothing process mitigates the ECP problem markedly

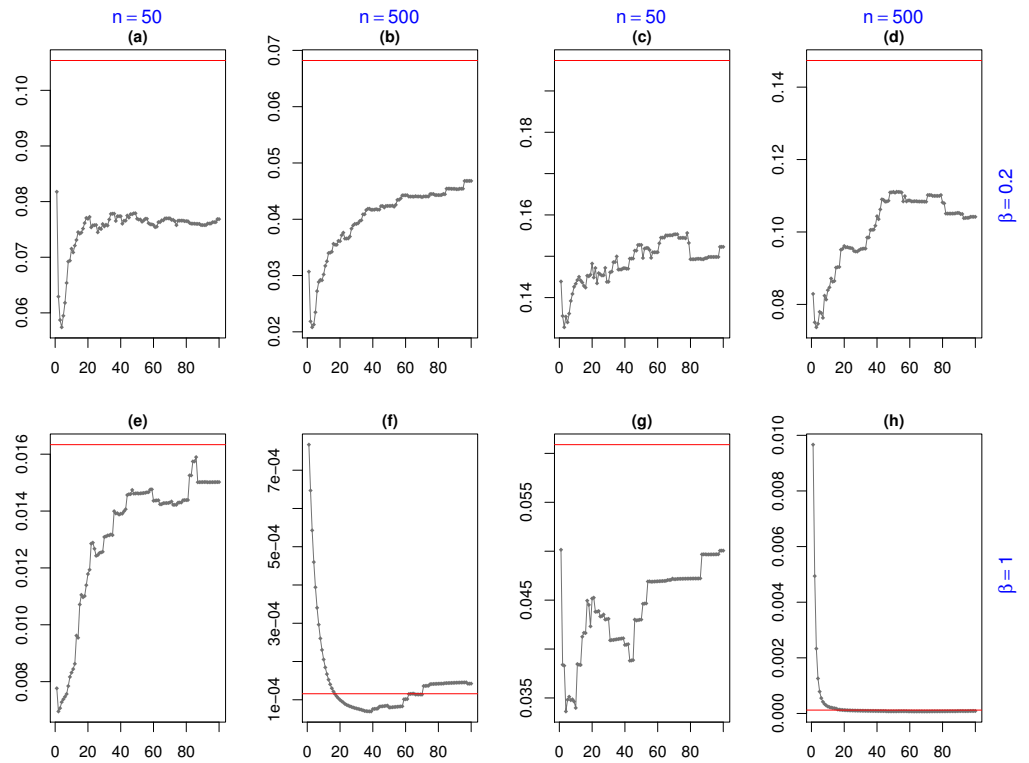


by changing the summations, e.g.,  $\sum_{i \in t_L} y_i$ , into weighted sums  $\sum_{i=1}^n s_i y_i$ . A smaller  $a$  is associated with more distributed weights, and seems more beneficial for this purpose. It is also of note that  $\text{logistic}(0) = 1/2$  in SSS regardless of the choice of  $a$ , as opposed to 0/1 indicator values in GS. These features make cutting at the extreme ends less likely for SSS. To remedy ECP for least squares splitting in regression trees, Torgo [27] suggested setting a minimum number of observations allowable for either child node, which can be implemented via the `minbucket` argument in the R function `rpart` [28]. A similar strategy can be made available in SSS as well. Because Brent's method asks for a range over which the optimum is sought, it is convenient to define the search interval as  $(q_\gamma, q_{1-\gamma})$ , where  $q_\gamma$  and  $q_{1-\gamma}$  denote the sample  $\gamma$ -th and  $(1 - \gamma)$ -th quantiles of  $X$ , respectively. In our implementation, we have included  $\gamma = 0.02$  as an option.

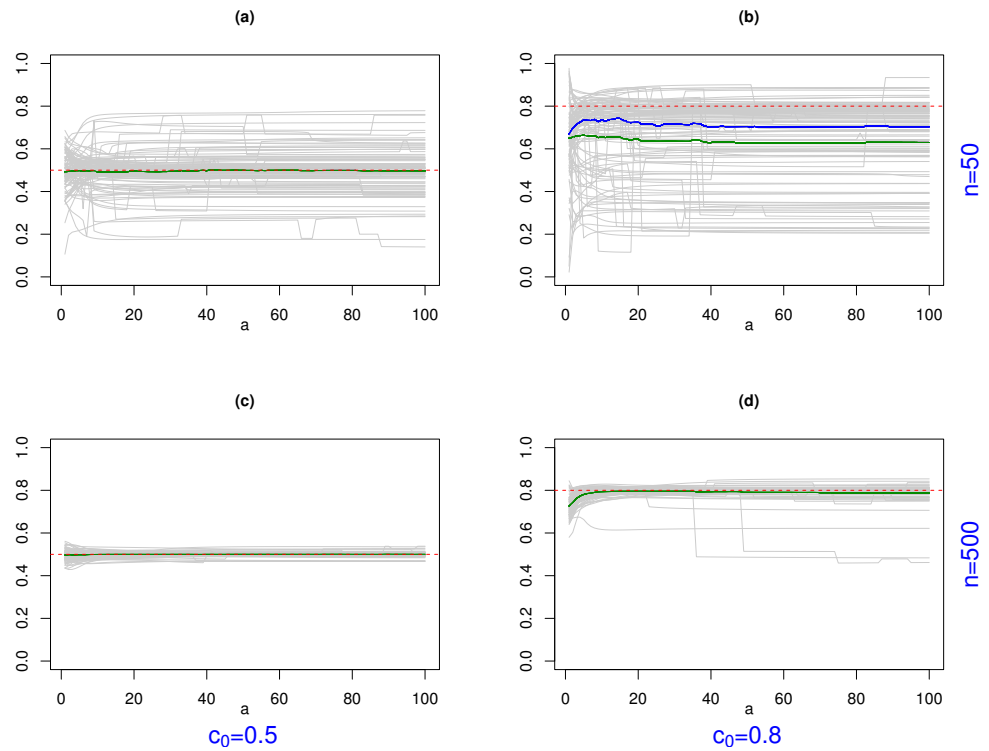
Figure 3c–f corresponds to the weak signal scenarios, where it can be seen that SSS outperforms GS by a great deal in finding the true cutpoint with all choices of  $a$ . The failure of GS in Model B can be partly attributable to the ECP problem. Breiman et al. [1] showed that ECP in LS splitting is a joint result of the Kolmogorov inequality and the law of the iterated logarithm in theory, as further confirmed in a recent article of [29] for other splitting criteria. Ishwaran [29] argued that although ECP is unfavorable for single tree analysis, it is a beneficial feature for random forests, as it maximizes the sample size for a tree to recover from a bad split based on a pure noise predictor. However, we note that the ECP effect can undesirably block out weaker signals, as illustrated here. In addition to the adverse effect of ECP, the splitting statistics computed with GS can be rather turbulent, with considerable variation, especially when the signal grows weak. An illustration from one simulation run is provided in Figure 1. In this scenario, the overall weak signal is likely to be surmounted by some local spikes owing to the large variation. Smoothing can be very helpful here, as in general smoothing preserves the global pattern more than the local ones. Because SSS facilitates a parametric mode of smoothing, in which  $a$  plays the role of a smoothing parameter, it is reasonable to expect that SSS helps to remove irregular local fluctuations in order to better isolate the weak signal with a more stable objective function, especially with a smaller  $a$ . A relatively smaller  $a$  is even preferable in recovering weak signals, as a smaller  $a$  helps to smooth out more local erratic patterns. Figure 3g–j illustrates the strong signal scenarios, in which both GS and SSS perform exceptionally well.

We also computed the mean squared error  $\text{MSE} = \sum_{i=1}^{200} (\hat{c}_i - c_0)^2 / 200$  as a performance measure for each non-null model configuration. The results are presented in Figure 4. Figure 4a–d depicts the weak signal scenarios and reinforces our earlier conclusion that SSS is more advantageous than GS in these contexts. The figure also reveals a pattern where MSE decreases as  $a$  decreases for a certain range of  $a$  values, say,  $a \geq 5$ . However, excessively small  $a$  values result in poor approximations, leading to diminished performance. This indicates a tradeoff between approximation and smoothness in terms of the choice of  $a$ . Figure 4e–h represents the strong signal scenarios, where both GS and SSS perform very well. Notably, SSS outperforms GS in the small sample scenario with  $n = 50$  across all choices of  $a$  considered. In contrast, with larger samples  $n = 500$  the results are mixed, and the difference between SSS and GS becomes negligible for reasonably large  $a$  values (e.g.,  $a \geq 10$ ). Additionally, both GS and SSS demonstrate better performance with the balanced cutpoint  $c_0 = 0.5$  compared to the unbalanced cutpoint  $c_0 = 0.8$ , as illustrated by the MSE scales in Figure 4.

To further investigate the sensitivity with respect to  $a$ , Figure 5 plots  $\hat{c}$  versus  $a$  in each simulation run for 100 simulation runs with  $\beta = 1$ . It can be seen that the lines are essentially flat, except for very small  $a$  values. This indicates that the performance of SSS remains fairly stable with respect to  $a$ , particularly for reasonably large values. Similar patterns are observed in the weaker signal case ( $\beta = 0.2$ ), although these results are not presented here. This suggests that while the scale parameter  $a$  functions as a smoothing parameter and further research on its fine-tuning may be valuable, setting  $a$  as a constant seems to be a sensible approach.



**Figure 4.** MSE of the estimated cutpoint  $\hat{c}$  vs.  $a$  in SSS. In Panels (a,b,e,f), the true cutpoint is  $c_0 = 0.5$ , while in Panels (c,d,g,h) it is  $c_0 = 0.8$ . The solid red horizontal line corresponds to the MSE value from GS.



**Figure 5.** Plot of estimated cutoff point  $\hat{c}$  versus  $a = 1, 2, \dots, 100$  for 100 simulation runs (a–d). Data were generated from Model A with  $\beta = 1$ . Each gray line corresponds to one simulation run. Superimposed on each plot are the true cutoff point (dotted red line) and mean cutoff at each  $a$  (solid green line). In Panel (b), the median cutoff is also added (solid blue line). Four scenarios were considered by combining two sample sizes  $n \in \{50, 500\}$  and two true cutoff points  $c_0 \in \{0.5, 0.8\}$ .

Additionally, the averaged best cutoff point  $\hat{c}$  at each  $a$  overlaps well with the true cutoff point  $c_0$ , except in Panel (b), where the estimation is biased downwards. GS exhibits a similar downward bias; see panel (i) in Figure 3. Figure 5b highlights the increased difficulty in estimating an unbalanced cutoff point, especially with a small sample size. On the other hand, the notable difference between the mean  $\hat{c}$  values and  $c_0$  in Panel (b) can be partially attributed to the nonrobustness of means. If we consider the median  $\hat{c}$  values (represented by the blue line), they align much more closely with the true value  $c_0$ . As the sample size increases, the bias becomes negligible, as shown in panel (d).

Another alternative for specifying  $a$  is to make  $a$  adaptive to the sample size  $n$ . To investigate this, we reran the experiments by considering sample sizes while varying in  $\{50, 100, 150, \dots, 1000\}$ . Two strategies for setting  $a$  values were tried: constant  $a$  in  $\{10, 30, 50\}$ , and adaptive choices of  $a$  in  $\{n, \sqrt{n}, \ln(n)\}$ . A total of 200 simulation runs was taken for each scenario and the MSE values were recorded. Because the scale for MSE drops dramatically as  $n$  increases, we computed the relative difference in MSE for the selected cutpoints from SSS as opposed to GS; thus, a negative value indicates improvement upon GS. Figure 6 plots the results versus  $n$  for each choice of  $a$ . It can be seen that SSS constantly outperforms GS with weaker signals ( $\beta = 0.2$ ) and that a smaller  $a$  seems preferable. In the stronger signal ( $\beta = 1.0$ ) scenario, the constant choices  $a \in \{30, 50\}$  work very well, while the adaptive choice  $a = \sqrt{n}$  outperforms the other two.

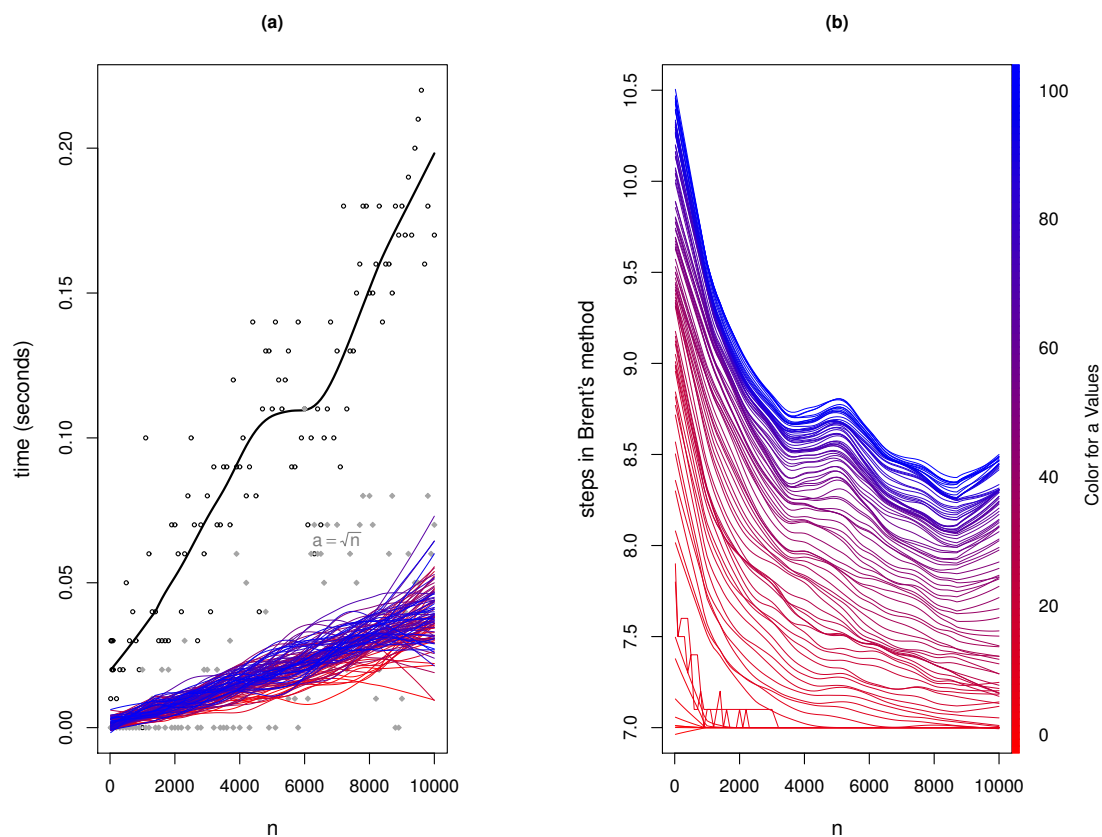


**Figure 6.** Relative difference in MSE of SSS versus GS for different sample sizes  $n \in \{50, 100, 150, \dots, 1000\}$ . A few constant  $a$  choices in  $\{10, 30, 50\}$  are considered for SSS, together with  $n$ -adaptive choices  $\{n, \sqrt{n}, \ln(n)\}$ . Negative values indicate advantages over GS.

In summary, a larger  $a$  yields a better approximation of the threshold effect, while a smaller  $a$  offers increased smoothness. With a larger  $a$ , the performance of SSS is relatively more stable, as shown in Figures 3–5. Conversely, a smaller  $a$  can be beneficial for detecting weak signals and addressing the end-cut preference (ECP) problem. Our empirical results demonstrate that the performance of SSS remains stable as  $a$  varies, as well as that SSS

outperforms GS in most scenarios with a reasonable choice of  $a$ . While some optimization of  $a$  could be explored, we believe that a reasonable choice of  $a$  will suffice for practical applications. Based on our findings, we tentatively recommend fixing  $a$  in the range of  $[20, 60]$ , while the adaptive choice of  $a = \sqrt{n}$  also seems advisable.

To investigate computational efficiency, we generated data from Model A in (7) with  $\beta = 1$  and  $n \in \{20, 30, \dots, 100, 200, 300, \dots, 10,000\}$ . Because  $x \sim \text{unif}(0, 1)$ , the number of distinct values that it has increases with  $n$ . We considered different constant choices of  $a \in \{1, 2, \dots, 100\}$  and an adaptive choice  $a = \sqrt{n}$ . For greedy search, we used the R function `rpart` [28] by setting its options `rpart.control` such that it produces one single split only and all other features, such as competitive and surrogate splits, are suppressed. The implementation of SSS was accomplished solely in R with the `optimize` function. Figure 7a plots the CPU time (in seconds) that SSS and GS spent on ten simulation runs. It can be seen that SSS consistently outperforms GS. Figure 7b plots the averaged number  $m$  of iterative steps in Brent’s method for SSS. It can be seen that  $m$  ranges from 7 to 10 in this example. In addition, a smaller  $a$ , corresponding to a smoother objective function, leads to a smaller number of iterative steps, as expected. In practice, similar experiments can be performed on specific datasets to determine the approximate threshold at which SSS becomes more advantageous than GS.



**Figure 7.** Computing time comparison between GS and SSS. The sample size  $n$  varies in  $\{20, 30, \dots, 100, 200, 300, \dots, 10,000\}$ . The total CPU time (in seconds) for ten simulation runs in each setting was recorded for both GS and SSS. For SSS, the number  $m$  of iterative steps in Brent’s optimization algorithm was also obtained and averaged over ten simulation runs. Panel (a) plots the CPU computing time versus the sample size  $n$ . The computing times from GS are plotted with black circles and superimposed with a smooth curve from lowess. The curves from SSS with  $a \in \{1, 2, \dots, 100\}$  are plotted. In addition, an adaptive choice  $a = \sqrt{n}$  is also included and its associated computing times are plotted with gray diamonds. In Panel (b), the averaged number of iterative steps involved in Brent’s method are plotted vs.  $n$ .

### 2.4. Variable Selection Bias

The variable selection bias problem is deemed inherent in recursive partitioning. By optimally selecting a goodness-of-split measure across all permissible splits from all variables, greedy search gives preference to variables that have more distinct levels or values. As a remedy, Loh [9] (GUIDE) proposed first selecting the most important splitting variable and then finding its best cutoff point via greedy search. In addition to its negligible variable selection bias, the method is advantageous in reducing the computational cost. A similar idea was followed in [30] (MOB). However, determination of the most important splitting variable is an equally if not more difficult problem. A predictor that is important in its threshold effect may not be so in other senses. In GUIDE, residuals from linear models are first grouped. Every predictor is treated or otherwise made categorical to form two-way contingency tables with residual-based grouping. The most important variable is then selected based on Pearson  $\chi^2$  tests. In MOB, the selection is based on a stability assessment of the estimated coefficients in a linear model. One potential problem with both methods is that the variable selected via linear models may not be the one with the most important threshold effect for the partitioning purpose. Furthermore, these methods lose power in detecting the true threshold signal on continuous or nominal predictors, leading to a different type of bias in variable selection.

In other approaches, the best cutoff point is associated with a maximally selected random variable. Akin to change-point problems (see, e.g., [31,32]), asymptotic inference on maximally selected statistics resorts to standard results in stochastic processes (see, e.g., [10]), while their exact distributions can be determined with permutation-based methods (see, e.g., [12], 2008). Shih and Tsai [11] evaluated these approaches in partitioning data and demonstrated their potential in reducing variable selection bias via simulations. In these approaches, the  $p$ -value associated with the best cutpoint for each predictor is first computed and possibly transformed. These  $p$ -values are then compared across predictors to determine the best split of the data. Different methods might be used to obtain the  $p$ -value for the best cutpoint of each predictor depending on the type of the predictor. Nevertheless, the associated distributions for maximally selected statistics may not be easily available, and the computation could be intensive when either the sample size  $n$  or the number of distinct levels  $K$  become large. For example, the permutation-based methods are clearly not a good choice for this purpose, as the exact  $p$ -value is often computationally infeasible while the approximate  $p$ -value does not have the necessary precision in order to compare with  $p$ -values obtained from the Brownian bridge for example. Moreover, most of the above-mentioned works on the distribution of maximally selected statistics are applicable or feasible only to continuous or at least ordinal predictors, and not to nominal variables in the strict sense.

The SSS method can help to get around the variable selection bias problem by providing a simple alternative way of evaluating splits. The significance assessment of the best cutoff point on a continuous predictor can be made via its connection to a nonlinear parametric model (3). While other testing procedures are available, it is convenient to use the likelihood ratio test for  $H_0 : \beta_2 = 0$  in Model (3). Computing the LRT entails estimation of Model (3). This is a separable nonlinear least squares problem that can be efficiently solved by the variable projection method of Golub and Pereyra [33]. The main idea is that  $\beta$  can be readily estimated given  $c$ . We define matrix  $\mathbf{S} = [\mathbf{j}, \mathbf{s}] \in \mathbb{R}^{n \times 2}$ . The nonlinear least square criterion is  $\| \mathbf{y} - \mathbf{S}\beta \|^2_2$  with  $\beta = (\beta_0, \beta_1)^T$ . For a given  $c$ , the estimate of  $\beta$  can be obtained via ordinary LS:  $\hat{\beta} = (\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T \mathbf{y}$ . Plugging it into the LS criterion leads to an objective function for  $c$  only:

$$\| \mathbf{y} - \hat{\mathbf{S}}\beta \|^2 = \| \mathbf{y} - \mathbf{S}(\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T \mathbf{y} \|^2 = \| \mathbf{P}_{\mathcal{C}^\perp(\mathbf{S})} \mathbf{y} \|^2 \tag{8}$$

where  $\mathcal{C}(\mathbf{S})$  denotes the column subspace of matrix  $\mathbf{S}$ ,  $\mathcal{C}^\perp(\mathbf{S})$  is the subspace perpendicular to  $\mathcal{C}(\mathbf{S})$ , matrix  $\mathbf{P}_{\mathcal{C}^\perp(\mathbf{S})} = \mathbf{I}_n - \mathbf{S}(\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T$  denotes the orthogonal projector to  $\mathcal{C}^\perp(\mathbf{S})$ , and  $\| \cdot \|$  denotes the Euclidean norm unless otherwise specified. Estimates of  $c$  can be obtained

by minimizing (8), which is equivalent to maximizing  $\tilde{\Delta}_l(c)$  in (5). A popular implementation is to combine the Gauss–Newton method with variable projection, as considered by Kaufman [34] and summarized in Osborne [35]. The algorithm is iterative with an approximately quadratic convergence rate. The corresponding pseudocode for estimating Model (3) is provided in Appendix A.

In our approach, the best cutoff  $\hat{c}$  is efficiently obtained by optimizing (8) or the approximated splitting statistic directly. Given  $\hat{c}$ , Model (3) becomes linear and  $\hat{\beta}$  can be obtained via OLS. The LRT statistic (see, e.g., Section 12.4 of [21], 2003) is provided by

$$\text{LRT} = n \log \frac{\mathbf{y}^T (\mathbf{I}_n - \mathbf{j}\mathbf{j}^T/n)\mathbf{y}}{\|\mathbf{y} - \mathbf{S}(\hat{c})\hat{\beta}\|^2},$$

where  $\mathbf{S}(\hat{c})$  denotes the  $\mathbf{S}$  matrix evaluated at  $\hat{c}$ . This test compares Model 3 with the null model  $y_i = \beta_0 + \varepsilon_i$ . Because  $a$  is fixed, the resulting LRT statistic follows a  $\chi^2(2)$  distribution under  $H_0$ . Alternatively, an  $F$  test can be used (see Section 5 of Seber and Wild [21]). This result suggests a conjecture that there may be a connection between the  $\chi^2(2)$  distribution and the asymptotic distribution derived from the Brownian bridge [10], which could be further investigated in future research. We define the associated logworth

$$\text{logworth} = -\log_{10} \Pr\{\chi^2(2) \geq \text{LRT}\}$$

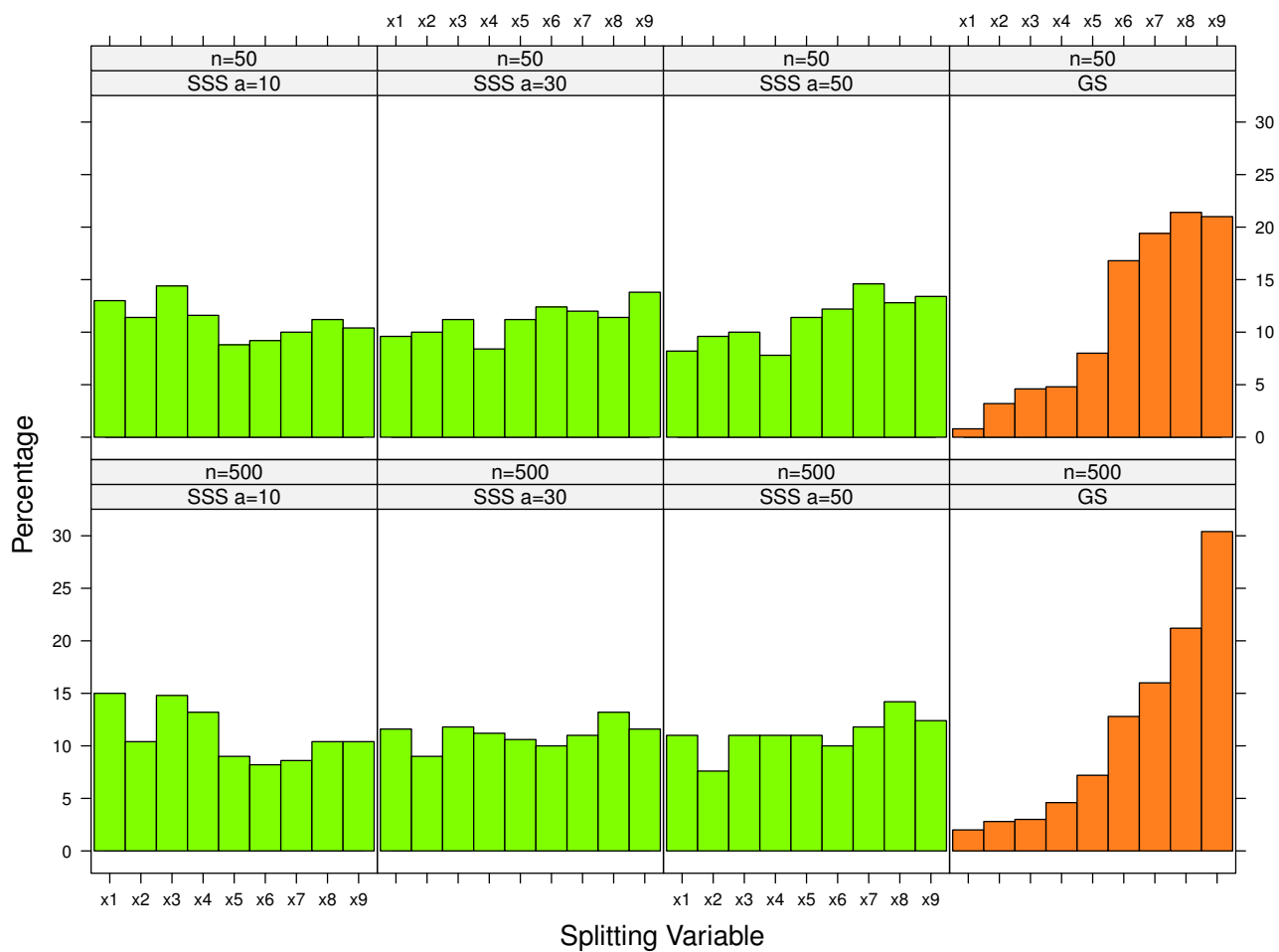
as the minus base-10 logarithm of the resulting  $p$ -value when referring to a  $\chi^2(2)$  distribution. The logworth provides a coalescent measure of the strength for the best cutoff point of each predictor regardless of the variable type or the methods used to evaluate its best cutpoint. The best split of the data is then the one with the maximum logworth among all predictors.

We next discuss how to handle other types of predictors in combination with SSS. For a 0–1 binary variable, the LRT for  $H_0 : \beta_1 = 0$  in Model (1) with  $c = 0.5$  can be used to evaluate the single split. The corresponding  $p$ -value, together with the associated logworth, can be computed by referring to the  $\chi^2(1)$  distribution. For an ordinal variable with  $K$  distinct values, we proceed with SSS by treating as if it is continuous. However, when  $3 \leq K \leq 10$  is small, it is possible to resort to the maximally selected statistics approaches. Several approaches are available for computing the associated  $p$ -value. A method that combines the change-point approach of Hawkins [36] and Genz [37], as implemented in the R package `maxstat` [38], is particularly appealing for the small  $K$  scenario in terms of computational feasibility and efficiency.

For a nominal variable  $X$  with  $K$  distinct levels, we first ‘ordinalize’ its levels by sorting its group means and then proceed by treating it as if it were continuous. This standard strategy in GS greatly reduces the total number of splits to be evaluated, and is theoretically justified (see CART, Section 9.4, [1]); however, both its steps introduce overoptimism or bias in terms of the association between  $Y$  and  $X$ . The ordinalization step may make an irrelevant  $X$  artificially correlated with  $Y$ , while the maximal selection of splitting criteria over the ordinalized levels of  $X$  induces additional overoptimism. Clearly, SSS can be used in the latter step to find the best cutpoint and help avoid bias owing to maximal selection. However, the overoptimism stemming from the ordinalization step remains an issue. In general, how to splitting nominal variables in regression trees without bias is not satisfactorily addressed in the extant literature, and further research is warranted.

To demonstrate this, we simulate data from a null model where the response  $Y$  has nothing to do with any of the nine either predictors  $\{X_1, X_2, \dots, X_9\}$ . For the above discussed reasons, we have restricted ourselves to ordinal or continuous predictors only. The numbers of distinct values for  $\{X_1, X_2, \dots, X_9\}$  were set as  $\{2, 3, 4, 5, 10, 20, 50, 100, 500\}$ , respectively. Two sample sizes  $n = 50$  and  $n = 500$  were considered. Figure 8 plots the frequency of each variable being selected for splitting data by SSS (combined with maximally selected statistics) and GS. For SSS, we tried different choices of  $a$  in  $\{10, 30, \text{ and } 50\}$ . It can be seen that greedy search suffers severely

from selection bias, while SSS substantially alleviates the problem. Different choices of  $a$  for SSS yield very similar results, with a slight preference to smaller  $a$  values.

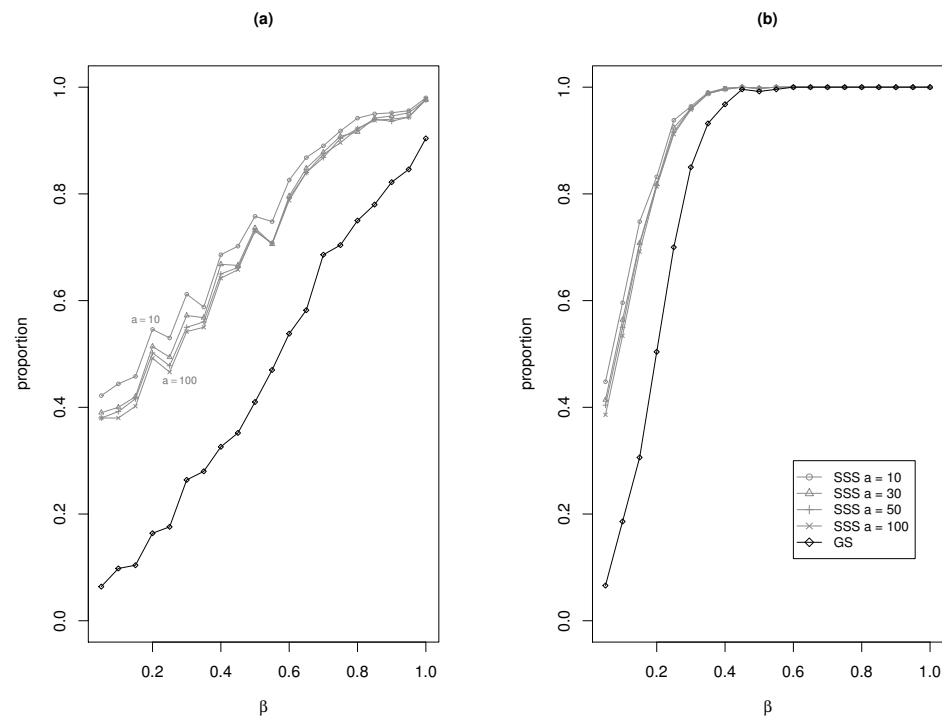


**Figure 8.** Bar plot of frequencies (from 1000 simulation runs) of splitting variables selected by SSS vs. greedy search. The data were generated from a null model. Nine variables  $\{X_1, X_2, \dots, X_9\}$ , with the possible number of distinct values equal to  $\{2, 3, 4, 5, 10, 20, 50, 100, 500\}$ , respectively, were included in each dataset. Two sample sizes  $n = 50$  and  $n = 500$  were considered. For SSS, three choices of  $a \in \{10, 30, 50\}$  were tried.

For further exploration, we consider another trial where data were generated from

$$\text{Model B: } y = 1 + \beta \cdot I\{x_1 \leq 0.5\} + \varepsilon \text{ with } \varepsilon \stackrel{IID}{\sim} \mathcal{N}(0, 1), \tag{9}$$

where  $X_1$  is binary 0–1 valued and we include a second variable  $X_2 \sim \mathcal{N}(0, 1)$ . In this model,  $Y$  is only related to the binary predictor  $X_1$ ; nevertheless, the unrelated predictor  $X_2$  has distinct values of  $K \approx O(n)$ . We then allowed the coefficient  $\beta$  to vary from  $\{0.01, 0.02, \dots, 1\}$ . We recorded the percentage of correct selection of the true split  $X_1 \leq 0.5$  made by GS and SSS. For SSS, we also tried different choices of  $a \in \{10, 30, 50, 100\}$ . Figure 9 plots the results against  $\beta$  for two sample sizes,  $n = 50$  and  $n = 500$ . SSS clearly outperforms GS in picking up the correct split. For weak signals, it is more likely for GS to have spurious splits introduced into the tree structure. As expected, both perform better as the signal grows stronger or the sample size grows larger. For SSS, a smaller  $a$  tends to yield a higher percentage of correct selection, yet with seemingly unimportant differences.



**Figure 9.** Percentages of correct selection (out of 500 simulation runs) by GS vs. SSS: (a)  $n = 50$  and (b)  $n = 500$ . The data were generated from Model B in (9).

In the above simulation, we included only one binary predictor in order to better isolate the specific challenges of variable selection bias. As one reviewer noted, the effects of multiple predictors on the response can complicate the delineation of a variable’s impact. While future research with more extensive simulations is necessary to address this issue, we would like to emphasize that the approach used in recursive partitioning is somewhat analogous to coordinate descent in optimization, where each variable is handled separately. This allows us to focus on the individual impact of each predictor at each split.

2.5. Tree Performance

Although our primary focus is on one single split of data, it is interesting to see how SSS compares with GS in whole-tree performance as well. To this end, we implemented SSS using the user-written splitting functions [8] in `rpart`. While this may not be a good idea with regard to computing speed, it allows us to utilize many useful features available in `rpart` [28], such as surrogate splits for missing value treatment.

2.5.1. Simulation Studies

We first consider simulation studies. Data were generated from the following models:

$$\text{Model C: } y = 0.5 + 0.5 \cdot x_1 + 0.5 \cdot I\{x_2 \leq 0.5\} + \varepsilon \tag{10}$$

$$\text{Model D: } y = 2 + 2 \cdot x_1 \cdot I\{x_2 \leq 0.5\} + \varepsilon \tag{11}$$

with  $\varepsilon \stackrel{IID}{\sim} \mathcal{N}(0, 1)$ . Models C and D are expected to have four and three terminal nodes and correspond to weaker and stronger signals, respectively. Five predictors were generated, with  $X_1$  being binary 0–1 and  $X_2, \dots, X_5$  from  $\text{unif}(0, 1)$ , where only  $X_1$  and  $X_2$  are related to  $Y$  in either an additive or interaction form.

There are a few approaches available for determining the best tree size. To facilitate the comparison, a unified ‘test sample’ approach (see, e.g., Section 11.4 of CART, [1]) was taken for both SSS and GS. Specifically, we first constructed a large initial tree based a training sample  $\mathcal{L}_1$  of



size  $n$  and pruned it to obtain a nest sequence of subtrees. Then, we generated an independent validation sample  $\mathcal{L}_2$  of the same size  $n$  and applied each subtree in the sequence to predict the responses in  $\mathcal{L}_2$ . The subtree yielding the minimum prediction mean squared error (PMSE) was identified as the ‘best’. Two sample sizes  $n = 100$  and  $n = 500$  were considered in our experiments. For SSS, the scale parameter was fixed at  $a = 50$ .

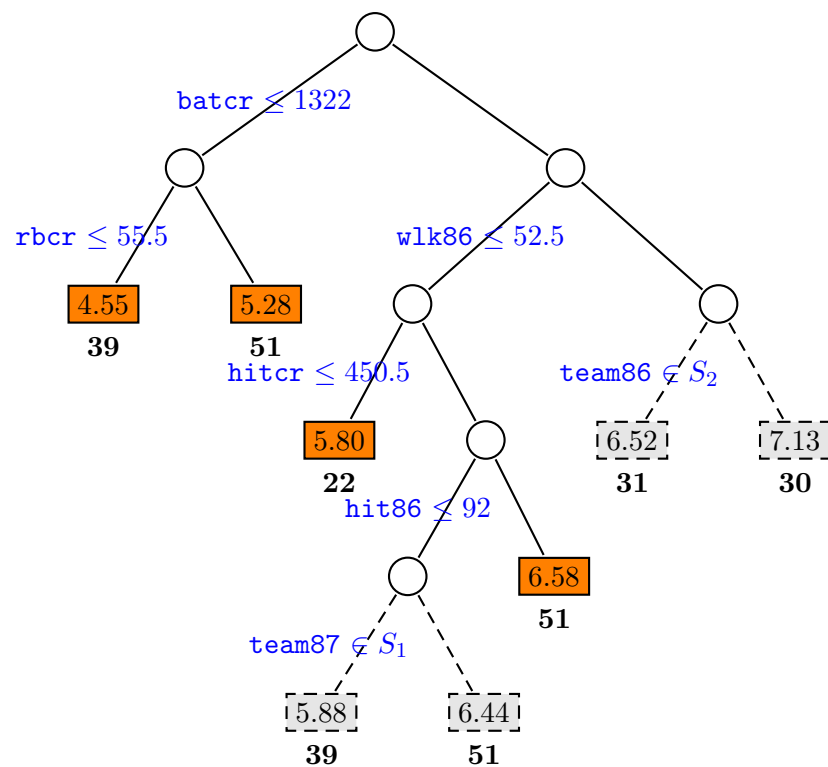
To compare the predictive accuracy, another independent testing sample  $\mathcal{L}_3$  of size  $n' = 5000$  was generated from the same model. In every simulation run, the best tree structure we identified was applied to  $\mathcal{L}_3$  for prediction. The resulting  $PMSE = \sum_{i \in \mathcal{L}_3} (y_i - \hat{y}_i)^2 / n'$  was recorded. A total of 200 simulation runs was carried out, and the averaged PMSE is reported in Table 1. Note that  $\mathcal{L}_3$  remained the same across all simulation runs. Table 1 also reports the ‘best’ tree size averaged over the simulation runs. It can be seen that SSS consistently outperforms GS in detecting weak signals. In the weak signal scenario (Model C), SSS leads to a final tree size that is closer to the true one of 4 and yields a smaller averaged PMSE. SSS is also highly competitive for Model F with stronger signals.

**Table 1.** Comparison of SSS vs. GS for whole tree performance in regression problems. For simulated data, the table reports the final tree size and PMSE, both averaged over 200 simulation runs. For benchmark datasets, the performance measures are based on one single tree procedure. The computing time refers to the duration (in seconds) taken to construct the initial tree using the entire dataset. For simulated data, the reported time is the average from 20 simulation runs.

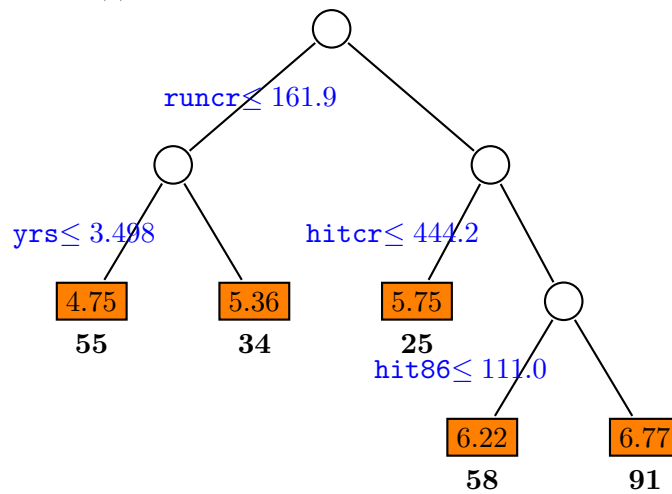
	Data Set	$n$	$p$	SSS			GS		
				Size	PMSE	Time	Size	PMSE	Time
Simulated	Model C	100	5	2.295	1.155	0.0715	1.800	1.163	0.0350
		500	5	4.080	1.036	0.0790	3.850	1.045	0.2365
	Model D	100	5	3.385	1.121	0.0675	3.380	1.136	0.0285
		500	5	3.135	1.060	0.0725	3.190	1.058	0.2315
Benchmark	airfoil	1503	5	35	19.795	0.20	36	19.227	1.14
	auto	398	7	9	23.940	0.12	9	24.606	0.22
	concrete	1030	8	59	58.326	0.27	42	71.689	1.30
	energy: heating	768	10	35	0.978	0.11	33	0.968	0.40
	energy: cooling	768	10	33	5.709	0.17	34	5.790	0.34
	wiki4HE	907	48	9	0.401	1.57	9	0.434	4.22

### 2.5.2. Real Data Examples

For real-world example illustrations, we first considered the 1987 baseball salary data, which have been widely analyzed in the literature. The final CART tree, selected with the 0-SE rule, has eight terminal nodes, as plotted in Figure 10a. Loh [9] commented that the two splits based on `team86` and `team87` (highlighted with dashed lines) are hard to interpret and may be attributable to the selection bias of greedy search. The 1-SE rule yields a final tree of four terminal nodes, which is a complete full tree of depth 3. Figure 10b plots the final SSS tree with  $a = 50$ , which has five terminal nodes. This tree structure was determined following the tree methodology based on the goodness-of-split [7], which combines split-complexity pruning with bootstrap-based bias correction. Of note is that the two categorical variables concerning baseball team membership no longer show up in the final SSS tree. The `yrs` variable is deemed important in the GUIDE tree, which can be confirmed via variable importance ranking in random forests as well. It does not show up in the CART tree, but does in the SSS tree. We also conducted a 10-fold cross-validation to compare their predictive performance. The mean squared prediction error measures for the SSS tree, 0-SE CART tree, and 1-SE CART tree were 0.1867, 0.1194, and 0.1915, respectively. Thus, the final SSS tree is highly competitive in terms of predictive accuracy.



(a) The CART Tree Selected via 0-SE



(b) The Final SSS Tree

**Figure 10.** Analysis of 1987 baseball salary data. In (a),  $S_1$  and  $S_2$  represent specific subsets of baseball teams. Within each terminal node is the mean response (log-transformed salary); underneath is the sample size.

For further illustration, we selected five more benchmark datasets from the Regression category at UCI Machine Learning Repository (<http://archive.ics.uci.edu/ml/>, accessed on 20 September 2024). These were Airfoil Self-Noise (airfoil), Auto MPG (auto), Concrete Compressive Strength (concrete), Energy Efficiency (energy), and Teaching Uses of Wikipedia (wiki4HE). As an “off-the-shelf” tool, tree analysis requires little data preparation. Of brief note, there are two responses in energy, namely, heating load and cooling load, which were analyzed separately. In wiki4HE, we summed up the item scores for the last category (“experience with Wikipedia usage”) and used it as the response variable. All missing values were handled using the surrogate splits approach.

The following paradigm was taken to analyze each dataset. We randomly partitioned the dataset into three parts, the training sample  $\mathcal{L}_1$ , validation sample  $\mathcal{L}_2$ , and testing sample  $\mathcal{L}_3$ , with approximately equal proportions 1:1:1. A large tree was grown and pruned using  $\mathcal{L}_1$  and the best subtree was selected according to the minimum PMSE based on  $\mathcal{L}_2$ . The best tree size and the PMSE measure recomputed from  $\mathcal{L}_3$  are reported in Table 1. It can be seen that SSS generally compares favorably to GS in predictive performance in most cases, although not always. This can be attributed to its superior ability to detect weak signals and its immunity to issues such as end-cut preference and variable selection bias. Despite the potential advantages of SSS over GS in various scenarios, its superiority in prediction accuracy ultimately depends on the specific dataset being analyzed.

### 2.5.3. Computing Time Comparison

To compare computing times, we recorded the duration (in seconds) required to construct the initial large tree using the entire dataset. To ensure a fair comparison, we implemented both SSS and GS with the user-defined splitting function from the R package **rpart** [8]. The maximum tree depth was set to 10 and the minimum node size was 20. All programs were executed on a ThinkPad workstation equipped with a 2.40 GHz CPU and 36 GB of memory. The results are presented in Table 1. For simulated data, the reported times are the average from 20 simulation runs.

Table 1 shows that SSS outperforms GS across all benchmark datasets. However, for simulated data, GS is faster than SSS when  $n = 100$ , although it quickly becomes slower as the sample size increases to  $n = 500$ . It is noteworthy that while the increase in sample size has a minimal impact on the computing time of SSS, it significantly influences the performance of GS. This intriguing result suggests that adaptive use of SSS and GS in practical implementations is advisable depending on factors such as sample size  $n$  and the number  $m$  of distinct values of a covariate. When  $n$  is small or  $m$  is limited, GS tends to be faster than SSS. These scenarios are commonly encountered as the tree continues to grow. Thus, a threshold might be established for  $n$  and  $m$  to determine when to use GS instead of SSS, applying GS when either  $n$  or  $m$  falls below the threshold.

### 3. SSS for Classification Trees

SSS can be extended to classification trees in a similar manner. Consider data  $\{(y_i, \mathbf{x}_i) : i = 1, \dots, n\}$  with binary responses  $y_i \in \{0, 1\}$ . Commonly-used node impurities for classification trees include the misclassification error rate, Gini index, and entropy. The Gini index corresponds to treating binary  $Y$  as if continuous and applying OLS directly with the linear regression model. With the entropy measure, minimizing the within-node impurity is equivalent to maximizing the LRT for testing  $H_0 : \beta_1 = 0$  in a logistic model:

$$\text{logit}(\pi_i) = \log \frac{\pi_i}{1 - \pi_i} = \beta_0 + \beta_1 \delta(x_i; c) \tag{12}$$

where  $\pi_i = \Pr(y_i = 1 | \mathbf{x}_i)$ .

Among the three common measures, the misclassification error rate is not recommended due to its serious defects, as discussed in Section 4.1 of CART [1]. The Gini index and entropy behave similarly. They both work well with strong signals and generalize well to multiclass responses, but suffer from the end-cut preference (ECP) problem. To remedy this, Morgan and Messenger [39] introduced a ‘delta splitting rule’ by replacing the squared Euclidean distance with the absolute deviation in the Gini index; however, this is slightly harder to compute. Breiman et al. [1] considered a ‘twoing rule’ in CART, which was shown to be equivalent to the delta splitting rule.

Just as earlier, replacing the threshold term  $\delta(x_i; c)$  with an SSS term yields a nonlinear logistic model:

$$\text{logit}(\pi_i) = \beta_0 + \beta_1 s(c; x_i). \tag{13}$$

To avoid optimization with multiple parameters, it is preferable to first form an objective function by approximating a legitimate splitting statistic and then optimizing it with respect to the cutpoint  $c$  only. This strategy, along a similar line to Neyman [40], shows considerable yield in terms of computational simplicity. For a node  $t$ , let  $n_1 = \sum_{i \in t} y_i$  denote the number of observations with  $y = 1$ ,  $n_L$  the number of observations in the left child node, and  $n_{L1}$  the number of observations with  $y = 1$  in the left child node. The following proposition expresses the reduction in impurity  $\Delta t$  in classification trees in terms of  $n_L$  and  $n_{L1}$  only.

**Proposition 2.** Let  $\pi_t$  denote the proportion of observations with  $y = 1$  in node  $t$ . Let  $\Delta t$  be the reduction in node impurity when node  $t$  is split into  $t_L$  and  $t_R$ . With entropy impurity measure  $i(t) = -\pi_t \log \pi_t - (1 - \pi_t) \log(1 - \pi_t)$ , we have

$$\begin{aligned} \Delta t \cong & n_{L1} \log n_{L1} + (n_L - n_{L1}) \log(n_L - n_{L1}) + (n_1 - n_{L1}) \log(n_1 - n_{L1}) \\ & + \{(n - n_1) - (n_L - n_{L1})\} \log\{(n - n_1) - (n_L - n_{L1})\} \\ & - n_L \log n_L - (n - n_L) \log(n - n_L) \end{aligned} \tag{14}$$

up to some constant. With Gini index  $i(t) = \pi_t(1 - \pi_t)$ , we have

$$\Delta t \cong - \frac{n_{L1}(n_L - n_{L1})}{n_L} - \frac{(n_1 - n_{L1})\{(n - n_L) - (n_1 - n_{L1})\}}{n - n_L}. \tag{15}$$

Proposition 2 provides the grounds for an efficient way of estimating the best cutoff point  $c$  via SSS. To do so, we need to approximate only  $n_L = \sum_{i=1}^n \delta(x_i; c)$  with  $\tilde{n}_L = \sum_{i=1}^n s_i$  and  $n_{L1} = \sum_{i=1}^n y_i \delta(x_i; c)$  with  $\tilde{n}_{L1} = \sum_{i=1}^n y_i s_i$  in either (14) or (15). The approximated  $\Delta t$  becomes a smooth objective function of  $c$  alone; we denote it as  $\tilde{\Delta}t(c)$ . The best cutoff point  $\hat{c}$  can be found as  $\hat{c} = \arg \max_c \tilde{\Delta}t(c)$ . This is again a one-dimensional optimization problem, and can be efficiently solved with the Brent [23] method.

SSS is flexible with the general idea of approximation and optimization. It is obvious that any other two-sample test statistic, such as Pearson  $\chi^2$ , can be used in this approach as well. An analogous strategy for dealing with variable selection bias, as in Section 2.4, can be used in classification trees. Estimation of the nonlinear logistic model (13) can be carried out by combining the iteratively re-weighted least squares (IRWLS) method with the Gauss–Newton plus variable projection method discussed earlier. The algorithm is described in Appendix A. The LRT for testing  $H_0 : \beta_1 = 0$  with the nonlinear logistic model (13) can be shown to have the following form:

$$\text{LRT} = 2 \sum_{d \in \{L,R\}} \sum_{k=0,1} \tilde{n}_{dk} \log \frac{\tilde{n}_{dk}}{E_{dk}}, \text{ with } E_{dk} = \frac{n_k \tilde{n}_d}{n},$$

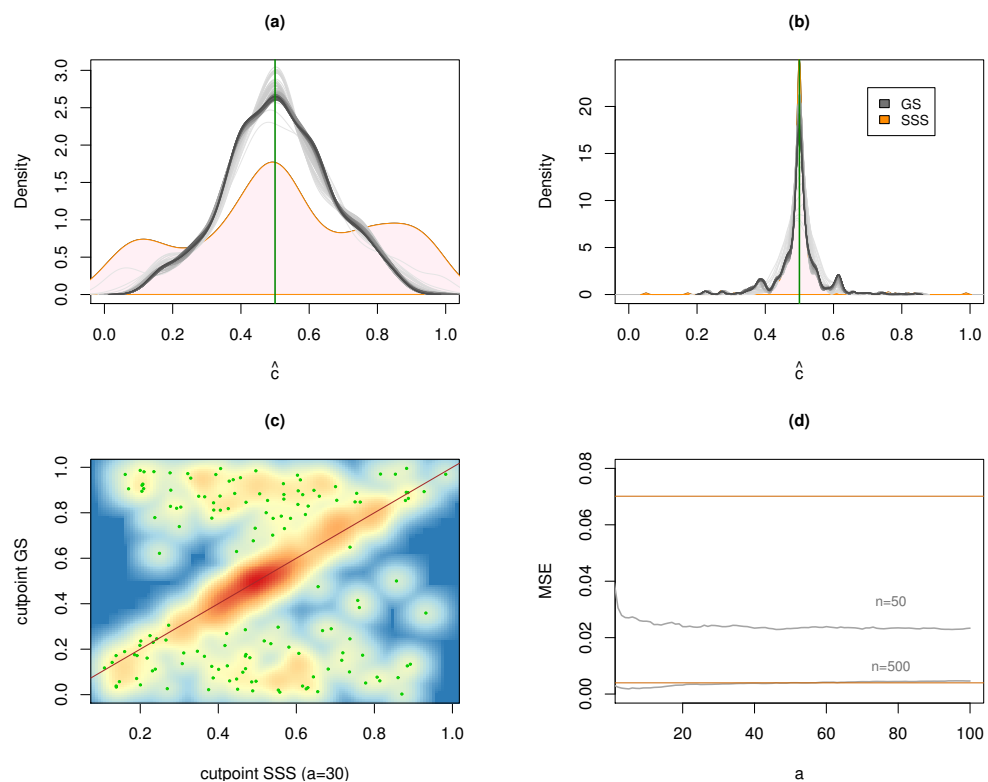
which is similar to its counterpart in the linear logistic model except that the counts  $n_{dk}$  involving the split are now replaced with their approximated surrogates  $\tilde{n}_{dk}$ . The statistical significance of LRT refers to a  $\chi^2(2)$  distribution except for the scenario with binary predictors where  $\text{LRT} \stackrel{H_0}{\sim} \chi^2(1)$ . Alternatively, Pearson  $\chi^2$  of form  $\sum_{d \in \{L,R\}} \sum_{k=0,1} (\tilde{n}_{dk} - E_{dk})^2 / E_{dk}$  can be used instead. Comparing across predictors, each with its best cutoff point, the split that yields the smallest  $p$ -value or the largest logworth is selected to partition the data. As before, both ordinal variables (with small  $3 \leq K \leq 10$  distinct levels) and nominal variables can be treated separately, resulting in a hybrid approach. In particular, Boulesteix [41,42] provided explicit formulas for computing the exact  $p$ -values associated with the maximally selected  $\chi^2$  variables in the classification scenario for both nominal and ordinal variables, as implemented in the R package `exactmaxsel`. However, the combinatorial approach of [42] becomes computationally infeasible when both  $n$  and  $K$  are large, a scenario that warrants further research.

### 3.1. Simulation Studies

To illustrate, we simulated data from a logistic model

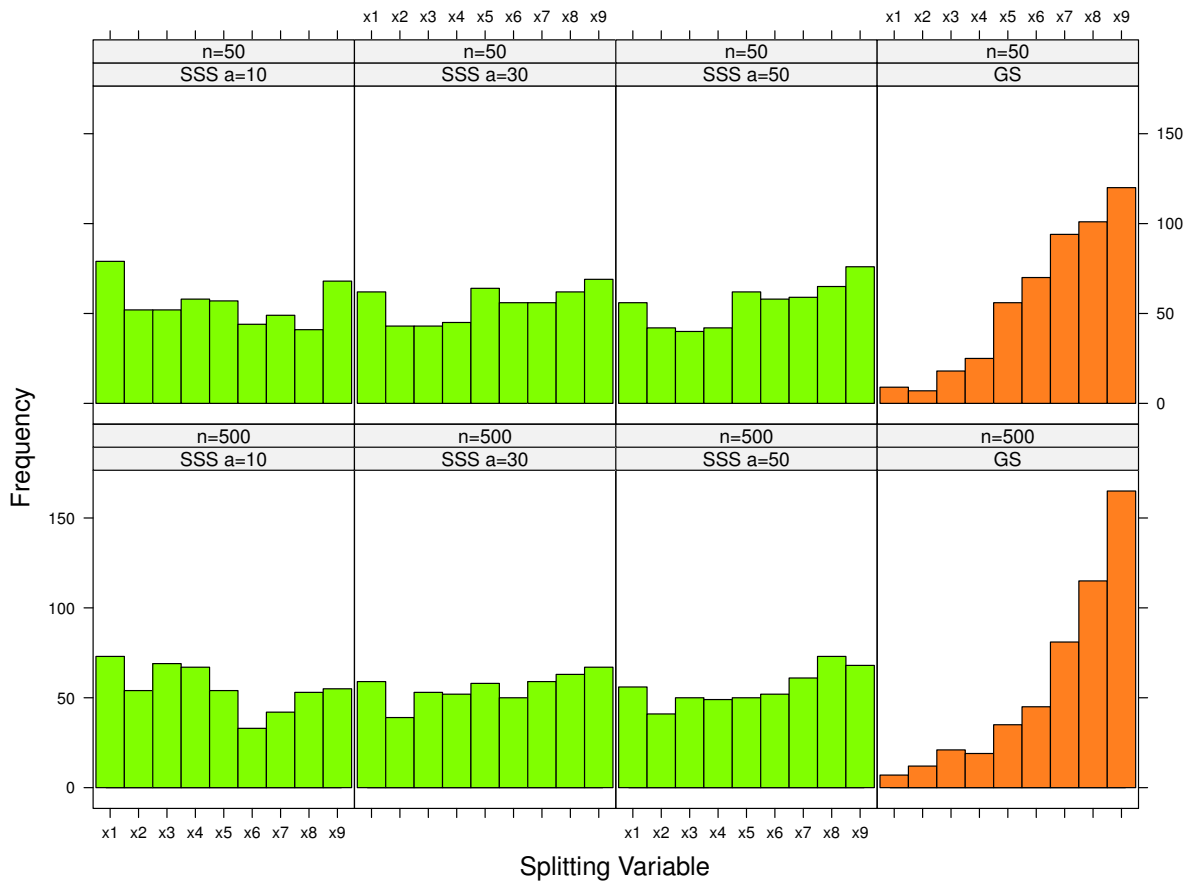
$$\text{Model E : } \Pr\{y = 1\} = \text{logistic}(-0.5 + I\{x \leq 0.5\}), \tag{16}$$

with  $x \sim \text{Unif}[0, 1]$ . Two sample sizes ( $n = 50$  and  $n = 500$ ) and 500 simulation runs were considered. We also experimented with different choices of  $a \{5, 10, 15, \dots, 100\}$  with SSS. Figure 11a,b provides the empirical density curves of  $\hat{c}$  identified by the entropy-based GS and SSS for  $n = 50$  and  $n = 500$ , respectively. It can be seen that GS and SSS have very similar performances for the large ( $n = 500$ ) samples. When  $n = 50$ , SSS substantially outperforms GS. The distribution of GS shows two bumps at either end. This can be explained again by the end-cut performance problem. To see this, Figure 11c presents a smooth scatterplot of the best cutoff point found by GS versus that found by SSS with  $a = 30$  in the smaller samples scenario with  $n = 50$ . Because there are many overlapping points, we have smoothed the scatterplot in such a way that the frequencies of overlapped points are represented in the electromagnetic spectrum. The zigzag pattern clearly indicates that GS selects more cuts that are near to either end of the data range. This is because the signal strength of Model E in (16) is not strong enough to overpower the inherent end-cut tendency, while SSS largely avoids this issue. To this end, a smaller  $a$  seemingly provides even better performance for SSS. Figure 11d plots the MSE with respect to the true cutoff point  $c_0 = 0.5$  versus the  $a$  value. SSS has very comparable performance with GS in the case of larger samples, but triumphs with smaller samples for all choices of  $a$ .



**Figure 11.** Comparison of SSS and GS in finding the best cutoff point  $c = 0.5$  for classification trees. The results are based on 500 simulation runs. Panels (a,b) plot the estimated density curves for  $\hat{c}$  with  $n = 50$  and  $n = 500$ , respectively. Panel (c) is the scatterplot (smoothed in such a way that the frequencies of overlapped points are represented in the electromagnetic spectrum) of the cutoff point identified by GS versus that identified by SSS when  $n = 50$ . Panel (d) plots the MSE vs.  $a$ , superimposed as orange solid lines with MSE from GS.

To investigate the variable selection bias, we simulated data from a null logistic model with similar settings to those in Section 2.4. Specifically, each dataset contained nine features  $\{X_1, X_2, \dots, X_9\}$ , with the possible number of distinct values being equal to  $\{2, 3, 4, 5, 10, 20, 50, 100, 500\}$ , respectively. Three choices of  $a$ , namely,  $\{10, 30, 50\}$ , were experimented with for SSS. The bar plots in Figure 12 provide a comparison of the selection frequencies from SSS and GS, where the hybrid method with SSS largely eliminates the selection bias of GS.



**Figure 12.** Comparison of SSS (combined with Boulesteix [42]’s method) and GS in terms of selection bias for classification trees. The bar plots are based on frequencies (out of 500 simulation runs) of splitting. Variables were selected by either method. Data were generated from a null model. Two sample sizes of  $n = 50$  and  $n = 500$  were considered.

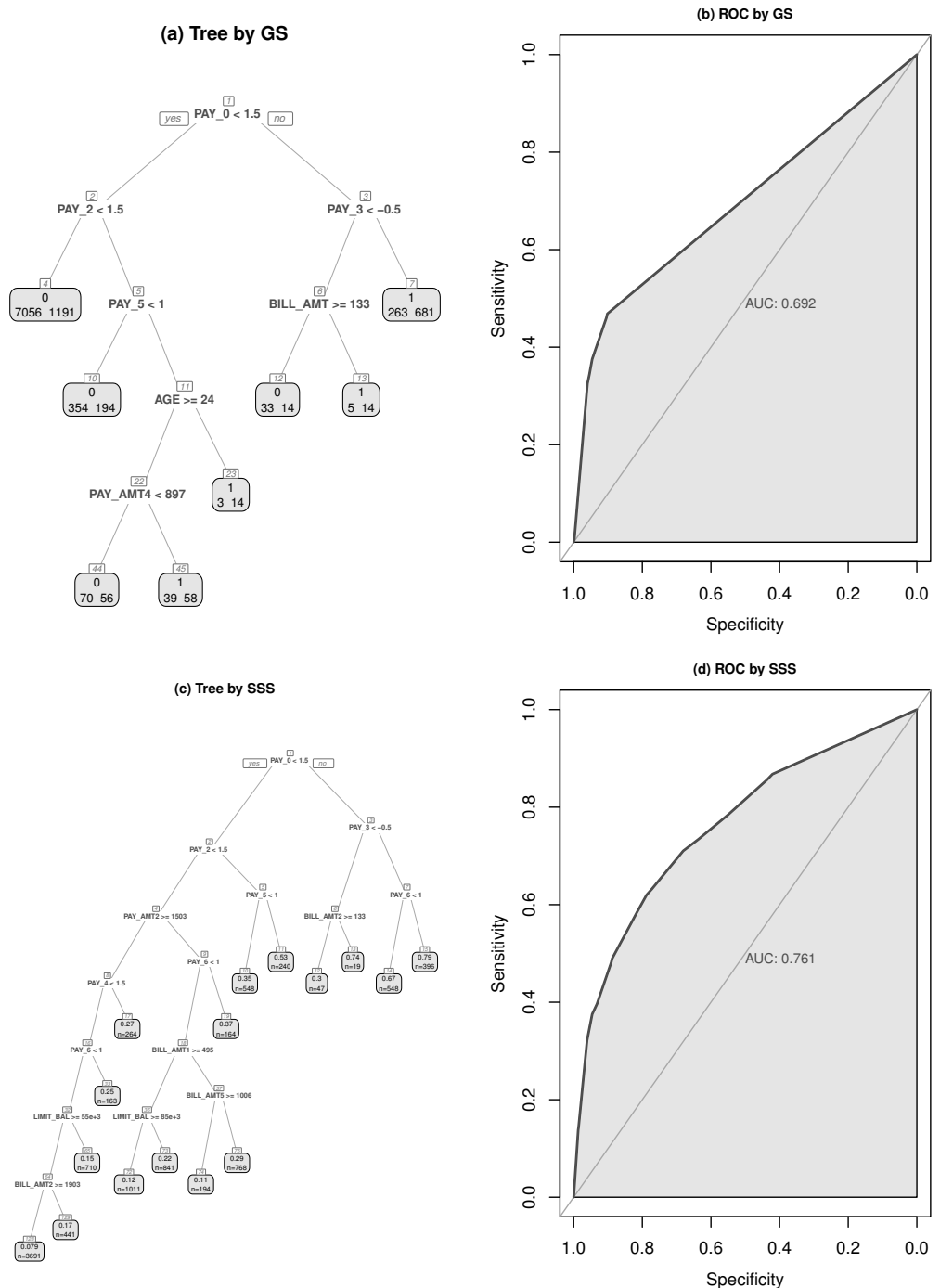
### 3.2. Example: Credit Card Default Data

To illustrate the use of SSS in classification trees, we considered the credit card default dataset [43] available at the UCI Machine Learning Repository. The primary objective of the study was to predict the default status of credit card holders based on 23 predictor variables related to their financial information, such as business financial statements, customer transactions, and repayment records. The dataset contains 30,000 observations, presenting a relatively large-scale problem. More details about the dataset can be found in [44].

As before, we randomly partitioned the data into three sets, the training sample  $\mathcal{L}_1$ , validation sample  $\mathcal{L}_2$ , and test sample  $\mathcal{L}_3$ , each containing one-third of the entire data. A large initial tree was grown and pruned using entropy as the impurity measure with  $\mathcal{L}_1$ . The final tree structure was determined through validation on  $\mathcal{L}_2$  based on the misclassification error. Finally, the resulting tree structure was evaluated using the test sample  $\mathcal{L}_3$ . Two performance metrics were computed: the misclassification error rate, and the area (AUC) under the receiver operating characteristic (ROC) curve.

Figure 13 displays the final tree structures for both GS and SSS. The GS tree contains eight terminal nodes, while the SSS tree has sixteen terminal nodes. The splits in the top

levels are essentially the same for both models. Based on the test sample, the misclassification error rates are 0.1792 for the SSS tree and 0.1798 for the GS tree, indicating that they are quite similar. These results align closely with those reported in [44]. Figure 13 also shows the ROC curves. The SSS tree achieves an AUC value of 0.760, significantly surpassing the AUC value of 0.692 for the GS tree. This indicates that the SSS tree is more effective at distinguishing between credible and non-credible cardholders.



**Figure 13.** Final classification trees for the credit card default data. The misclassification error rates are 0.1792 for the SSS tree and 0.1798 for the GS tree.

#### 4. Discussion

In this paper, we have explored an intuitive alternative to greedy search in decision trees by approximating step functions with smooth sigmoid functions. We demonstrate

that the proposed SSS method can be reformulated as a one-dimensional optimization in various recursive partitioning procedures, resulting in improved computational efficiency. SSS facilitates a parametric smoothing or regularization of the erratic goodness-of-split measures computed in greedy search. The smoothing effect provides SSS with remarkable immunity to the end-cut preference problem in GS. By smoothing out local peaks that stem from large variation and end-cut preference, SSS outperforms GS substantially in identifying weak signals. Moreover, its natural connection with nonlinear regression renders a way of alleviating the variable selection bias problem. To this end, our ongoing research efforts are being directed towards investigating how valid statistical inferences can be made based on the best split on nominal predictors and fully addressing overoptimism owing to ‘ordinalization’.

We have confined our attention to the use of SSS in settings where the splitting statistic is of explicit form. This allows us to formulate identification of the best cutoff point as a one-dimensional smooth optimization problem. To solve the optimization, we have chosen the Brent [23] method for its easy accessibility and fast and stable performance. Other algorithms can be used instead to provide faster convergence rates or carry out global optimization. SSS can also be useful in scenarios without a closed form for the splitting criterion or for updating (see Chen et al. [45] and Su et al. [46] for examples where evaluation of each cutpoint involves iterations). The approximation idea of SSS can be incorporated into the associated models, which can then be estimated with the aid of separable least squares, as discussed in Appendix A.

Our discussion has mainly focused on searching for the best cutoff point of a predictor and one single split of data. We have demonstrated that SSS can be a superior alternative to GS, with several appealing advantages. Our results suggest that implementation of many recursive partitioning methods and their extensions can benefit from the SSS approach. Tree-based methods have been extended to accommodate various types of data, and their applications have expanded to serve a wider range of analytical purposes. A key idea is to formulate the search for the optimal cutoff point as a one-dimensional optimization problem whenever possible. For example, incorporating SSS into ensemble learning algorithms may enhance predictive accuracy. The use of SSS in random forests [6,47] may lead to better variable importance ranking. Future research to see how the benefits of SSS carry over to various recursive partitioning settings and impact their final results is warranted. Additionally, SSS can be especially beneficial for large-scale or high-dimensional data. Tree models process variables individually and perform automatic variable selection, making them well suited for high-dimensional analysis even in cases where  $p \gg n$ . However, increased dimensionality can lead to greater computational burden. In these scenarios, the efficiency of SSS makes its incorporation especially advantageous, helping to alleviate the challenges associated with high dimensionality. Implementing SSS in a lower-level programming language is another promising future research avenue regarding computational efficiency concerns.

**Author Contributions:** All authors contributed equally to the conceptualization, methodology, software, validation, formal analysis, investigation, resources, writing—original draft preparation, writing—review and editing, and visualization. All authors have read and agreed to the published version of the manuscript.

**Funding:** Author Lei Liu is partially supported by NIH R21 AG084054 and UL1 TR002345.

**Data Availability Statement:** All datasets used in our analysis are publicly available from the UCI Machine Learning Repository (<https://archive.ics.uci.edu/>, accessed on 20 September 2024) and the R package mlbench (Machine Learning Benchmark Problems). Additionally, we have made our R code available on GitHub <https://github.com/xgsu/SSS>, accessed on 20 September 2024.

**Conflicts of Interest:** Author Yishu Wei was employed by the company Reddit Inc. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.



### Appendix A. Gauss–Newton Algorithm with Variable Projection

To estimate the nonlinear model in (3), i.e.,  $y_i = \beta_0 + \beta_1 s(c; x_i) + \varepsilon_i = \eta_i + \varepsilon_i$ , the Gauss–Newton algorithm with variable projection (Kaufman, 1975) described in Algorithm A1 was used. In the algorithm, note that the intermediary updating step on  $\beta$  is actually

$$\beta := \beta + \delta = (\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T \mathbf{y}.$$

We denote  $\eta = \mathbf{S}\beta = [\eta_i]$ . The Jacobian matrix of  $\eta$ , denoted as  $\mathbf{F}$ , is

$$\mathbf{F} = \frac{d\eta}{d\theta} = [\mathbf{S}, \mathbf{v}], \text{ with } \mathbf{v} = \frac{d\eta}{dc} = [-a \ \beta_1 s_i (1 - s_i)].$$

The matrix  $\mathbf{F}$  is closely related to the linearization step in nonlinear models, and plays a critical role in both the optimization algorithm and statistical inference.

---

#### Algorithm A1 Gauss–Newton Algorithm with Variable Projection

---

initialize  $\theta = (\beta_0, \beta_1, c)^T$ .

repeat

    compute  $\delta = (\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T \mathbf{r}$  by first obtaining  $\mathbf{S}$  and residual vector  $\mathbf{r} = \mathbf{y} - \mathbf{S}\beta$ ;

    update  $\beta := \beta + \delta$ .

    evaluate Jacobian  $\mathbf{F} = d\eta/d\theta$  at current  $(\beta, c)$ ;

    compute direction  $\mathbf{d} = (\mathbf{F}^T \mathbf{F})^{-1} \mathbf{F}^T (\mathbf{y} - \mathbf{S}\beta)$ ;

    update  $\theta := \theta - \alpha \mathbf{d}$  with step size  $\alpha > 0$ ;

until convergence.

---

We next consider estimation of the nonlinear logit model (13):  $\log\{\pi_i/(1 - \pi_i)\} = \beta_0 + \beta_1 s(c; x_i)$ . The associated log-likelihood function is

$$\ell(\theta) = \sum_{i=1}^n \{y_i \eta_i + \log(1 - \pi_i)\}.$$

Note that  $\theta$  enters the likelihood through  $\eta_i$  and  $\pi_i = \text{logistic}(\eta_i)$ . Fisher scoring, a popular variant of Newton–Raphson in which the Hessian matrix is replaced by its expected value for simpler form, is the standard method for GLM. To apply it, we can verify that the gradient is

$$\mathbf{g} = \frac{d\ell}{d\theta} = \mathbf{F}^T (\mathbf{y} - \boldsymbol{\pi})$$

and the minus expected Hessian matrix (which is  $n$  times the Fisher information matrix) is  $\mathbf{F}^T \mathbf{W} \mathbf{F}$ , where matrix  $\mathbf{W}$  is diagonal with diagonal elements  $w_i = \pi_i(1 - \pi)$  for  $i = 1, \dots, n$ . It is worth noting that the observed Fisher information matrix is not equal to the expected Fisher information matrix with this nonlinear logistic model, even though the canonical logistic link is used.

Therefore, the updating step is

$$\begin{aligned} \theta &:= \theta + (\mathbf{F}^T \mathbf{W} \mathbf{F})^{-1} \mathbf{F}^T (\mathbf{y} - \boldsymbol{\pi}) \\ &:= (\mathbf{F}^T \mathbf{W} \mathbf{F})^{-1} \mathbf{F}^T \mathbf{W} \left\{ \mathbf{F}\theta + \mathbf{W}^{-1} (\mathbf{y} - \boldsymbol{\pi}) \right\}. \end{aligned}$$

We define a new continuous response vector  $\mathbf{y}' = \mathbf{F}\theta + \mathbf{W}^{-1} (\mathbf{y} - \boldsymbol{\pi})$ . The updated  $\theta$  can be computed as a weighted least squares (WLS) estimate through model

$$\mathbf{y}' = \mathbf{F}\theta + \boldsymbol{\varepsilon}, \text{ with } \boldsymbol{\varepsilon} \sim (\mathbf{0}, \mathbf{W}^{-1}).$$

The associated WLS criterion is  $\| \mathbf{W}^{1/2} (\mathbf{y}' - \mathbf{F}\theta) \|^2$ . Consequently, it can be immediately recognized in the spirit of the Gauss–Newton approach that the updated  $\theta$  can also be solved through a separable (weighted) nonlinear least squares problem  $\| \mathbf{W}^{1/2} (\mathbf{y}' - \mathbf{S}\beta) \|^2$  associated

with the model  $\mathbf{y}' = \mathbf{S}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ . This implies that the updated  $\boldsymbol{\theta}$  can also be solved via Algorithm A1 with response  $\mathbf{y}'$  and additional weights  $\mathbf{W}$ . We may call this method iteratively re-weighted separable nonlinear least squares (IRW-SNLS), as presented in Algorithm A2.

---

**Algorithm A2** Iteratively Re-Weighted Separable Nonlinear Least Squares (IRWSNLS)

---

initialize  $\boldsymbol{\theta} = (\boldsymbol{\beta}, c)^T$ .  
**repeat**  
    compute quantities  $s_i = s(c; x_i)$  and  $\pi_i = \text{logistic}(\beta_0 + \beta_1 s_i)$  for  $i = 1, \dots, n$ ;  
    form  $\boldsymbol{\pi} = [\pi_i]$ ,  $\mathbf{W} = \{\pi_i(1 - \pi_i)\}$ ,  $\mathbf{v} = [-a \ \beta_1 s_i(1 - s_i)]$ , and  $\mathbf{F} = \{\mathbf{j}, \mathbf{s}, \mathbf{v}\}$ ;  
    form new response vector  $\mathbf{y}' = \mathbf{F}\boldsymbol{\theta} + \mathbf{W}^{-1}(\mathbf{y} - \boldsymbol{\pi})$ ;  
    update  $\boldsymbol{\theta} = \arg \min_{\boldsymbol{\theta}} \|\mathbf{W}^{1/2}(\mathbf{y}' - \mathbf{S}\boldsymbol{\beta})\|^2$  via Algorithm A1;  
**until** convergence.

---

**Appendix B. Proofs**

*Appendix B.1. Proof of Proposition 1*

When the GS splitting is conducted via the updating formula, the method entails sorting the response  $Y$  values in the ascending (or descending) order of  $X$ . For general-purpose sorting, a stable algorithm is at best  $O\{\ln(n) n\}$ . For example, the `sort` function in R is either  $O(n^{4/3})$  if Shellsort is used, or  $O\{\ln(n) n\}$  if the Quicksort method is used. As noted earlier, centering  $Y$  yields further simplification of the objective function. The operation of centering is  $O(n)$ , and consequently does not escalate the computational complexity level asymptotically.

Without updating, GS computes the least squares criterion for every distinct splitting point of  $X$ . Note that extracting the unique values of  $X$  is  $O(n)$  in general; therefore, the total complexity amounts to  $O(Kn)$  in this case, where  $K$  is the number of distinct values of  $X$ . Because  $X$  is continuous,  $K = O(n)$ ; hence, the complexity becomes  $O(n^2)$ .

For SSS, each iterative step involves evaluation of the objective function, which is  $O(n)$ . SSS requires standardization of  $X$  and transformation of  $\hat{c}$  back to the original scale, both operations being  $O(n)$ . We also suggest an option of using the range of the  $\gamma$ th and  $(1 - \gamma)$ th percentiles in Brent’s method. Note that computation of the two percentiles is  $O(n)$  as well. Put together, its complexity is  $O(mn)$ , where  $m$  is the number of iterative steps in the optimization algorithm.

*Appendix B.2. Proof of Proposition 2*

Recall that the reduction  $\Delta i$  in node impurity due to a split is provided by

$$\Delta i = i(t) - \{p_L i(t_L) + p_R i(t_R)\},$$

where  $p_L = n_L/n$  and  $p_R = n_R/n = 1 - p_L$ . The first term  $i(t)$  does not depend on the specific split at node  $t$ ; thus, we can omit it and focus on the remaining part  $p_L i(t_L) + p_R i(t_R)$ , which isolates the impact of the split on node impurity. Note that eliminating the negative sign in the front transforms the optimization problem from a maximization problem to a minimization problem.

With the entropy impurity measure, up to a constant, we have the following:

$$\Delta i \hat{=} p_L \{p_{L1} \log p_{L1} + (1 - p_{L1}) \log(1 - p_{L1})\} + p_R \{p_{R1} \log p_{R1} + (1 - p_{R1}) \log(1 - p_{R1})\}.$$

Note that

$$p_L p_{L1} = \frac{n_L}{n} \frac{n_{L1}}{n_L} = \frac{n_{L1}}{n}.$$

A similar form holds for the other terms  $p_L(1 - p_{L1})$ ,  $p_R p_{R1}$ , and  $p_R(1 - p_{R1})$ . The constant  $n$  in the denominator for all of these terms can be removed, as it does not depend on the specific split. Thus,  $\Delta I$  becomes

$$\Delta I \hat{=} n_{L1} \log \frac{n_{L1}}{n_L} + (n_L - n_{L1}) \log \frac{n_L - n_{L1}}{n_L} + n_{R1} \log \frac{n_{R1}}{n_R} + (n_R - n_{R1}) \log \frac{n_R - n_{R1}}{n_R},$$

for which the denominator parts can be further reduced:

$$-n_{L1} \log n_L - (n_L - n_{L1}) \log n_L - n_{R1} \log n_R - (n_R - n_{R1}) \log n_R = -n_L \log n_L - n_R \log n_R.$$

This yields

$$\Delta I \hat{=} n_{L1} \log n_{L1} + (n_L - n_{L1}) \log(n_L - n_{L1}) + n_{R1} \log n_{R1} + (n_R - n_{R1}) \log(n_R - n_{R1}) - n_L \log n_L - n_R \log n_R.$$

Finally, plugging in  $n_R = n - n_L$  and  $n_{R1} = n_1 - n_{L1}$  yields the form as provided in (14). Using the Gini index and similarly omitting or canceling out irrelevant constants, we have

$$\begin{aligned} \Delta I &\hat{=} p_L p_{L1}(1 - p_{L1}) + p_R p_{R1}(1 - p_{R1}) \\ &\hat{=} \frac{n_L n_{L1}}{n} \frac{n_L - n_{L1}}{n_L} + \frac{n_R n_{R1}}{n} \frac{n_R - n_{R1}}{n_R} \\ &\hat{=} \frac{n_{L1}(n_L - n_{L1})}{n_L} + \frac{n_{R1}(n_R - n_{R1})}{n_R} \\ &\hat{=} \frac{n_{L1}(n_L - n_{L1})}{n_L} + \frac{(n_1 - n_{L1})\{(n - n_L) - (n_1 - n_{L1})\}}{n - n_L}, \end{aligned}$$

which yields (15). This completes the proof.

## References

- Breiman, L.; Friedman, J.; Olshen, R.; Stone, C. *Classification and Regression Trees*; Wadsworth International Group: Belmont, CA, USA, 1984.
- Morgan, J.; Sonquist, J. Problems in the analysis of survey data and a proposal. *J. Am. Stat. Assoc.* **1963**, *58*, 415–434. [\[CrossRef\]](#)
- Friedman, J. Multivariate adaptive regression splines (with discussion). *Ann. Stat.* **1991**, *19*, 1–141.
- Breiman, L. Bagging predictors. *Mach. Learn.* **1996**, *26*, 123–140. [\[CrossRef\]](#)
- Freund, Y.; Schapire, R. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*; Morgan Kaufman: San Francisco, CA, USA, 1996; pp. 148–156.
- Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [\[CrossRef\]](#)
- LeBlanc, M.; Crowley, J. Survival trees by goodness of split. *J. Am. Stat. Assoc.* **1993**, *88*, 457–467. [\[CrossRef\]](#)
- Therneau, T.M. User Written Splitting Functions for RPART. 2023. Available online: <https://cran.r-project.org/web/packages/rpart/vignettes/usercode.pdf> (accessed on 20 September 2024).
- Loh, W.-Y. Regression trees with unbiased variable selection and interaction detection. *Stat. Sin.* **2002**, *12*, 361–386.
- Miller, R.; Siegmund, D. Maximally selected chi square statistics. *Biometrics* **1982**, *38*, 1011–1016. [\[CrossRef\]](#)
- Shih, Y.-S.; Tsai, H.-W. Variable selection bias in regression trees with constant fits. *Comput. Data Anal.* **2004**, *45*, 595–607. [\[CrossRef\]](#)
- Hothorn T.; Zeileis A. Generalized maximally selected statistics. *Biometrics* **2008**, *64*, 1263–1269. [\[CrossRef\]](#)
- Segal, M.R. Tree-structured methods for longitudinal data. *J. Am. Stat. Assoc.* **1992**, *87*, 407–418. [\[CrossRef\]](#)
- Su, X.; Tsai, C.-L.; Wang, M. Tree-structured model diagnostics for linear regression. *Mach. Learn.* **2009**, *74*, 111–131. [\[CrossRef\]](#)
- Domingos, P.; Hulten, G. Mining high-speed data streams. In *Proceedings KDD 2000*; ACM Press: New York, NY, USA, 2000; pp. 71–80.
- Rosenblatt, F. *Principles of Neurodynamics*; Spartan Books: Washington, DC, USA, 1962.
- Jordan, M.; Jacobs, R. Hierarchical mixtures of experts and the EM algorithm. *Neural Comput.* **1994**, *6*, 181–214. [\[CrossRef\]](#)
- Ciampi, A.; Couturier, A.; Li, S. Prediction trees with soft nodes for binary outcomes. *Stat. Med.* **2002**, *21*, 1145–1165. [\[CrossRef\]](#)
- Balakrishnan, N. (Ed.) *Handbook of the Logistic Distribution*; Dekker: New York, NY, USA, 1992.
- Johnson, N.L.; Kotz, S.; Balakrishnan, N. *Continuous Univariate Distributions*, 2nd ed.; Wiley-Interscience: New York, NY, USA, 1995; Volume 2.
- Seber, G.A.; Wild, C.J. *Nonlinear Regression*; Wiley-Interscience: New York, NY, USA, 2003.
- Björck, A. *Numerical Methods for Least Squares Problems*; SIAM: Philadelphia, PA, USA, 1996.
- Brent, R. *Algorithms for Minimization without Derivatives*; Prentice-Hall: Englewood Cliffs, NJ, USA, 1973.

24. R Core Team. *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2024. Available online: <https://www.R-project.org/> (accessed on 20 September 2024).
25. Shpak, A. Global optimization in one-dimensional case using analytically defined derivatives of objective function. *Comput. Sci. J. Mold.* **1995**, *3*, 168–184.
26. Xu, P. A hybrid global optimization method: The one-dimensional case. *J. Comput. Appl. Math.* **2002**, *147*, 301–314. [[CrossRef](#)]
27. Torgo, L. A study on end-cut preference in least squares regression trees. In *Progress in Artificial Intelligence, Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 104–115.
28. Therneau, T.M.; Atkinson, E.J. An Introduction to Recursive Partitioning Using the RPART Routines. 2023. Available online: <https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf> (accessed on 20 September 2024).
29. Ishwaran, H. The effect of splitting on random forests. *Mach. Learn.* **2015**, *99*, 75–118. [[CrossRef](#)] [[PubMed](#)]
30. Zeileis, A.; Hothorn, T.; Hornik, K. Model-based recursive partitioning. *J. Comput. Graph. Stat.* **2008**, *17*, 492–514. [[CrossRef](#)]
31. Yao, Y.-C.; Davis, R.A. The asymptotic behavior of the likelihood ratio statistic for testing a shift in mean in a sequence of independent normal variables. *Sankhyā Indian J. Stat. Ser. A* **1986**, *48*, 339–353.
32. Bai, J. Estimation of a change point in multiple regression models. *Rev. Econ. Stat.* **1997**, *79*, 551–563. [[CrossRef](#)]
33. Golub, G.; Pereyra, V. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM J. Numer. Anal.* **1973**, *10*, 413–432. [[CrossRef](#)]
34. Kaufman, L. Variable projection method for solving separable nonlinear least squares problems. *BIT Numer.* **1975**, *15*, 49–57. [[CrossRef](#)]
35. Osborne, M.R. Separable least squares, variable projection, and the Gauss-Newton Algorithm. *Electron. Numer. Anal.* **2007**, *28*, 1–15.
36. Hawkins, D. M. Testing a sequence of observations for a shift in location. *J. Am. Stat.* **1977**, *72*, 180–186. [[CrossRef](#)]
37. Genz, A. Numerical computation of multivariate normal probabilities. *J. Comput. Graph.* **1992**, *1*, 141–149. [[CrossRef](#)]
38. Hothorn, T.; Lausen, B. On the exact distribution of maximally selected rank statistics. *Comput. Data Anal.* **2003**, *43*, 121–137. [[CrossRef](#)]
39. Morgan, J.N.; Messenger, R.C. *THAID: A Sequential Search Program for the Analysis of Nominal Scale Dependent Variables*; Survey Research Center, Institute for Social Research, University of Michigan: Ann Arbor, MI, USA, 1973.
40. Neyman, J. Contributions to the theory of the  $\chi^2$  test. In *First Berkeley Symposium on Mathematical Statistics and Probability*; University of California Press: Berkeley, CA, USA, 1949; pp. 239–273.
41. Boulesteix, A.-L. Maximally selected chi-square statistics for ordinal variables. *Biom. J.* **2006**, *48*, 451–462. [[CrossRef](#)]
42. Boulesteix, A.-L. Maximally selected chi-square statistics and binary splits of nominal variables. *Biom. J.* **2006**, *48*, 838–848. [[CrossRef](#)]
43. Yeh, I. Default of Credit Card Clients [Dataset]. UCI Machine Learning Repository. 2009. Available online: <https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients> (accessed on 20 September 2024).
44. Yeh, I.; Lien, C. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Syst. Appl.* **2009**, *36*, 2473–2480. [[CrossRef](#)]
45. Chen, J.; Yu, K.; Hsing, A.; Therneau, T.M. A partially linear tree-based regression model for assessing complex joint gene-gene and gene-environment effects. *Genet. Epidemiol.* **2007**, *31*, 238–251. [[CrossRef](#)]
46. Su, X.; Meneses, K.; McNeese, P.; Johnson, W.O. Interaction Trees: Exploring the differential effects of an intervention program for breast cancer survivors. *J. R. Stat. Soc. Ser. C* **2011**, *60*, 457–474. [[CrossRef](#)]
47. Su, X.; Peña, A.T.; Liu, L.; Levine, R.A. Random forests of interaction trees for estimating individualized treatment effects in randomized trials. *Stat. Med.* **2018**, *37*, 2547–2560. [[CrossRef](#)] [[PubMed](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.