Universidade Federal do Rio Grande do Sul
Instituto de Matemática e Estatística
Programa de Pós-Graduação em Estatística

Combining LASSO-Type Methods with a Smooth Transition Random Forest

Alexandre Luís Debiasi Gandini

Porto Alegre, Setembro de 2022.

Dissertação submetida por Alexandre Luís Debiasi Gandini como requisito parcial para a obtenção do título de Mestre em Estatística pelo Programa de Pós-Graduação em Estatística da Universidade Federal do Rio Grande do Sul.

Orientador:

Prof. Dr. Flávio A. Ziegelmann (UFRGS)

Comissão Examinadora:

Prof. Dr. Álvaro Veiga (PUC-Rio)

Prof. Dr. Eduardo Fonseca Mendes (FGV EESP)

Prof. Dr. Márcio Valk (UFRGS)

Data de Apresentação: 27 de Setembro de 2022.

Acknowledgements

Firstly, I'd like to express my thanks to my supervisor Flávio who has supported me throughout this research project. Also thanks to all classmates and professors I had contact during the program, I wouldn't be able to finish it without some degree of participation of every one of you.

Last, but not least, I'd like to thank my family, starting with my parents Ari and Inês, who provided me with good values and a great education, my wife Nádia and my kids Gustavo, Amanda and Júlia, for allowing me the time needed to do this work instead of being with you.

Abstract

In this work, we propose a novel hybrid method for the estimation of regression models, which is based on a combination of LASSO-type methods and smooth transition (STR) random forests. Tree-based regressions are known for their flexibility and skills to learn even very nonlinear patterns. The STR-Tree model introduces smoothness into traditional splitting nodes, leading to a non-binary labeling, which can be interpreted as a group membership degree for each observation. Our proposed approach has two steps, as follows: in the first step we fit a penalized linear regression, via LASSO-type methods, then, in the second step, we take the residuals from the first step fit and estimate a STR random forest for these residuals once more against the original covariates. Therefore, by doing so, we can capture the possibly important linear relationship in the data generating process, if any, via a parametric approach in the first step, and let a highly flexible model "attack" the non-linearities in the second step. We present numerical studies, both with simulated and real data, to illustrate the performance of our method. Our proposal has shown advantages in terms of predictive power in comparisson with other benchmarks, especially if the data possesses both linear and nonlinear features.

Resumo

Neste trabalho, propomos um novo método híbrido para a estimação de modelos de regressão, baseado em uma combinação de métodos do tipo LASSO e de random forest com transição suave (STR). Modelos de regressão baseados em árvores são conhecidos por sua flexibilidade e capacidade em reconhecer padrões até mesmo altamente não-lineares nos dados. O modelo STR-Tree introduz suavidade nos nós da árvore, levando à uma atribuição não-binária das observações em cada grupo, o que pode ser interpretado como diferentes graus de pertencimento a eles. Nossa proposta consiste em um método em dois passos, da seguinte forma: primeiro ajustamos uma regressão penalizada através do LASSO, então, no segundo passo, utilizamos os resíduos obtidos no primeiro ajuste e estimamos uma random forest com transição suave (STR) dos resíduos novamente contra as covariáveis originais. Procedendo dessa forma, podemos capturar as possíveis importantes relações lineares nos dados, caso presentes, de forma paramétrica no primeiro passo e deixar um modelo muito mais flexível "atacar" as não-linearidades em um segundo momento. Apresentamos estudos numéricos, tanto com dados simulados quanto com dados reais, para ilustrar o desempenho do nosso método. Nossa proposta se mostrou vantajosa em termos de poder preditivo em comparação com outras referências, especialmente se os dados contêm atributos tanto lineares quanto não-lineares.

Index

Chapter 1

INTRODUCTION

In this dissertation, we propose a novel hybrid method for the estimation of regression models, which is based on a combination of LASSO and STR random forests. The main contribution is presented in Chapter 2 in an article structure (Gandini and Ziegelmann, 2022). The current chapter introduces the main concepts employed in this study.

1.1 Tree-based methods

Tree-based methods have been widely applied in regression since the publication of the seminal CART (Classification and Regression Trees) paper from Breiman et al. (1984), which consolidated several approaches that were developed after the first appearence of a tree algorithm in the literature, with Morgan and Sonquist (1963). By the occasion of the 50 years anniversary of this innovative publication, Loh (2014) compiled some of the most important contributions to modeling using trees that happened in that time period.

Some of the development ever since have focused in solving classification problems, the ones where the response has a finite number of classes, which do not present a numerical order meaning. Well known tree algorithms for classification include ID3 (Quinlan, 1986), its successor C4.5 (Quinlan, 1993) and CHAID (Kass, 1980). In this work we center our attention on regression problems, i.e. we have a continuous ordered response.

The CART approach is a greedy, top-down algorithm that searches for the feature and value that split the observations from the root node on, recursivelly, until some stop criteria is met. This spliting proccess is guided by the response, because the estimated parameters for each node (variable and value) are the ones that produce the lowest combined mean squared error (MSE) in the two resulting groups after the split (Breiman et al., 1984). More on this can be found in the attached research paper (Gandini and Ziegelmann, 2022) in Chapter 2.

Besides this classical approach of *univariate* trees, the ones where a single variable

value is compared against a threshold in each node, there are some works that use the full vector of the variables space, defining a linear discriminant in every split. Those are called *multivariate linear* trees, see for example Murthy et al. (1994) and Yildiz and Alpaydın (2000). The sppliting can also be nonlinear, those are called *multivariate nonlinear* trees, as in the work of Guo and Gelfand (1992), whom use a simple neural network as the splitting function.

Other relevant variant to growing trees is to allow for oblique splits, instead of the typical axis-aligned, where the later fail to adequately capture linear relationships, using canonical correlation of the features as in the Canonical Correlation Trees paper (Rainforth and Wood, 2017). Another work worth mentioning is the Extremely Randomized Trees from Geurts et al. (2006), where both the set of variables and the splitting values are choosen at random in every node.

In all these works, the approach of hard splits in the observations space (a particular observation either meet or do not meet the criteria estimated by the parameters of a particular node of the tree) can cause problems in the estimation in the context of noisy data, especially on the boundaries of the training set, as demonstrated by Irsoy et al. (2012) and Linero and Yang (2017).

1.2 Introducing smoothness

More recently there has been an effort to develop trees that do not do hard assignments of data points to regions, with the introduction of some kind of smoothness in the spliting function. In contrast to the hard split trees, where an observation traverses a single path from the root to one of the terminal nodes, in smoothed trees an observation is linked to all regions, but with different associated degrees of group membership. An observation follows all the paths to all the regions, but with different weights. The final prediction is a combination of the predictions of each node. In essence, while in hard split regression trees there is only one possible final region for a particular observation, in smoothed trees we consider all K regions, but with different degrees of group membership.

Soft trees (Irsoy et al., 2012), Fuzzy trees (Suarez and Lutsko, 2000) and STR-trees (da Rosa et al., 2008) all use a sigmoid-like function in each node, assigning a group membership degree to the observations to be on the left or on the right child node. Soft trees and Fuzzy trees are very similar to each other, with only minor differences. They both are multivariate trees, accessing the whole vector of variables to establish the membership of observations down the structure of the tree. STR-trees follow the same guidelines but, in contrast, use a single variable in the sigmoid function.

Probabilistic Regression trees (Alkhoury et al., 2020) are another multivariate tree method with a smoothness effort, but it takes a different approach and rely on a parametric probability distribution chosen beforehand via expert knowledge or

selected through cross-validation.

In general, trees with soft assignments have the main advantage of providing a smooth response near the boundaries of the training data, where traditional hard split trees present discontinuity. This allows smooth trees to have lower bias in the predictions around the split boundaries (Irsoy et al., 2012). They also lead to fits that are smoother, therefore generalizing better, in the sense that they are less prone to overfit (Yildiz et al., 2016). Smoothness also brings in the possibility of estimating derivatives, such as elasticities, as well as a way of accessing variable importance.

Despite this advantage, smoothed trees are computationaly more intensive than traditional trees, and suffer from the same high variance problem that the latter ones. That is, the estimated parameters and predictions of a single tree present high variability and are for the most part dependent on the random subset of data they were fitted for (Wehenkel, 1997; Geurts et al., 2006; Geurts and Wehenkel, 2007).

1.3 Ensembling trees

One of the drawbacks of regression trees is that they are prone to overfitting, which can be thought as the situation where an estimated model fits the characteristics of a particular sample, instead of generalizing to the population structure (Miller et al., 2015). It usually is the result of an overly complex model, which leads to high error in the predictions of new data on which the model was not trained on (Breiman, 1996; Friedman, 1997; Dietterich, 2000).

This balance between fitting a model that has low prediction errors and also have the ability to generalize to new data is known as the bias-variance tradeoff. If we fully grow a tree until there is only one observation in each terminal node, for example, the model will have low bias for the predictions on that particular training set, but it will have high variance because even a small change in the data will lead to a very different structure of the tree. Usually, highly complex models have low bias and high variance, and for low complexity models it is the opposite (Berk, 2016).

One way of reducing the complexity of a regression tree is to prune it (remove some of the splits near the leafs) after it is fully grown (Hastie et al., 2009). Another way of overcoming overfitting is to create an ensemble of trees, in the sense that the final forecast is a combination of several individual predictions. The classical approachs are the Random Forest (Breiman, 2001) and Boosting (Friedman, 2001).

Ho (1995) shows that combining several trees improves predictive performance while simultaneously providing regularization. This way, there is no need to use pruning to attack the overfitting problem, we can fully grow trees and average their predictions. Every tree is fitted individually and the final estimator is an average of all individual models, resulting in a predictor with better performance than any of the individual trees (Rokach, 2010).

Since the growth of a traditional individual tree follows a deterministic procedure, in order to use a combination of them it is necessary to introduce some kind of randomness, enabling the creation of several slightly different trees. Ho (1998) accomplished that via randomly selecting the subset of candidate covariates in each splitting node (the random subspace method).

Breiman (1996) takes another approach and employs bagging, short for bootstrap aggregation, which is a procedure that trains several simple models, each one with slightly different data through bootstrap sampling (Efron, 1977) the original dataset. The final estimator is an average from all trees, as follows:

$$\hat{f}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_b(x)\,,$$

where B is the number of individual models to be trained. This increases the stability of the model, i.e., decreases its variance, easying the impact of extreme values on the data. We can think of this averaging of predictions as a way of imposing a regularizing effect on the model (Yildiz et al., 2016).

Random forest, introduced by Breiman (2001), combines the ideas behind bagging and the random subspace method. In this new perspective, the growing of several individual trees has two sources of randomization: the first one is the bootstrap sampling of the training set, where sometimes an observation will be drawn more than once and sometimes will not be drawn at all, whereas the second source is the random selection of the subset of candidate regressors in each split of the trees. This results in trees that are less correlated with each other, and, in a final estimator, averaged from the trees, with better predictive performance and less variance than either individual approaches (Hastie et al., 2009; Berk, 2016).

The insight behind the good predictions that random forests provide is that in a forest, the effect of any individual tree that overfits because of noisy training observations is dilluted in the averaging process. The individual trees will be slightly different from each other, therefore averaging them improves predicive performance in comparisson with any individual tree. Breiman (2001) presents theoretical results showing that the lower the correlations among trees, the lower the upper bound for the error of the ensemble. So it is critical to achieve diversity in the individual components that make a forest of trees. Typically any particular tree will have suboptimal splits, because the parameters are estimated locally, i.e., the whole tree structure is not considered at each node, only the data that reached that point. A node is not "aware" of further splits and their implications on the parameters estimation. Thus any individual tree is almost never the global optimal solution over all possible models, but an ensemble of those *weak learners* improves the overall predictive performance (Strobl et al., 2009).

As a complex method, with several degrees of randomization process, it is naturally hard to explain matematically the mecanics of how and why random forest works, besides intuitive explanations and simulation studies. Biau et al. (2008) and Lin and Jeon (2006) explore this issue, although imposing several simplifying assump-

tions, due to the inner complexity of the ensembling procedure.

More recently, other types of random forests ensemblings have been proposed, notably Extremely Randomized Forests (ERF) (Geurts et al., 2006), Canonical Correlation Forests (CCF) (Rainforth and Wood, 2017), Acceptance–Rejection Forests (ARF) (Calhoun et al., 2020) and Probabilistic Forests (PF-RF) (Alkhoury et al., 2020). They are all natural extensions of Extremely Randomized trees, Canonical Correlation trees, Acceptance-Rejection trees and Probabilistic trees to the classical random forest framework.

Finally, another way of ensembling trees is the boosting procedure, which was introduced by Friedman (2001) and is significantly different from the random forest. In boosting, trees are trained sequentially, where the next tree is fitted with greater weights on the observations that were poorly fitted in the previous step and lower weights on the ones that had a good fit. This *boosts* the performance of the model by forcing the trees to choose parameters that approximate better the observations that previous trees had difficulties dealing with (Miller et al., 2015). The final estimator is an additive model of all individual trees (Awaya and Ma, 2021). In the case of STR-Trees, the boosting approach for regression was stablished by the "BooST" method (Fonseca et al., 2020), which we use as one of the benchmarks in the simulation study (Chapter 2). Exploring the details of boosting is out of the scope of this work.

1.4 LASSO-type methods

Variable selection is important in modern statistical modeling, especially with the very large sizes of datasets available in recent years, both in terms of number of observations and in quantity of variables. Traditional methods like best subset or stepwise selection have their own difficulties dealing with a high number of regressors, with reduced accuracy of forecasts or large computational cost, which can make the estimation infeasible (Hastie et al., 2009).

Another approach is regularization, with methods like Ridge regression (Hoerl and Kennard, 1970) and Least Absolute Shrinkage and Selection Operator (LASSO) (Tibshirani, 1996). These procedures employ a penalization on the coefficients of a model, and shrink the magnitude of some of them, hence being known as shrinkage methods. While Ridge applies a ℓ_2-penalty on the coefficients estimation, resulting in lower βs than those estimated by OLS, LASSO imposes a ℓ_1-penalty and sets some of them to zero, effectively performing simultaneously parameter estimation and variable selection.

Zhao and Yu (2006) demonstrated that the ordinary LASSO regression do not have consistency in selecting the relevant variables and the true coefficients when the size of training set increases, and also do not have the *oracle* property (the estimator identifies the true model and has the optimal estimation rate) (Fan and Li, 2001; Zou, 2006). Zou (2006) proposes the Adaptive LASSO method (adaLASSO), which

has the *oracle* property and is asymptotically unbiased. This is done through the addition of different weights for each variable in the penalty factor.

Notice that nevertheless these LASSO-type approaches present the property of automatic variable selection, they are still linear methods, and capture only the linear relationships in the data. If the data has nonlinear (or both linear and nonlinear) generating terms, neither LASSO nor adaLASSO will be a good solution, and for this reason we propose a combination of those methods with a more flexible, non-parametric approach.

1.5 Proposal of this work

This work focus on regression problems where the DGP has both linear and nonlinear components. We propose a novel hybrid method for the estimation of regression models, which is based on a combination of LASSO and STR random forests, as a way of capturing the linearities and non-linearities in the data, in a flexible way.

The proposal is presented in the form of an article in Chapter 2, where i) first we make a brief revision of the methods in which we build upon, namely the traditional CART decision tree, the Tree-Structured Smoothed Transition Regression (STR-Tree), the Random Forest and the LASSO family of regularization methods; ii) we present our method, combining LASSO-type methods with a STR random forest, followed by numerical examples both with simulated and empirical data, and comparisson of our approach with other benchmarks; iii) we present our conclusions and provide suggestions for future related work.

Chapter 2

ARTICLE

The attached research article, *Combining LASSO-Type Methods with a Smooth Transition Random Forest* (Gandini and Ziegelmann, 2022), comprises the main contribution of the present Dissertation.

Combining LASSO-Type Methods with a Smooth Transition Random Forest

Alexandre L. D. Gandini[a], Flavio A. Ziegelmann[a]

[a]*Department of Statistics, Universidade Federal do Rio Grande do Sul*

Abstract

In this work, we propose a novel hybrid method for the estimation of regression models, which is based on a combination of LASSO-type methods and smooth transition (STR) random forests. Tree-based regressions are known for their flexibility and skills to learn even very nonlinear patterns. The STR-Tree model introduces smoothness into traditional splitting nodes, leading to a non-binary labeling, which can be interpreted as a group membership degree for each observation. Our proposed approach has two steps, as follows: in the first step we fit a penalized linear regression, via LASSO-type methods, then, in the second step, we take the residuals from the first step fit and estimate a STR random forest for these residuals once more against the original covariates. Therefore, by doing so, we can capture the possibly important linear relationship in the data generating process, if any, via a parametric approach in the first step, and let a highly flexible model "attack" the non-linearities in the second step. We present numerical studies, both with simulated and real data, to illustrate the performance of our method. Our proposal has shown advantages in terms of predictive power in comparisson with other benchmarks, especially if the data possesses both linear and nonlinear features.

Keywords: Regression, LASSO, adaLASSO, STR-Tree, Random Forest, Smoothness.

1. INTRODUCTION

Tree-based methods have been widely applied in regression since the publication of the seminal CART (Classification and Regression Trees) paper from

Email addresses: agandini@gmail.com (Alexandre L. D. Gandini), flavioaz@mat.ufrgs.br (Flavio A. Ziegelmann)

Breiman et al. (1984), which consolidated several approaches that were developed after the first appearence of a tree algorithm in the literature, with Morgan & Sonquist (1963). They are known to be highly flexible in the way they can capture hidden, possibly nonlinear, patterns from the data (Hastie et al., 2009).

The CART approach is a greedy, top-down algorithm that searches for the feature and value that split the observations from the root node on, recursivelly, until some stop criteria is met. This spliting proccess is guided by the response, because the estimated parameters for each node (variable and value) are the ones that produce the lowest combined mean squared error (MSE) in the two resulting groups after the split (Breiman et al., 1984).

In addition to this classical *univariate* tree, with hard splits aligned to the axis of the selected variable, there are also models that use the full vector of the variables space, defining a linear discriminant in every split (*multivariate linear*, Murthy et al. (1994); Yildiz & Alpaydın (2000)) and nonlinear splitting surfaces (*multivariate nonlinear*, Guo & Gelfand (1992)). Other strategies include allowing oblique splits, instead of the typical axis-aligned, where the later fail to adequately capture linear relationships (Rainforth & Wood, 2017), and randomizing both the set of variables and the value in every split, regardless of the response (Geurts et al., 2006). In all these works, the approach of hard splits in the observations space can cause problems in the estimation in the context of noisy data, especially on the boundaries of the training set, as demonstrated by Irsoy et al. (2012) and Linero & Yang (2017).

More recently there has been an effort to develop trees that do not do hard assignments of data points to regions, with the introduction of some kind of smoothness in the spliting function. In contrast to the hard split trees, where an observation traverses a single path from the root to one of the terminal nodes, in smoothed trees an observation is linked to all regions, but with different associated degrees of group membership. An observation follows all the paths to all the regions, but with different weights. The final prediction is a combination of the predictions of each node. In essence, while in hard split regression trees there is only one possible final region for a particular observation, in smoothed trees we consider all K regions, but with different degrees of group membership.

Soft trees (Irsoy et al., 2012), Fuzzy trees (Suarez & Lutsko, 2000) and STR-trees (da Rosa et al., 2008) all use a sigmoid-like function in each node, assigning a group membership degree to the observations to be on the left or on the right child node. Soft trees and Fuzzy trees are very similar to each other, with only minor differences. They both are multivariate trees, accessing the whole vector

of variables to establish the membership of observations down the structure of the tree. STR-trees follow the same guidelines but, in contrast, use a single variable in the sigmoid function.

Probabilistic Regression trees (Alkhoury et al., 2020) are another multivariate tree method with a smoothness effort, but it takes a different approach and rely on a parametric probability distribution chosen beforehand via expert knowledge or selected through cross-validation.

In general, trees with soft assignments have the main advantage of providing a smooth response near the boundaries of the training data, where traditional hard split trees present discontinuity. This allows smooth trees to have lower bias in the predictions around the split boundaries (Irsoy et al., 2012). They also lead to fits that are smoother, therefore generalizing better, in the sense that they are less prone to overfit (Yildiz et al., 2016). Smoothness also brings in the possibility of estimating derivatives (such as elasticities) as well as a way of accessing variable importance.

Despite this advantage, smoothed trees are computationally more intensive than traditional trees, and suffer from the same high variance problem that the latter ones. That is, the estimated parameters and predictions of a single tree present high variability and are for the most part dependent on the random subset of data they were fitted for (Wehenkel, 1997; Geurts et al., 2006; Geurts & Wehenkel, 2007).

One of the drawbacks of regression trees is that they are prone to overfitting, which can be thought as the situation where an estimated model fits the characteristics of a particular sample, instead of generalizing to the population structure (Miller et al., 2015; Berk, 2016). It usually is the result of an overly complex model, which leads to high error in the predictions of new data on which the model was not trained on (Breiman, 1996; Friedman, 1997; Dietterich, 2000).

In order to overcome this bias-variance tradeoff in the predictions, we can prune the tree (remove some of the splits near the leafs) after it is fully grown, in order to reduce the complexity of the model (Hastie et al., 2009). Another way to control overfitting is to create an ensemble of trees, in the sense that the final forecast is a combination of several individual predictions. The classical approachs are the Random Forest (Breiman, 2001) (explained in detail in Section 2) and Boosting (Friedman, 2001) (whose exploration is out of the scope of this work).

This work focus on regression problems where the DGP has both linear and

nonlinear components. Our novel approach combines a LASSO linear regression (Tibshirani, 1996), in the first step, with a STR random forest on the residuals (of the first step) in the second step. Therefore, any inner linearity in the data can be captured in the first step, leaving the non-linearities to be depicted by the more flexible model afterwards.

In the simulations, our approach resulted in better out-of-sample predictive performance especially when the data generating process presents both linear and nonlinear terms, with bigger coefficients in the linear terms. In tests with real data, the method performed well against the benchmarks and has established itself as an alternative for regression problems.

This paper is organized as follows: i) first we make a brief revision of the methods in which we build upon (Section 2), namely the traditional CART decision tree, the Tree-Structured Smoothed Transition Regression (STR-Tree), the Random Forest and the LASSO family of regularization methods; ii) we present our method, combining LASSO-type methods with a STR random forest (Section 3), followed by numerical examples both with simulated and empirical data, and comparisson of our approach with other benchmarks (Section 4); iii) we present our conclusions and provide suggestions for future related work (Section 5).

2. TREE-BASED AND SHRINKAGE METHODS

2.1. Regression trees

A standard regression tree is a non-parametric model whose estimation is obtained by recursivelly partitioning the space of covariates, aiming to approximate the unknown data generating function (Breiman et al., 1984). It accomplishes that building a structure that resembles an inverted tree: starting from the root, or the initial node, the model searchs the variable s_j and the value of that particular variable c_j that generates the (locally) optimal spliting of the data. In the context of regression, this can be represented by the lowest combined MSE (mean squared error) of the two groups that resulted from the first split (Hastie et al., 2009). After this first division, two new groups are generated, each one containing the observations that met the criteria defined in the first split: the ones in which the value of the variable s_j is less than or equal to the value c_j are attributed to the left child node, and observations in which the value of the variable s_j is greater than the value c_j are attributed to the right child node. This process continues until some stop criteria is met,

usually a minimum number of observations on the terminal node or a negligible improvement of the MSE in a new division of the tree.

More generally, we can formally describe a regression tree as follows. Let

$$\boldsymbol{x_i} = (x_{i1}, \ldots, x_{im}) \in \mathbb{R}^m \text{ for } i = 1, 2, ..., n, \text{aleatory vector of covariates}$$

and

$$y_i \in \mathbb{R} \text{ the response variable}$$

such that

$$y_i = f(\boldsymbol{x_i}) + \varepsilon_i, \text{ where } \mathbb{E}\big[\varepsilon_i | \boldsymbol{X}_j\big] = 0.$$

Then

$$\hat{f}(\boldsymbol{x_i}) = \sum_{k=1}^{K} \hat{\beta}_k \mathbb{I}_k(\boldsymbol{x_i} \in R_k)\,,$$

where

- K is the number of terminal nodes or regions;
- $\mathbb{I}_k(\boldsymbol{x_i} \in R_k)$ is the indicator function, returning 1 if the observation $\boldsymbol{x_i}$ is in the region R_k, and 0 otherwise;
- $\hat{\beta}_k$ is sample mean of the observations in the kth region.

In the case of a simple tree, with only one splitting node (the root), depth equal to one and two terminal nodes (or leafs), the estimated model is represented by:

$$\hat{y}_i = \hat{\beta}_1 \mathbb{I}(\boldsymbol{x_i}; s_0, c_0) + \hat{\beta}_2 [1 - \mathbb{I}(\boldsymbol{x_i}; s_0, c_0)]\,, \tag{1}$$

where

- s_0 is the variable used in the root node;
- c_0 is the value used in the root node;
- $\hat{\beta}_1$ and $\hat{\beta}_2$ are the response sample mean for the observations in regions $k = 1$ and $k = 2$.

The estimation of the model usually is done by exaustive search of the parameters in each node in a way that minimizes the residual sum of squares. In the example of our simple tree it means finding the variable s_0 and value c_0 which minimize the combined MSE of the two terminal regions $k = 1$ and $k = 2$:

$$(s_0, c_0) = \arg\min_{s,c} \left\{ \sum_{i:\boldsymbol{x_i} \in k=1} (y_i - \hat{y}_i(s,c))^2 + \sum_{i:\boldsymbol{x_i} \in k=2} (y_i - \hat{y}_i(s,c))^2 \right\},$$

where $\hat{y}_i$ is represented by equation 1. In a deeper tree, this process is reapeted recursivelly and with every division a lower number of observations reach the nodes that are more distant from the root, until a minimum threshold is met, when the growth stops.

More generally, a regression tree can be expressed by the following notation, which we borrow from da Rosa et al. (2008) and Fonseca et al. (2020), summarized in Table 1.

Table 1: Identifying nodes in a regression tree

Type of node	Set of nodes	Quantity of nodes	Index of a node
Internal (or parent)	$\mathbb{J}$	J	j
Terminal (or leaf or region)	$\mathbb{K}$	K	k

Then $f(\boldsymbol{x}; \boldsymbol{\theta})$ is the additive model representing the tree, where $\boldsymbol{\theta}$ is the vector of parameters. More especifically:

$$\hat{y}_i = \hat{f}(\boldsymbol{x_i}; \hat{\boldsymbol{\theta}}) = \sum_{k \in \mathbb{K}} \hat{\beta}_k B_{\mathbb{J}k}(\boldsymbol{x_i}; \hat{\boldsymbol{\theta}}_k), \tag{2}$$

where the function $B_{\mathbb{J}k}(\cdot)$ is defined as

$$B_{\mathbb{J}k}(\boldsymbol{x_i}; \boldsymbol{\theta_k}) = \prod_{j \in \mathbb{J}} \mathbb{I}(x_{sj,i}; c_j)^{\frac{n_{kj}(1+n_{kj})}{2}} [1 - \mathbb{I}(x_{sj,i}; c_j)^{(1-n_{kj})(1+n_{kj})}] \tag{3}$$

and

$$\mathbb{I}(x_{sj,i}; c_j) = \begin{cases} 1, & \text{if } x_{sj,i} \leq c_j \\ 0, & \text{otherwise} \end{cases},$$

with

$$n_{kj} = \begin{cases} -1, & \text{if the path to leaf } k \text{ does not include the parent node } j \\ 0, & \text{if the path to leaf } k \text{ include the right-hand child of the parent node } j \\ 1, & \text{if the path to leaf } k \text{ include the left-hand child of the parent node } j \end{cases},$$

where

$$0 \leq B_{\mathbb{J}k}(\boldsymbol{x_i}; \boldsymbol{\theta}_k) \leq 1\,,$$
$$\sum_{k \in \mathbb{K}} B_{\mathbb{J}k}(\boldsymbol{x_i}; \boldsymbol{\theta}_k) = 1\,,$$

with $\boldsymbol{\theta}$ being the vector of parameters in the form

$$\boldsymbol{\theta} = (\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_K),$$

and finally $\boldsymbol{\theta}_k$ is the set of values for every node:

$$\boldsymbol{\theta_k} = \{c_j\}, j \in \mathbb{J}, k \in \mathbb{K}\,.$$

Note that the functions $B(\cdot)$ return one for the complete path, from the root to the terminal node, where a particular observation $\boldsymbol{x_i}$ was fitted, and zero for all other possible paths. The expoents in equation 3 always return either one or zero, depending on the path took by the observation $\boldsymbol{x_i}$ down the structure of the tree. Thus, equation 3, being a product of indicator functions, represents the set of nodes traversed by a particular observation. This way we can fully specify any tree structure.

2.2. STR-Tree model

In da Rosa et al. (2008), the regression tree is modified by replacing the indicator function by a logistic one, defined as follows:

$$L(x_{s_j,i}; \gamma_j, c_j) = \frac{1}{1 + e^{-\gamma_j (x_{s_j,i} - c_j)}}\,, \tag{4}$$

where

- γ_j is the smoothness parameter;
- c_j is the location parameter, similar to the splitting value of a traditional tree.

The parameters γ_j and c_j define the smoothness of the transition close to each tree node. If γ_j if large, $L(\cdot)$ behaves like an indicator function, and if γ_j is small, $L(\cdot)$ becomes linear.

The STR-Tree model can then be represented by equation 2 with $B(\cdot)$ re-

placed by

$$B_{\mathbb{J}k}(\boldsymbol{x_i};\boldsymbol{\theta}_{kS}) = \prod_{j\in\mathbb{J}} L(x_{s_j,i};\gamma_j,c_j)^{\frac{n_{kj}(1+n_{kj})}{2}}\left[1-L(x_{s_j,i};\gamma_j,c_j)^{(1-n_{kj})(1+n_{kj})}\right], \tag{5}$$

where $L(\cdot)$ is the logistic function defined in equation 4 and $\boldsymbol{\theta}_{kS}$ is the set of variable, smoothness and location parameters s, γ and c for each node j:

$$\boldsymbol{\theta}_{kS} = \{s_j,\gamma_j,c_j\}, j \in \mathbb{J}_k.$$

Note that now we do not have sharp splits in the nodes of the tree anymore. For every region k we have a value in the $[0,1]$ interval for a particular observation, meaning the degree of that observation belonging to that region. In contrast to the traditional tree, where an observation follows only one path to the bottom and ignores the others, in STR-tree every observation belongs to all the terminal regions, but with different degrees of membership.

Growing a tree means to find, for each node, the transition variable and to estimate its respective smoothness and location parameters. This is done by minimizing the sum of the squared errors conditional on the structure of the tree until that particular node. If we have a grown tree, the set of estimated parameters $\boldsymbol{\delta}$ of a new split is

$$\hat{\boldsymbol{\delta}} = (j,\hat{s}_j,\hat{\gamma}_j,\hat{c}_j,\hat{\beta}_{2j+1},\hat{\beta}_{2j+2}) = \arg\min_{\boldsymbol{\delta}} \sum_{i=1}^{n}\left[y_i - \hat{z}(\boldsymbol{x_i},\boldsymbol{\delta})\right]^2,$$

where

- j is the index which identify a particular node;
- $\hat{s}_j$ is the transition variable for node j;
- $\hat{\gamma}_j$ is the smoothness parameter for node j;
- $\hat{c}_j$ is the location parameter for node j;
- $\hat{\beta}_{2j+1}$ is the coefficient for the left child node of this new split;
- $\hat{\beta}_{2j+2}$ is the coefficient for the right child node of this new split;

and

$$\begin{aligned}\hat{z}(\boldsymbol{x_i},\boldsymbol{\delta}) = &\sum_{k\in\mathbb{K},k\neq j}\hat{\beta}_k B_{\mathbb{J}k}(\boldsymbol{x_i};\hat{\boldsymbol{\theta}}_{kS})\\ &+\hat{\beta}_{2j+1}L(x_{sj,i};\gamma_j,c_j)B_{\mathbb{J}j}(\boldsymbol{x_i};\hat{\boldsymbol{\theta}}_{jS})\\ &+\hat{\beta}_{2j+2}\left[1-L(x_{sj,i};\gamma_j,c_j)\right]B_{\mathbb{J}j}(\boldsymbol{x_i};\hat{\boldsymbol{\theta}}_{jS}).\end{aligned}$$

The expression above means that, for every split one must choose the particular node j to split, the variable s_j and the γ_j, c_j parameters that minimize the squared error considering the actual structure of the tree until that point and the two new child nodes generated by this new split.

Algorithm 1, from Fonseca et al. (2020), describes the overall procedure of growing a STR-Tree.

Algorithm 1 Growing a STR-Tree

Data: $\{x_i\}_{i=1}^n, \{y_i\}_{i=1}^n$
Input: $\eta = 0, \mathbb{J} = \varnothing, \mathbb{K} = \varnothing$ and K
Output: $\{\beta_k\}_{k\in\mathbb{K}}, \{\theta_k\}_{k\in\mathbb{K}}, \{\hat{y}_i\}_{i=1}^n$
  while $\eta < K$ **do**
    if $\eta = 0$ **then**
      Find the index of the best splitting variable, s_0 and threshold c_0;
      Compute $L(\mathbf{x}_{i,s_0}, \gamma_0, c_0)$;
      Compute $B_{\mathbb{J}k}(\mathbf{x}_i; \theta_k)$, for $k = 1, 2$;
      Compute β_k, for $k = 1, 2$;
      Set $\mathbb{J} = 0$ and $\mathbb{K} = \{1, 2\}$;
    else
      Find the node to split, $j \in \mathbb{K}$, the index of the best splitting variable, s_j, the smoothness parameter, γ_j, and splitting threshold c_j;
      Compute $L(\mathbf{x}_{i,s_j}, \gamma_j, c_j)$, $B_{\mathbb{J}k}(\mathbf{x}_i; \theta_k)$, for $k = 2j + 1, k = 2j + 2$;
      Compute β_k, for $k = 2j + 1, k = 2j + 2$;
      Update $\mathbb{J}$ and $\mathbb{K}$;
    end if
    Update $\eta = \eta + 1$;
  end while

2.3. Ensembling trees

Ho (1995) shows that combining several trees improves predictive performance while simultaneously providing regularization. This way, there is no need to use pruning to attack the overfitting problem, we can fully grow trees and average their predictions. Every tree is fitted individually and the final estimator is an average of all individual models, resulting in a predictor with better performance than any of the individual trees (Rokach, 2010).

Since the growth of a traditional individual tree follows a deterministic procedure, in order to use a combination of them it is necessary to introduce some kind of randomness, enabling the creation of several slightly different trees. Ho (1998) accomplished that via randomly selecting the subset of candidate covariates in each splitting node (the random subspace method).

Breiman (1996) takes another approach and employs bagging, short for bootstrap aggregation, which is a procedure that trains several simple models, each one with slightly different data through bootstrap sampling (Efron, 1977) the original dataset. The final estimator is an average from all trees, as follows:

$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x)\,,$$

where B is the number of individual models to be trained. This increases the stability of the model, i.e., decreases its variance, easying the impact of extreme values on the data. We can think of this averaging of predictions as a way of imposing a regularizing effect on the model (Yildiz et al., 2016).

Random forest, introduced by Breiman (2001), combines the ideas behind bagging and the random subspace method. In this new perspective, the growing of several individual trees has two sources of randomization: the first one is the bootstrap sampling of the training set, where sometimes an observation will be drawn more than once and sometimes will not be drawn at all, whereas the second source is the random selection of the subset of candidate regressors in each split of the trees. This results in trees that are less correlated with each other, and, in a final estimator, averaged from the trees, with better predictive performance and less variance than either individual approaches (Hastie et al., 2009; Berk, 2016).

The insight behind the good predictions that random forests provide is that, in a forest, the effect of any individual tree that overfits because of noisy training observations is dilluted in the averaging process. The individual trees will be slightly different from each other, therefore averaging them improves predicive performance in comparisson with any individual tree. Breiman (2001) presents theoretical results showing that the lower the correlations among trees, the lower the upper bound for the error of the ensemble. So it is critical to achieve diversity in the individual components that make a forest of trees. Typically any particular tree will have suboptimal splits, because the parameters are estimated locally, i.e., the whole tree structure is not considered at each node, only the data that reached that point. A node is not "aware" of further splits and their implications on the parameters estimation. Thus any individual tree is almost never the global optimal solution over all possible models, but an ensemble of those *weak learners* improves the overall predictive performance (Strobl et al., 2009).

As a complex method, with several degrees of randomization process, it is naturally hard to explain matematically the mecanics of how and why random forest works, besides intuitive explanations and simulation studies. Biau et al. (2008) and Lin & Jeon (2006) explored this issue, although imposing several simplifying assumptions, due to the inner complexity of the ensembling procedure.

Finally, another way of ensembling trees is the boosting procedure, which was introduced by Friedman (2001) and is significantly different from the random forest. In boosting, trees are trained sequentially, where the next tree is fitted with greater weights on the observations that were poorly fitted in the previous step and lower weights on the ones that had a good fit. This *boosts* the performance of the model by forcing the trees to choose parameters that approximate better the observations that previous trees had difficulties dealing with (Miller et al., 2015). The final estimator is an additive model of all individual trees (Awaya & Ma, 2021). In the case of STR-Trees, the boosting approach for regression was stablished by the "BooST" method (Fonseca et al., 2020), which we use as one of the benchmarks in the simulation study (Section 4). Exploring the details of boosting is out of the scope of this work.

2.4. Shrinkage methods and variable selection

Variable selection is important in modern statistical modeling, especially with the very large sizes of datasets available in recent years, both in terms of number of observations and in quantity of variables. Traditional methods like best subset or stepwise selection have their own difficulties dealing with a high number of regressors, with reduced accuracy of forecasts or large computational cost, which can make the estimation infeasible (Hastie et al., 2009).

Another approach is regularization, with methods like Ridge regression (Hoerl & Kennard, 1970) and Least Absolute Shrinkage and Selection Operator (LASSO) (Tibshirani, 1996). These procedures employ a penalization on the coefficients of a model, and shrink the magnitude of some of them, hence being known as shrinkage methods. While Ridge applies a ℓ_2-penalty on the coefficients estimation, resulting in lower βs than those estimated by OLS, LASSO imposes a ℓ_1-penalty and sets some of them to zero, effectively performing simultaneously parameter estimation and variable selection.

Consider the following linear model:

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_p x_{pi} + \varepsilon_i \,, \tag{6}$$

where we have p predictors, $i = 1, \ldots, n$ is the index of a particular observation and can also be writen in matrix notation as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}.$$

Then the LASSO estimator is obtained by the following optimization problem:

$$\hat{\boldsymbol{\beta}}_{\text{LASSO}} = \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ji} \right)^2 \right\}$$

subject to

$$\sum_{j=1}^{p} |\beta_j| \leq t\,,$$

that can also be writen in the equivalent following way:

$$\hat{\boldsymbol{\beta}}_{\text{LASSO}} = \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ji} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\},$$

where λ is the parameter that controls the amount of shrinkage (or penalty).

Zhao & Yu (2006) demonstrated that the ordinary LASSO regression do not have consistency in selecting the relevant variables and the true coefficients when the size of training set increases, and also do not have the *oracle* property (the estimator identifies the true model and has the optimal estimation rate) (Fan & Li, 2001; Zou, 2006). Zou (2006) proposes the Adaptive LASSO (adaLASSO), which presents the *oracle* property and is asymptotically unbiased. This is done through the addition of different weights for each variable in the penalty factor:

$$\hat{\boldsymbol{\beta}}_{\text{adaLASSO}} = \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ji} \right)^2 + \lambda \sum_{j=1}^{p} \omega_j |\hat{\beta}_j| \right\},$$

where $\omega_j = |\hat{\beta}_j|^{-\tau}$, $\tau > 0$, are the coefficients obtained in a previous step via OLS or even through a standard LASSO regression. Hence, differente variables coefficients have different weights in this optimization problem.

Notice that nevertheless these LASSO-type approaches present the property of automatic variable selection, they are still linear methods, and capture only the linear relationships in the data. If the data has nonlinear (or both linear

and nonlinear) generating terms, neither LASSO nor adaLASSO will be a good solution, and for this reason we propose a combination of those methods with a more flexible, non-parametric approach, in the next section.

3. COMBINING LASSO-TYPE METHODS WITH A SMOOTH TRANSITION RANDOM FOREST

The idea behind the proposed method is to capture the linear relationships in the data, if any, in a parametric first step with the adaLASSO regression and let the flexibility of a STR random forest accesss the non-linearities in the second step. The ensemble of STR-Trees tend to produce better predictions near the boundaries of the training set due to its smoothness, compared to the traditional regression tree alternative, so we expect lower out-of-sample errors in some cases. Furthermore, in the context of sparse data with irrelevant variables, the adaLASSO fitting in the first step can act as a "filter", capturing most of the linearities, and letting an easier task to be completed by the second step non-parametric regression (STR random forest) in attacking the nonlinearities.

Our method is a two step procedure described as bellow:

1. Fit an adaLASSO regression on the data with the response y, as defined by model 6, generating the prediction $\hat{y}_i^{\text{1st}}$, for $i = 1, \ldots, n$;
2. Calculate the residuals of the previous regression: $\hat{\varepsilon}_i^{\text{1st}} = y_i - \hat{y}_i^{\text{1st}}$, for $i = 1, \ldots, n$;
3. Draw B bootstrap samples of the data;
4. Fit B STR-trees (using only a fraction of the predictors in each split of the tree), on each sample, with the response variable being the residuals from the previous step ($\varepsilon_i^{\text{1st}}$), generating the prediction $\hat{\varepsilon}_i^{\text{2nd}}$, for $i = 1, \ldots, n$;
5. Average the B predictions of the individual trees to have the random forest forecast: $\hat{\varepsilon}_{i\ \text{RF}}^{\text{2nd}} = \frac{1}{B} \sum_i^n \hat{\varepsilon}_i^{\text{2nd}}$;
6. Sum both intermediary predictions to have the final one: $\hat{y}_i = \hat{y}_i^{\text{1st}} + \hat{\varepsilon}_{i\ \text{RF}}^{\text{2nd}}$, for $i = 1, \ldots, n$.

Our method have shown advantages in terms of predictive performance in some cases in contrast to other benchmarks, as shown in the next section, especialy if the data generating process presents both linear and nonlinear components.

4. NUMERICAL EXAMPLES

4.1. Settings

As the first step of our method involves an adaLASSO regression, we are naturally interested in evaluate its variable selection capabilities, in scenarios where the data include linear and nonlinear terms. In order to access the ability of the method in capturing the linearity in the data, we follow the works of Medeiros & Mendes (2015) and Konzen & Ziegelmann (2016) and use the variable selection evaluation metrics described in Table 2.

Table 2: Variable selection metrics

Metric	Description
FVCI	Fraction of variables correctly identified. It is the sum of the number of relevant variables included and the number of irrelevant variables excluded, divided by the total number of variables, in every replication, averaged across all replications;
TMI	True model included. The fraction of replications in that all relevant variables were correctly included;
FRVI	Fraction of relevant variables included. For each replication, the fraction of relevant variables that were correctly included, averaged across all replications;
FIVE	Fraction of irrelevant variables excluded. For each replication, the fraction of irrelevant variables that were correctly excluded, averaged across all replications.

In the simulations, we calculated these metrics after the first step of the method, i.e., after the estimation of the adaLASSO coefficients. Specifically for the FRVI metric, we make the following segmentation: first we consider all relevant variables (FRVI all variables), and then compute it only for the variables generated linearly (FRVI linear variables) and finally for the variables generated from nonlinear functions (FRVI nonlinear variables).

In Section 4.2 we performed simulation studies, with datasets generated with sample size $n = (50, 200, 1000)$ and subsequently fitted by all regression methods in 500 replications on a training partition with size $\frac{2}{3}n$, randomly choosen each time, and we evaluate its performance on the remaining test partition (with size $\frac{1}{3}n$).

We use the Root Mean Squared Relative Errors (RMSRE[1]) of the out-of-sample predictions as the metric to evaluate the models. This is because we want to access the predictive performance with different sets of coefficients for the linear terms, and simultaneously be able to compare the errors with slightly different DGPs.

In the case of the real datasets (Section 4.3), where we do not know the true DGP, the data remained the same across the replications, but the training and test sets of observations were choosen randomly each time. The performance metric choosen was the Root Mean Squared Error (RMSE) of the out-of-sample predictions.

The regression methods used as benchmarks are listed in Table 3. Even though OLS and (standalone) adaLASSO are not equiped to deal with non-linearities, we opted to keep them in the simulations for comparisson purposes.

Table 3: Regression methods used as benchmarks

Classical	Tree-based	Proposed methods
Ordinary Least Squares (OLS)	Traditional Random Forest (RF)	adaLASSO + RF
Adaptive LASSO (adaLASSO)	STR-Tree Random Forest (STR RF)	adaLASSO + STR RF
Support Vector Regression (SVR)	STR-Tree Boosting (BooST)[2]	

Usually the estimation of the parameters of tree-based methods are not affected by the magnitude of the variables, but for other methods it is normally a problem if the variables have big differences in scale (Hastie et al., 2009). So in order to be able to compare the errors of different regression models, we pre-process the X matrix standardizing each variable before the fittings.

In respect to the number of STR-trees used in the random forest, we ran a simulation on the "example" DGP generated in Section 4.2, where we measured the MSE of the out of sample prediction for different numbers of smooth trees in the random forest ensemble. We observed that the prediction error keeps falling when more trees are added into the random forest, plateauing at about 200 trees onwards. So we choose to use 200 trees in the random forest of our simulations, since only modest gains in predictive performance are obtained past that point, and also not to create unnecessary computational burden.

[1] $\text{RMSRE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\hat{y}_i - y_i}{y_i}\right)^2}$, Göçken et al. (2016).

[2] Fonseca et al. (2020).

All models were fitted using the default parameters of their respective implementations[3]. Our implementation can be found at the authors repository[4].

4.2. Monte Carlo studies

The idea behind this section is to test the predictive performance of our method against simulated datasets in which we control the magnitude of the linear features while constructing the response. This is done varying the coefficients of the linear terms in the additive model, attributing more (or less) importance to the linear features, in contrast to the nonlinear variables.

We also include irrelevant variables, which do not contribute to the response y, to access whether the feature selection in the first step of the model contributes to an overall better fitting or not. We start with the "example" DGP from Table 4.

Table 4: Example DGP

DGP	Variables
$y = a\,x_1 + b\,x_2 + c\,x_3 + 3\,sin(x_4) + 3\,e^{-x_5^2} + \sqrt{x_6} + \varepsilon$	$x_1, \ldots, x_{50} \sim N(\frac{\pi}{2}, \frac{1}{2})$ $\rho_{x_i, x_j} = 0.85\ \forall\ i \neq j$ $\varepsilon \sim N_{iid}(0, \frac{1}{4})$

The mean and variance of the normally distributed variables were choosen in order to not generate a response y with values near zero, which would yield irrealistic measures of relative error, since its calculation involves dividing the error by the true response. Note that 44 variables (x_7 to x_{50}) are non-informative, they do not contribute to the response y, but nevertheless all 50 variables were used in the fittings.

The out-of-sample prediction RMSRE for the "example" DGP, for various values of the (a, b, c) coefficients is shown in Table 5.

[3]We use the Scikit-Learn library implementations (https://scikit-learn.org/), except for the BooST model https://github.com/gabrielrvsc/BooST.

[4]https://github.com/alexgand/adalasso_STR_RF. The code for the STR-tree model was partially modified from the BooST repository.

Table 5: Test RMSRE for the example DGP from Table 4

Size	(a, b, c) coefficients	OLS	adaLASSO	SVR	RF	STR RF	BooST	adaLASSO + RF	adaLASSO + STR RF
50	$(\frac{1}{2},\frac{1}{2},\frac{1}{2})$	0.1126	0.0846	0.0749	0.0715	0.0669	0.0783	**0.0717**	0.0796
	$(\frac{1}{2},\frac{1}{2},1)$	0.1033	0.0795	0.0769	0.0692	0.0827	0.0744	**0.0670**	0.0724
	$(\frac{1}{2},1,1)$	0.0945	0.0743	0.0792	0.0673	0.0898	0.0688	**0.0627**	0.0680
	(1,1,1)	0.0878	0.0707	0.0829	0.0668	0.1001	0.0647	**0.0593**	0.0636
	(1,1,2)	0.0794	0.0615	0.0948	0.0688	0.1195	0.0600	**0.0518**	0.0542
	(1,2,2)	0.0721	0.0550	0.1060	0.0709	0.1334	0.0569	**0.0459**	0.0496
	(2,2,2)	0.0666	0.0504	0.1168	0.0736	0.1443	0.0566	**0.0419**	0.0437
	(2,2,3)	0.0642	0.0464	0.1302	0.0765	0.1573	0.0537	**0.0382**	0.0441
	(2,3,3)	0.0607	0.0423	0.1369	0.0773	0.1667	0.0505	**0.0352**	0.0378
	(3,3,3)	0.0586	0.0395	0.1518	0.0816	0.1698	0.0540	**0.0323**	0.0356
200	$(\frac{1}{2},\frac{1}{2},\frac{1}{2})$	0.0865	0.0829	0.0701	0.0670	0.0700	0.0575	0.0635	**0.0569**
	$(\frac{1}{2},\frac{1}{2},1)$	0.0781	0.0754	0.0698	0.0646	0.0854	0.0529	0.0578	**0.0518**
	$(\frac{1}{2},1,1)$	0.0712	0.0694	0.0701	0.0623	0.1002	0.0496	0.0531	**0.0480**
	(1,1,1)	0.0657	0.0642	0.0718	0.0612	0.1121	0.0458	0.0491	**0.0444**
	(1,1,2)	0.0572	0.0566	0.0801	0.0597	0.1346	0.0416	0.0429	**0.0383**
	(1,2,2)	0.0507	0.0508	0.0874	0.0600	0.1517	0.0390	0.0383	**0.0346**
	(2,2,2)	0.0457	0.0462	0.0959	0.0617	0.1650	0.0363	0.0346	**0.0309**
	(2,2,3)	0.0416	0.0419	0.1062	0.0624	0.1770	0.0341	0.0315	**0.0284**
	(2,3,3)	0.0381	0.0387	0.1146	0.0637	0.1871	0.0327	0.0289	**0.0271**
	(3,3,3)	0.0354	0.0363	0.1238	0.0657	0.1953	0.0315	0.0269	**0.0242**
1000	$(\frac{1}{2},\frac{1}{2},\frac{1}{2})$	0.0736	0.0758	0.0565	0.0540	0.0755	**0.0450**	0.0481	0.0457
	$(\frac{1}{2},\frac{1}{2},1)$	0.0662	0.0684	0.0541	0.0523	0.0914	**0.0408**	0.0435	0.0411
	$(\frac{1}{2},1,1)$	0.0603	0.0626	0.0527	0.0507	0.1060	0.0374	0.0399	**0.0372**
	(1,1,1)	0.0556	0.0580	0.0524	0.0497	0.1189	0.0346	0.0368	**0.0345**
	(1,1,2)	0.0485	0.0507	0.0546	0.0475	0.1428	0.0304	0.0320	**0.0297**
	(1,2,2)	0.0431	0.0454	0.0574	0.0459	0.1610	0.0275	0.0285	**0.0264**
	(2,2,2)	0.0388	0.0410	0.0610	0.0452	0.1759	0.0253	0.0257	**0.0239**
	(2,2,3)	0.0355	0.0376	0.0658	0.045	0.1891	0.0231	0.0234	**0.0215**
	(2,3,3)	0.0326	0.0347	0.0706	0.0443	0.2004	0.0219	0.0215	**0.0197**
	(3,3,3)	0.0303	0.0324	0.0754	0.0445	0.2097	0.0208	0.0199	**0.0184**

First we can confirm our intuition in the sense that, the greater the influence of the linear terms in y, the lower the error of the proposed adaLASSO+STR RF method. This behaviour is present for all sample sizes and is also true for the adaLASSO+RF and BooST methods. Naturally OLS and standalone adaLASSO also benefit from greater linear features.

For small sized data, with $n = 50$, the method with the lower overall errors was the adaLASSO+RF and for $n = 200$ the best method was the adaLASSO+STR RF. For a larger dataset, with $n = 1000$, BooST performed better for lower values of the (a, b, c) coefficients, but the higher the magnitude of the linear terms, the less it surpassed the adaLASSO+STR RF method. For high values of (a, b, c), the adaLASSO+STR RF presented the lowest errors.

It is worth mentioning that, with the implementations used in this study, the BooST algorithm was the slowest, approximately five to ten times slower than the adaLASSO+STR RF one. Our implementation takes advantage of the parallel computing when training the random forest trees, while boosting is intrinsically a sequential process. In real world problems where speed is paramount, it may be worth to give up a little gain in reducing the predictive error in order to have a "good enough" fast answer.

In terms of selection of the correct variables in the adaLASSO step of our method, we can see in Table 6 that the higher the magnitude of the (a, b, c) coefficients, the more the relevant variables were correctly included in the model (FVCI and FRVI metrics), for all sample sizes. The true model (TMI) was almost never fully identified in this first step, although the linear variables were easily correctly included, even with a low n. Finaly, the metric for the correctly excluded variables (FIVE) remained relatively stable across all combination of the linear coefficients.

Table 6: adaLASSO variable selection metrics for the example DGP from Table 4

Size	(a, b, c) coefficients	FVCI	TMI	FRVI all vars.	FRVI linear vars.	FRVI non-linear vars.	FIVE
50	$(\frac{1}{2},\frac{1}{2},\frac{1}{2})$	0.8499	0.0	0.1808	0.2942	0.0675	0.9411
	$(\frac{1}{2},\frac{1}{2},1)$	0.8503	0.0	0.2922	0.5062	0.0782	0.9264
	$(\frac{1}{2},1,1)$	0.8486	0.0	0.3701	0.6625	0.0778	0.9138
	(1,1,1)	0.8537	0.0	0.4415	0.7975	0.0854	0.9099
	(1,1,2)	0.8560	0.0	0.4759	0.8609	0.0909	0.9078
	(1,2,2)	0.8592	0.0	0.5034	0.9273	0.0796	0.9077
	(2,2,2)	0.8624	0.0	0.5396	0.9945	0.0847	0.9064
	(2,2,3)	0.8637	0.0	0.5399	0.9972	0.0826	0.9078
	(2,3,3)	0.8609	0.0	0.5353	0.9961	0.0745	0.9053
	(3,3,3)	0.8659	0.0	0.5395	1.0	0.0790	0.9104
200	$(\frac{1}{2},\frac{1}{2},\frac{1}{2})$	0.8863	0.0	0.4171	0.7587	0.0755	0.9503
	$(\frac{1}{2},\frac{1}{2},1)$	0.8896	0.0	0.4769	0.8685	0.0852	0.9458
	$(\frac{1}{2},1,1)$	0.8863	0.0056	0.5046	0.9204	0.0889	0.9384
	(1,1,1)	0.8866	0.0055	0.5497	1.0	0.0994	0.9326
	(1,1,2)	0.8902	0.0	0.5493	1.0	0.0987	0.9366
	(1,2,2)	0.8913	0.0056	0.5456	1.0	0.0912	0.9384
	(2,2,2)	0.8899	0.0	0.5496	1.0	0.0993	0.9363
	(2,2,3)	0.8892	0.0	0.5481	1.0	0.0963	0.9357
	(2,3,3)	0.8913	0.0	0.5506	1.0	0.1013	0.9377
	(3,3,3)	0.8923	0.0	0.5454	1.0	0.0907	0.9396
1000	$(\frac{1}{2},\frac{1}{2},\frac{1}{2})$	0.9275	0.0	0.5984	1.0	0.1968	0.9724
	$(\frac{1}{2},\frac{1}{2},1)$	0.9258	0.0	0.6092	1.0	0.2183	0.9690
	$(\frac{1}{2},1,1)$	0.9272	0.0	0.6158	1.0	0.2317	0.9697
	(1,1,1)	0.9268	0.0	0.6115	1.0	0.2230	0.9698
	(1,1,2)	0.9269	0.0	0.6115	1.0	0.2230	0.9699
	(1,2,2)	0.9272	0.0	0.6056	1.0	0.2113	0.9710
	(2,2,2)	0.9257	0.0	0.6087	1.0	0.2175	0.9689
	(2,2,3)	0.9268	0.0	0.6080	1.0	0.2160	0.9702
	(2,3,3)	0.9273	0.0	0.6092	1.0	0.2183	0.9707
	(3,3,3)	0.9276	0.0	0.5986	1.0	0.1972	0.9725

Now we pass to analyse a more challenging dataset, introduced by Friedman (1991), as shown in Table 7.

Table 7: Modified Friedman #1 dataset

DGP	Variables
$y = 10\,sin(\pi x_1 x_2) + 20\,(x_3 - 0.5)^2 + a\,x_4 + b\,x_5 + \varepsilon$	$x_1, \ldots, x_{50} \sim U_{iid}(0,1)$ $\varepsilon \sim N_{iid}(0,1)$

Friedman used $a = 10$ and $b = 5$ in the original paper. We appropriate

these (a, b) coefficients to do the same analysis of the previous "example" DGP, varying their values in order to access the variable selection and predictive results of our method. Here again we have irrelevant variables (x_6 to x_{50}) which do not contribute to the response.

Table 8 presents the test RMSRE for sample sizes $n = (50, 200, 1000)$ and for various values of the (a, b) coefficients for the modified Friedman #1 DGP.

Table 8: Test RMSRE for the Friedman #1 DGP (Table 7)

Size	(a, b) coefficients	OLS	adaLASSO	SVR	RF	STR RF	BooST	adaLASSO + RF	adaLASSO + STR RF
50	(1,1)	2.1265	2.0558	2.3674	1.9421	**0.7752**	1.5565	1.8200	1.7006
	(5,1)	1.0260	1.1102	1.3255	1.1024	**0.7518**	0.8515	0.9930	0.9367
	(5,5)	0.6057	0.5729	0.6724	0.5738	**0.3837**	0.4494	0.5222	0.5061
	(10,5)	0.8741	0.8464	1.0848	0.9534	**0.3472**	0.7413	0.8011	0.7576
	(15,10)	0.4776	0.4057	0.6715	0.5313	**0.3177**	0.3957	0.3902	0.3833
	(25,20)	0.3723	0.2160	0.6069	0.4087	0.3441	0.2458	0.2043	**0.2003**
	(35,30)	0.3622	0.1667	0.6843	0.4298	0.3808	0.2266	0.1562	**0.1532**
	(45,40)	0.3690	0.1334	0.7256	0.4442	0.4047	0.2030	0.1251	**0.1206**
	(55,50)	0.3674	0.1096	0.6867	0.4106	0.4271	0.2035	0.1025	**0.1018**
200	(1,1)	1.8421	2.3409	3.0613	1.7931	1.9126	**1.0529**	1.4596	1.2401
	(5,1)	0.8414	1.0068	1.4857	0.9001	0.5894	**0.4813**	0.6474	0.5960
	(5,5)	0.5601	0.6734	1.0568	0.7247	0.7431	**0.3960**	0.4899	0.4413
	(10,5)	0.4055	0.4563	0.8409	0.5127	0.6342	**0.2739**	0.3354	0.3036
	(15,10)	0.2390	0.2611	0.6427	0.3651	0.4251	**0.1690**	0.1912	0.1798
	(25,20)	0.1556	0.1661	0.6649	0.2842	0.4324	0.1244	0.1237	**0.1159**
	(35,30)	0.1183	0.1242	0.7224	0.2542	0.4645	0.1081	0.0938	**0.0881**
	(45,40)	0.0963	0.1003	0.7759	0.2422	0.4937	0.0997	0.0756	**0.0712**
	(55,50)	0.0814	0.0845	0.8222	0.2375	0.5162	0.0888	0.0643	**0.0606**
1000	(1,1)	2.2719	2.7768	3.1292	1.6490	2.7814	**0.9946**	1.2736	1.3306
	(5,1)	1.3411	1.7564	2.1321	1.3180	1.4815	**0.6515**	0.8918	0.9506
	(5,5)	0.6645	0.7909	1.0541	0.6581	0.6087	**0.3070**	0.3957	0.4341
	(10,5)	0.3127	0.3663	0.5520	0.3321	0.5114	**0.1533**	0.1917	0.2170
	(15,10)	0.2367	0.2764	0.5835	0.3167	0.4995	**0.1350**	0.1522	0.1649
	(25,20)	0.1396	0.1607	0.5695	0.2217	0.4795	**0.0833**	0.0873	0.0985
	(35,30)	0.1028	0.1147	0.6304	0.1829	0.4907	**0.0625**	0.0648	0.0723
	(45,40)	0.0830	0.0926	0.7016	0.1642	0.5189	0.0546	**0.0527**	0.0592
	(55,50)	0.0701	0.0778	0.7595	0.1501	0.5478	0.0461	**0.0445**	0.0495

Again, we see that for all sample sizes, the higher the (a, b) coefficients, the lower the error for the adaLASSO+STR RF method (also true for adaLASSO+RF, BooST, OLS and adaLASSO), confirming our hypothesis that predictive performance would increase with greater linear terms.

For this dataset, with sample sizes $n = 50$ and $n = 200$, initially BooST had the lowest prediction errors, but as the magnitude of the linear terms grew,

the adaLASSO+STR RF method started to be the best performer. For a large sample size ($n = 1000$), this pattern was once more observed, but the best method for larger linear (a, b) coefficients was the adaLASSO+RF, instead of the adaLASSO+STR RF.

Analysing the variable selection metrics of the adaLASSO step of our method in Table 9, we can see the same patterns explained in the "example" DGP, for instance, the true model (TMI) was almost never selected, the fraction of correctly identified variables grew as the linear coefficients were being increased (FVCI, FRVI), and the proportion of correctly excluded variables (FIVE) was relatively stable across all variations.

Table 9: adaLASSO variable selection metrics for the Friedman #1 DGP (Table 7)

Size	(a, b) coefficients	FVCI	TMI	FRVI all vars.	FRVI linear vars.	FRVI non-linear vars.	FIVE
	(1,1)	0.8946	0.0034	0.3116	0.0582	0.4806	0.9594
	(5,1)	0.8914	0.0237	0.4210	0.3085	0.4960	0.9437
	(5,5)	0.8847	0.0712	0.5119	0.5356	0.4960	0.9262
	(10,5)	0.8845	0.0918	0.6456	0.7738	0.5601	0.9110
50	(15,10)	0.8873	0.1288	0.7390	0.9712	0.5842	0.9037
	(25,20)	0.8880	0.0847	0.7329	1.0	0.5548	0.9052
	(35,30)	0.8925	0.0746	0.7295	1.0	0.5492	0.9106
	(45,40)	0.8965	0.0442	0.7259	1.0	0.5431	0.9155
	(55,50)	0.8975	0.1270	0.7302	1.0	0.5503	0.9160
	(1,1)	0.9338	0.0	0.4213	0.0532	0.6667	0.9907
	(5,1)	0.9439	0.0	0.6243	0.5536	0.6714	0.9794
	(5,5)	0.9514	0.0423	0.8085	1.0	0.6808	0.9673
	(10,5)	0.9506	0.0352	0.8070	1.0	0.6784	0.9665
200	(15,10)	0.9555	0.0210	0.8042	1.0	0.6737	0.9723
	(25,20)	0.9531	0.0420	0.8084	1.0	0.6807	0.9692
	(35,30)	0.9569	0.0211	0.8042	1.0	0.6737	0.9739
	(45,40)	0.9551	0.0282	0.8056	1.0	0.6761	0.9717
	(55,50)	0.9575	0.0210	0.8042	1.0	0.6737	0.9745
	(1,1)	0.9470	0.0	0.4785	0.1963	0.6667	0.9990
	(5,1)	0.9643	0.0	0.6618	0.6544	0.6667	0.9979
	(5,5)	0.9778	0.0146	0.8029	1.0	0.6715	0.9972
	(10,5)	0.9785	0.0	0.8000	1.0	0.6667	0.9984
1000	(15,10)	0.9768	0.0	0.8000	1.0	0.6667	0.9964
	(25,20)	0.9751	0.0	0.8000	1.0	0.6667	0.9946
	(35,30)	0.9770	0.0	0.8000	1.0	0.6667	0.9967
	(45,40)	0.9779	0.0	0.8000	1.0	0.6667	0.9977
	(55,50)	0.9771	0.0	0.8000	1.0	0.6667	0.9967

4.3. Real datasets

We use a selection of real data from a broad range of areas, with diversity both in sample size and in the number of covariates, inspired by the works of Calhoun et al. (2020) and Breiman (2001). They were obtained either from the UCI (Dua & Graff, 2021) and OpenML (Vanschoren et al., 2013) repositories or directly from the Scikit-Learn library (Pedregosa et al., 2011).

All datasets present numeric responses, and the majority of them only contain numeric variables. In the case of the ones that contain categorical or qualitative variables, those have been excluded. None present missing values. In Table 10 we can find some information about the selected real datasets.

Table 10: Information about the real datasets

Name	Size n	Number of variables	Short description from source
Abalone	4,177	7	*Predicting the age of abalone from physical measurements. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope – a boring and time-consuming task.*
Ailerons	13,740	40	*This dataset addresses a control problem, namely flying a F16 aircraft. The attributes describe the status of the aeroplane, while the goal is to predict the control action on the ailerons of the aircraft.*
Bodyfat	252	14	*Lists estimates of the percentage of body fat determined by underwater weighing and various body circumference measurements for 252 men.*
Debutanizer	2,394	7	*Temperature and pressure measures of butane from the debutanizer process.*
Diabetes	442	10	*Ten baseline variables were obtained for each of diabetes patients, as well as the response of interest, a quantitative measure of disease progression one year after baseline.*
Diamonds	53,940	6	*This dataset contains the prices and other attributes of about 54,000 diamonds.*

Sulfur	10,081	6	*The sulfur recovery unit removes environmental pollutants from acid gas streams before they are released into the atmosphere. Furthermore, elemental sulfur is recovered as a valuable by-product. The inputs variables are gas and air flows. Output to predict is H_2S concentrations.*

Table 10: The data from the above datasets can be found at: Abalone, Ailerons, Bodyfat, Debutanizer, Diabetes, Diamonds and Sulfur.

Table 11 shows the out-of-sample prediction RMSE for the real datasets (standard errors in brackets).

Table 11: Test RMSE for the real datasets

Dataset	OLS	adaLASSO	SVR	RF	STR RF	BooST	adaLASSO + RF	adaLASSO + STR RF
Abalone	2.2466	2.2990	2.1894	2.2078	2.2042	**2.1380**	2.2166	2.2028
	(0.0627)	(0.0686)	(0.0646)	(0.0584)	(0.0621)	(0.0854)	(0.0690)	(0.0674)
Ailerons	0.000171	0.000174	0.000997	0.000169	0.000193	0.000200	**0.000160**	0.000165
	(0.000004)	(0.000004)	(0.000121)	(0.000004)	(0.000006)	(0.000150)	(0.000004)	(0.000004)
Bodyfat	1.3215	1.7836	4.0707	1.4104	1.3297	1.2833	1.2024	**1.1838**
	(0.5463)	(0.2416)	(0.5399)	(0.4818)	(0.4099)	(0.4054)	(0.6306)	(0.5989)
Diamonds	1520.5072	1552.4293	3886.2191	1482.0128	1426.6054	1447.8242	1471.6791	**1422.1754**
	(66.6279)	(46.8758)	(107.7864)	(47.6162)	(51.1793)	(72.9805)	(45.0823)	(47.8769)
Debutanizer	0.1410	0.1454	0.1023	0.0738	0.1278	0.0998	**0.0705**	0.1256
	(0.0057)	(0.0056)	(0.0041)	(0.0048)	(0.0053)	(0.0041)	(0.0044)	(0.0054)
Diabetes	55.0954	56.9455	71.3997	58.1980	56.1425	61.0016	56.4176	**55.0640**
	(2.3727)	(2.3054)	(3.2751)	(2.6141)	(2.4211)	(2.9232)	(2.4612)	(2.3377)
Sulfur	0.0434	0.0470	0.0439	0.0259	0.0277	**0.0258**	0.0259	0.0278
	(0.0048)	(0.0050)	(0.0034)	(0.0053)	(0.0051)	(0.0047)	(0.0053)	(0.0051)

First, we can see that the adaLASSO+STR method performed best in three out of the seven real datasets (Bodyfat, Diamonds and Diabetes), a higher count than any of the other methods, and was the second best in one dataset (Ailerons). The related adaLASSO+RF method presented the lowest errors in two datasets (Ailerons and Debutanizer). Combined, these two methods were the top performers in 71.4% of this particular selection of real data. BooST had the best performance in two datasets.

In this selection of real data, in which the true generating process is unknown, we can see the adaLASSO+STR RF method has shown to be a competitive alternative to other benchmarks for prediction in regression problems.

5. CONCLUSION

We presented a novel regression method combining LASSO-type penalties and a STR random forest, which has shown advantages in predictive performance with respect to other benchmarks on several selected simulated and real data. We ran experiments which led us to believe that our hybrid method is capable of capturing the linearities in the data, if any, better than any of the standalone procedures. By employing a parametric penalized linear regression in the first step, most of the linear relations presented in the data is captured first, and we let a highly flexible non-parametric approach to access the remaining nonlinear relations which were not properly "understood" in the first step.

Tree-based methods are ideal in this case, because they do not demand any assumption about the distribution of the data, and can capture patterns which linear methods cannot. In addition to that, we innovate in using an ensemble of smoothed trees, in contrast to the traditional hard split tree, which is known to generalize the fit better near the boundaries of the training region. To the best of our knowledge, our work is the first to implement a random forest of STR-trees.

Also, our simulations revealed that the higher the influence of linear components in the data, the better the adaLASSO+STR RF and adaLASSO+RF methods predictive performance is. The predited errors tend to be lower when the magnitude of the linear terms are greater. With a response formed by a combination of linear and nonlinear terms, the proposed methods had performed better predicting the out-of-sample values when the linear terms are bigger, in comparisson to the other terms. This is in line with the rational developed in Section 2.

Other advantage of the proposed method, especially in contrast to the boosting approach, is that, in the case of random forests, the individual tree growth process can be parallelized, and our implementation takes advantage of this, while boosting is intrinsically a sequential process. This result in a much faster execution by the adaLASSO+STR RF method in contrast to the BooST model, for example[5].

Finally, one possible future work is to expand the method to deal with time series data, using the WLadaLASSO (Konzen & Ziegelmann, 2016) model, in

[5]In our simulations, adaLASSO+STR RF was five to ten times faster than BooST.

which every lagged variable has a different penalty weight, in the first step of the proposed method. Another path of development may be the adaptation of the proposed procedure to deal with classification problems, possibly employing other types of smoothed trees, such as Soft, Fuzzy or Probabilistic trees, mentioned in the introduction.

REFERENCES

References

Alkhoury, S., Devijver, E., Clausel, M., Tami, M., Gaussier, E., & Oppenheim, g. (2020). Smooth and consistent probabilistic regression trees. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems* (pp. 11345–11355). Curran Associates, Inc. volume 33. URL: https://proceedings.neurips.cc/paper/2020/file/8289889263db4a40463e3f358bb7c7a1-Paper.pdf.

Awaya, N., & Ma, L. (2021). Tree boosting for learning probability measures. URL: https://arxiv.org/abs/2101.11083.

Berk, R. A. (2016). *Statistical Learning from a Regression Perspective*. (2nd ed.). Springer International Publisher Switzerland. doi:10.1007/978-3-319-44048-4.

Biau, G., Devroye, L., & Lugosi, G. (2008). Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, *9*, 2015–2033. doi:10.1145/1390681.1442799.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140. doi:10.1007/BF00058655.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32. doi:10.1023/A:1010933404324.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees*. Chapman and Hall. doi:10.1201/9781315139470.

Calhoun, P., Hallett, M. J., Su, X., Cafri, G., Levine, R. A., & Fan, J. (2020). Random forest with acceptance–rejection trees. *Computational Statistics*, *35*, 983–999. doi:10.1007/s00180-019-00929-4.

Dietterich, T. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, *40*. doi:10.1023/A:1007607513941.

Dua, D., & Graff, C. (2021). UCI machine learning repository. URL: http://archive.ics.uci.edu/ml.

Efron, B. (1977). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, *7*, 1–26. doi:10.1214/aos/1176344552.

Fan, J., & Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, *96*, 1348–1360. doi:10.1198/016214501753382273.

Fonseca, Y., Medeiros, M. C., Vasconcelos, G., & Veiga, A. (2020). Boost: Boosting smooth transition regression trees for partial effect estimation in nonlinear regressions, . doi:10.48550/arXiv.1808.03698.

Friedman, J. (1997). On bias, variance, 0/1—loss, and the curse-of-dimensionality. *Data Min. Knowl. Discov.*, *1*, 55–77. doi:10.1023/A:1009778005914.

Friedman, J. H. (1991). Multivariate Adaptive Regression Splines. *The Annals of Statistics*, *19*, 1 – 67. URL: https://doi.org/10.1214/aos/1176347963. doi:10.1214/aos/1176347963.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, *29 (5)*, 1189–1232. doi:10.1214/aos/1013203451.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*, 1573–0565. URL: https://doi.org/10.1007/s10994-006-6226-1. doi:10.1007/s10994-006-6226-1.

Geurts, P., & Wehenkel, L. (2007). Investigation and reduction of discretization variance in decision tree induction. (pp. 162–170). volume 1810. doi:10.1007/3-540-45164-1_17.

Guo, H., & Gelfand, S. (1992). Classification trees with neural network feature extraction. *IEEE Transactions on Neural Networks*, *3*, 923–933. doi:10.1109/72.165594.

Göçken, M., Özçalıcı, M., Boru, A., & Dosdoğru, A. T. (2016). Integrating metaheuristics and artificial neural networks for improved stock price prediction. *Expert Systems with Applications*, *44*, 320–331. URL: https://www.sciencedirect.com/science/article/pii/S0957417415006570. doi:https://doi.org/10.1016/j.eswa.2015.09.029.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer, New York, NY. doi:10.1007/978-0-387-84858-7.

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition* (pp. 278–282 vol.1). volume 1. doi:10.1109/ICDAR.1995.598994.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*, 832–844. doi:10.1109/34.709601.

Hoerl, A., & Kennard, R. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*, 55–67. doi:10.1080/00401706.1970.10488634.

Irsoy, O., Yildiz, O. T., & Alpaydin, E. (2012). Soft decision trees. *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, (pp. 1819–1822).

Konzen, E., & Ziegelmann, F. A. (2016). Lasso-type penalties for covariate selection and forecasting in time series. *Journal of Forecasting*, *35*, 592–612. doi:10.1002/for.2403.

Lin, Y., & Jeon, Y. (2006). Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, *101*, 578–590. URL: http://www.jstor.org/stable/27590719.

Linero, A., & Yang, Y. (2017). Bayesian regression tree ensembles that adapt to smoothness and sparsity. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *80*. doi:10.1111/rssb.12293.

Medeiros, M. C., & Mendes, E. F. (2015). *l1-Regularization of High-Dimensional Time-Series Models with Flexible Innovations*. Texto para discussão 636, PUC-Rio, Department of Economics. URL: http://hdl.handle.net/10419/176119.

Miller, P., Lubke, G., Mcartor, D., & Bergeman, C. (2015). Finding structure in data using multivariate tree boosting. *Psychological Methods*, *21*. doi:10.1037/met0000087.

Morgan, J. N., & Sonquist, J. A. (1963). Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, *58*, 415–434. URL: http://www.jstor.org/stable/2283276.

Murthy, S. K., Kasif, S., & Salzberg, S. (1994). A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, *2*, 1–32. doi:https://doi.org/10.1613/jair.63.

Pedregosa, F. et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. URL: https://dl.acm.org/doi/10.5555/1953048.2078195.

Rainforth, T., & Wood, F. (2017). Canonical correlation forests, . URL: https://arxiv.org/abs/1507.05444. doi:https://doi.org/10.48550/arXiv.1507.05444.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, *33*, 1–39. doi:10.1007/s10462-009-9124-7.

da Rosa, J. C., Veiga, A., & Medeiros, M. C. (2008). Tree-structured smooth transition regression models. *Computational Statistics & Data Analysis*, *52*, 2469–2488. doi:10.1016/j.csda.2007.08.018.

Strobl, C., Malley, J., & Tutz, G. (2009). An introduction to recursive partitioning: Rationale, application, and characteristics of classification and regression trees, bagging, and random forests. *Psychological methods*, *14*, 323–48. doi:10.1037/a0016973.

Suarez, A., & Lutsko, J. (2000). Globally optimal fuzzy decision trees for classification and regression. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *21*, 1297 – 1311. doi:10.1109/34.817409.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, *58*, 267–288. doi:10.1111/j.2517-6161.1996.tb02080.x.

Vanschoren, J., van Rijn, J. N., Bischl, B., & Torgo, L. (2013). Openml: networked science in machine learning. *SIGKDD Explorations*, *15*, 49–60. URL: http://doi.acm.org/10.1145/2641190.264119. doi:10.1145/2641190.2641198.

Wehenkel, L. (1997). Discretization of continuous attributes for supervised learning: variance evaluation and variance reduction. In *In Proc. of The Int. Fuzzy Systems Assoc. World Congress IFSA 97* (pp. 381–388).

Yildiz, O., & Alpaydın, E. (2000). Linear discriminant trees. (pp. 1175–1182). volume 19. doi:10.1142/S0218001405004125.

Yildiz, O. T., Irsoy, O., & Alpaydin, E. (2016). Bagging soft decision trees. In *Machine Learning for Health Informatics*.

Zhao, P., & Yu, B. (2006). On model selection consistency of lasso. *Journal of Machine Learning Research*, *7*, 2541–2563. URL: http://jmlr.org/papers/v7/zhao06a.html.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, *101 (476)*, 1418–1429. doi:10.1198/016214506000000735.

Chapter 3

FINAL REMARKS AND FUTURE WORK

We presented a novel regression method combining LASSO-type penalties and a STR random forest, which have shown advantages in predictive performance with respect to other benchmarks on several selected simulated and real data. We ran experiments which led us to belive that our hybrid method is capable of capturing the linearities in the data, if any, better than any of the standalone procedures. By employing a parametric penalized linear regression in the first step, most of the linear relations presented in the data is captured first, and we let a highly flexible non-parametric approach to access the remaining nonlinear relations which were not properly "understood" in the first step.

One possible future work is to expand the method to deal with time series data, using the WLadaLASSO Konzen and Ziegelmann (2016) model, in which every lagged variable has a different penalty weight, in the first step of the proposed method. Another possibility is to incorporate the idea of variable importance, which can be done via permutation of the features that are trained each time in the ensemble of trees, such as random forests, following the works of Strobl et al. (2007), Louppe et al. (2013) and Gregorutti et al. (2016), combined with the property of variable selection of the LASSO, to build a way of simultaneously identify linear and nonlinear features which have more importance to the response. Another path of development may be the adaptation of the proposed procedure to deal with classification problems, possibly employing other types of smoothed trees, such as Soft, Fuzzy or Probabilistic trees. Lastly, it may be worth exploring the interpretability of the method. A random forest is by nature much less interpretable than an individual tree, but recently there has been an effort to understand better its modeling process, see for example the works of Audemard et al. (2011) and Miller et al. (2015). The selection of important linear terms done in the first step of our method can contribute to the interpretation of the whole model.

Bibliography

Alkhoury, S., Devijver, E., Clausel, M., Tami, M., Gaussier, E., Oppenheim, g., 2020. Smooth and consistent probabilistic regression trees, in: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc.. pp. 11345–11355. URL: https://proceedings.neurips.cc/paper/2020/file/8289889263db4a40463e3f358bb7c7a1-Paper.pdf.

Audemard, G., Bellart, S., Bounia, L., Koriche, F., Lagniez, J.M., Marquis, P., 2011. Trading complexity for sparsity in random forest explanations URL: https://arxiv.org/abs/2108.05276, doi:https://doi.org/10.48550/arXiv.2108.05276.

Awaya, N., Ma, L., 2021. Tree boosting for learning probability measures. URL: https://arxiv.org/abs/2101.11083.

Berk, R.A., 2016. Statistical Learning from a Regression Perspective. 2 ed., Springer International Publisher Switzerland. doi:10.1007/978-3-319-44048-4.

Biau, G., Devroye, L., Lugosi, G., 2008. Consistency of random forests and other averaging classifiers. Journal of Machine Learning Research 9, 2015–2033. doi:10.1145/1390681.1442799.

Breiman, L., 1996. Bagging predictors. Machine Learning 24, 123–140. doi:10.1007/BF00058655.

Breiman, L., 2001. Random forests. Machine Learning 45, 5–32. doi:10.1023/A:1010933404324.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984. Classification and Regression Trees. Chapman and Hall. doi:10.1201/9781315139470.

Calhoun, P., Hallett, M.J., Su, X., Cafri, G., Levine, R.A., Fan, J., 2020. Random forest with acceptance–rejection trees. Computational Statistics 35, 983–999. doi:10.1007/s00180-019-00929-4.

Dietterich, T., 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning 40. doi:10.1023/A:1007607513941.

Efron, B., 1977. Bootstrap methods: Another look at the jackknife. The Annals of Statistics 7, 1–26. doi:10.1214/aos/1176344552.

Fan, J., Li, R., 2001. Variable selection via nonconcave penalized likelihood and its oracle properties. Journal of the American Statistical Association 96, 1348–1360. doi:10.1198/016214501753382273.

Fonseca, Y., Medeiros, M.C., Vasconcelos, G., Veiga, A., 2020. Boost: Boosting smooth transition regression trees for partial effect estimation in nonlinear regressions doi:10.48550/arXiv.1808.03698.

Friedman, J., 1997. On bias, variance, 0/1—loss, and the curse-of-dimensionality. Data Min. Knowl. Discov. 1, 55–77. doi:10.1023/A:1009778005914.

Friedman, J.H., 2001. Greedy function approximation: A gradient boosting machine. Annals of Statistics 29 (5), 1189–1232. doi:10.1214/aos/1013203451.

Gandini, A., Ziegelmann, F., 2022. Combining LASSO-Type Methods with a Smooth Transition Random Forest. URL: https://github.com/alexgand/adalasso_STR_RF.

Geurts, P., Ernst, D., Wehenkel, L., 2006. Extremely randomized trees. Machine Learning 63, 1573–0565. URL: https://doi.org/10.1007/s10994-006-6226-1, doi:10.1007/s10994-006-6226-1.

Geurts, P., Wehenkel, L., 2007. Investigation and reduction of discretization variance in decision tree induction, pp. 162–170. doi:10.1007/3-540-45164-1_17.

Gregorutti, B., Michel, B., Saint-Pierre, P., 2016. Correlation and variable importance in random forests. Statistics and Computing 27, 659–678. URL: https://doi.org/10.1007%2Fs11222-016-9646-1, doi:10.1007/s11222-016-9646-1.

Guo, H., Gelfand, S., 1992. Classification trees with neural network feature extraction. IEEE Transactions on Neural Networks 3, 923–933. doi:10.1109/72.165594.

Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning. Springer, New York, NY. doi:10.1007/978-0-387-84858-7.

Ho, T.K., 1995. Random decision forests, in: Proceedings of 3rd International Conference on Document Analysis and Recognition, pp. 278–282 vol.1. doi:10.1109/ICDAR.1995.598994.

Ho, T.K., 1998. The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20, 832–844. doi:10.1109/34.709601.

Hoerl, A., Kennard, R., 1970. Ridge regression: Biased estimation for nonorthogonal problems. Technometrics 12, 55–67. doi:10.1080/00401706.1970.10488634.

Irsoy, O., Yildiz, O.T., Alpaydin, E., 2012. Soft decision trees. Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012) , 1819–1822.

Kass, G.V., 1980. An exploratory technique for investigating large quantities of categorical data. Journal of the Royal Statistical Society. Series C (Applied Statistics) 29, 119–127. URL: http://www.jstor.org/stable/2986296.

Konzen, E., Ziegelmann, F.A., 2016. Lasso-type penalties for covariate selection and forecasting in time series. Journal of Forecasting 35, 592–612. doi:10.1002/for.2403.

Lin, Y., Jeon, Y., 2006. Random forests and adaptive nearest neighbors. Journal of the American Statistical Association 101, 578–590. URL: http://www.jstor.org/stable/27590719.

Linero, A., Yang, Y., 2017. Bayesian regression tree ensembles that adapt to smoothness and sparsity. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 80. doi:10.1111/rssb.12293.

Loh, W.Y., 2014. Fifty years of classification and regression trees. International Statistical Review / Revue Internationale de Statistique 82, 329–348. URL: http://www.jstor.org/stable/43298996.

Louppe, G., Wehenkel, L., Sutera, A., Geurts, P., 2013. Understanding variable importances in forests of randomized trees.

Miller, P., Lubke, G., Mcartor, D., Bergeman, C., 2015. Finding structure in data using multivariate tree boosting. Psychological Methods 21. doi:10.1037/met0000087.

Morgan, J.N., Sonquist, J.A., 1963. Problems in the analysis of survey data, and a proposal. Journal of the American Statistical Association 58, 415–434. URL: http://www.jstor.org/stable/2283276.

Murthy, S.K., Kasif, S., Salzberg, S., 1994. A system for induction of oblique decision trees. Journal of Artificial Intelligence Research 2, 1–32. doi:https://doi.org/10.1613/jair.63.

Quinlan, J.R., 1986. Induction of decision trees. Machine Learning 1, 1573–0565. URL: https://doi.org/10.1007/BF00116251, doi:10.1007/BF00116251.

Quinlan, J.R., 1993. C4.5: Programs for machine learning. Machine Learning 16, 235–240. URL: https://doi.org/10.1007/BF00993309, doi:10.1007/BF00993309.

Rainforth, T., Wood, F., 2017. Canonical correlation forests URL: https://arxiv.org/abs/1507.05444, doi:https://doi.org/10.48550/arXiv.1507.05444.

Rokach, L., 2010. Ensemble-based classifiers. Artificial Intelligence Review 33, 1–39. doi:10.1007/s10462-009-9124-7.

da Rosa, J.C., Veiga, A., Medeiros, M.C., 2008. Tree-structured smooth transition regression models. Computational Statistics & Data Analysis 52, 2469–2488. doi:10.1016/j.csda.2007.08.018.

Strobl, C., Boulesteix, A.L., Zeileis, A., Hothorn, T., 2007. Bias in random forest variable importance measures: Illustrations, sources and a solution. BMC Bioinformatics 8, 1471–2105. URL: https://doi.org/10.1186/1471-2105-8-25, doi:10.1186/1471-2105-8-25.

Strobl, C., Malley, J., Tutz, G., 2009. An introduction to recursive partitioning: Rationale, application, and characteristics of classification and regression trees, bagging, and random forests. Psychological methods 14, 323–48. doi:10.1037/a0016973.

Suarez, A., Lutsko, J., 2000. Globally optimal fuzzy decision trees for classification and regression. Pattern Analysis and Machine Intelligence, IEEE Transactions on 21, 1297 – 1311. doi:10.1109/34.817409.

Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society, Series B 58, 267–288. doi:10.1111/j.2517-6161.1996.tb02080.x.

Wehenkel, L., 1997. Discretization of continuous attributes for supervised learning: variance evaluation and variance reduction, in: In Proc. of The Int. Fuzzy Systems Assoc. World Congress (IFSA'97, pp. 381–388.

Yildiz, O., Alpaydın, E., 2000. Linear discriminant trees., pp. 1175–1182. doi:10.1142/S0218001405004125.

Yildiz, O.T., Irsoy, O., Alpaydin, E., 2016. Bagging soft decision trees, in: Machine Learning for Health Informatics.

Zhao, P., Yu, B., 2006. On model selection consistency of lasso. Journal of Machine Learning Research 7, 2541–2563. URL: http://jmlr.org/papers/v7/zhao06a.html.

Zou, H., 2006. The adaptive lasso and its oracle properties. Journal of the American Statistical Association 101 (476), 1418–1429. doi:10.1198/016214506000000735.

Chapter 4

APPENDICES

4.1 Glossary

- Regression Tree: a non-parametric regression method whose estimation is obtained by recursivelly partitioning the space of covariates, aiming to approximate the unknown data generating function. It accomplishes that building a structure that resembles an inverted tree (Hastie et al., 2009);
- CART: Classification and Regression Trees, a greedy, top-down algorithm that searches for the feature and cut point that split the observations from the root node on, recursivelly, until some stop criteria is met (Breiman et al., 1984);
- STR-Tree: a regression tree method which uses a sigmoid-like function in each node, assigning a group membership degree to the observations (da Rosa et al., 2008);
- Random Forest: a method that combines the ideas behind bagging and the random subspace method, where the growing of several individual trees has two sources of randomization: the first one is the bootstrap sampling of the training set, where sometimes an observation will be drawn more than once and sometimes will not be drawn at all, whereas the second source is the random selection of the subset of candidate regressors in each split of the trees (Breiman, 2001);
- LASSO: Least Absolute Shrinkage and Selection Operator, a regression method with a ℓ_1 penalization on the coefficients of a model, which sets some of them to zero, effectively performing simultaneously parameter estimation and variable selection (Tibshirani, 1996).