Università di Firenze, Università di Perugia, INdAM consorziate nel CIAFM

DOTTORATO DI RICERCA
IN MATEMATICA, INFORMATICA, STATISTICA

CURRICULUM IN STATISTICA
CICLO XXXI

Sede amministrativa Università degli Studi di Firenze
Coordinatore Prof. Graziano Gentili

Interpretable Semilinear Regression Trees

Settore Scientifico Disciplinare SECS-S/01

Dottoranda
Giulia Vannucci

Tutore
Prof.ssa Anna Gottard

Coordinatore
Prof. Graziano Gentili

Anni 2015/2018

To Emiliano

Aknowledgments

Journey of a thousand miles starts with a single step. First of all I would like to thank Francesco and Giulia, without whom this journey would have never begun, and Ilaria, without whom this journey would have never ended. In addition, this dissertation would certainly not have come to its conclusion without the trust, support and love of colleagues, friends and family. I would like to sincerely thank my advisor Anna Gottard for her support and guide during my PhD study. I would like to thank Professor Giovanni Maria Marchetti for its essential contribution in one of the works presented in this thesis. I would like to thank Professor Bruno Scarpa and Professor Paola Cerchiello for their detailed and very constructive comments on my thesis, which contributed to improving my work. I am grateful to all my colleagues who support me in every desperate situation and listen patiently each stupid question I made during these years. Thanks to Nedka, Luigi, Federica, Marco and Alessandro. I would also like to thank the students present to my tutor lessons, thanks to them and to my experience as tutor, I clearly see the path ahead for me. I am very lucky to share my life with lots of lovely people, who believe in me even when I don't believe in myself. My family is the first place in the world where I learnt about freedom, the freedom to be the person that I want to be, the freedom to follow my dreams and the freedom to make choices in the direction of the happiness that I deserve. Thanks to Maura, Luciano and Chiara, I love you. During the years the family becomes larger with Roberto and Matteo, who I would like to thank always being at my side. Recently my family has become larger another time. I feel Lea, Fernando and Emily as part of my family, and I would like to thank them for their support in my everyday life. A special thanks to Emily, my lovely "papera", for every fight, for every encouragement, for every joke, for every break that we share together. Thank also for the last minute help in every situation, you are precious. I love you. Family is also made from invisible bands that link people on the basis of a special feeling: friendship. To my beloved friends, thank you for everything that you did for me, from a crazy night spent together to every gif or message shared, from a dinner to a patient Pilates lesson. Special thanks to Cheren, Valentina, Silvia, Valentina, Gabriele, Lorena, Maddalena, Annagiulia, Alessio, Jonata, Elisa, Diletta, Elisa, Silvia, Laura, Valentina, Giulia, Lucia, Denise, Chiara and Claudia. A special and particular thanks to Giulia and Benedetta, for their incredible help in every anxiety moment, and for the fundamental work of revision. I will never forget what you did for me and for the success of this dissertation. I love you. An enormous thank to Mariachiara, for her absolute love and help in every situation. I love you, sister. I would like to thank Ikeda Sensei and Nichiren Daishonin for every encouragement they gave to me in these years. Special thanks to members of the "Smilea" group. Last but not least, Emiliano. I am forever grateful for your unconditional support and love, and I would say yes a thousand times over. I dedicate my success to you and to all our dreams that we will fulfil together. I love you to the moon and back.

Abstract

Tree-based methods refer to a class of predictive models largely employed in many scientific areas. Regression trees partition the variable space into a set of hyper-rectangles, and fit a model within each of them. They are conceptually simple, apparently easy to interpret and capable to deal with non linearities and interactions. Random forests are an ensemble of regression trees constructed on subsamples of statistical units and on a subset of explanatory variables randomly selected. The prediction is a combination of this kind of trees. Despite the loss in interpretability, thanks to their high predictive performance, random forests have achieved great success. The aim of this thesis is to propose a class of models combining a linear component and a tree, able to discover the relevant variables directly influencing a response. The proposal is a semilinear model that can handle linear and non linear dependencies and maintains a good predictive performance, while ensuring a simple and intuitive interpretation in a generative model sense. Moreover, two different algorithms for estimation, a two-stage estimation procedure based on a backfitting algorithm and one based on evolutionary algorithms are proposed.

Contents

List of Figures

List of Tables

Introduction

Regression trees (Breiman et al., 1984) are a class of predictive models used in many scientific areas, such as artificial intelligence, engineering, information technology, medicine, epidemiology, bioinformatics, marketing and psychology. These models are appealing as they easily deal with non linearities and interactions. To this aim, regression trees based on the CART algorithm lead to a piecewise-constant representation of the regression function trough the recursive partitioning in binary split of the explanatory variable space. Moreover, these models are particularly appreciated for being easy interpretable, if small, due to the diagram that represent the partition selected by the algorithm as a tree. Because of this diagram, regression trees are sometimes used to select the relevant variables or risk factors, which are interpreted as directly influencing the response variable (see, for example, Karaolis et al., 2010). From regression trees introduction, many other tree-based models have been developed to move on from their first formulation, for example oblique trees (Murthy et al., 1994) and conditional inference trees (Hothorn et al., 2006).

Alongside the vast work on regression trees, another class of predictive models, the so-called ensemble methods, have been developed in the past decades. These methods are based on the basic idea of using weak multiple learners and then combine their predictions. Ensemble algorithms are usually significantly more accurate in prediction than single learners. Thanks to their high performance, ensemble methods have achieved great success. A representative example of ensemble methods are the random forests (Breiman, 2001). This algorithm combines several individual regression trees computed on samples of units and on a subset of explanatory variables randomly selected. Random forests inherit from regression trees the ability of handle non linearities and interactions. Despite their performance, ensemble methods have an important drawback: they are in essence a *black box*, as they lose the easy interpretation of regression trees. The interpretability of these models raises lot of interest in the scientific community, strengthened further from European new regulations. In fact, the European Parliament recently adopted the General Data Protection Regulation AA.VV. (2016), introducing the right for all individuals to obtain a meaningful explanation of the logic involved in

the decision making automatic algorithms. When these algorithms are employed in real contexts it is important to know if they induce biases and prejudice, leading to unfair and wrong decisions (Guidotti et al., 2018).

The aim of this dissertation is to study a class of models based on trees, able to discover the relevant variables, determinants or risk factors directly influencing a response. As one of the major drawback of regression trees is their difficulty in handle linear relationships, the class of model adopted here is particularly adequate in case of linear and quasi-linear relationships, maintaining a good predictive performance while ensuring a simple and intuitive interpretation.

This dissertation is organized as follows.

Chapter 1 gives a brief overview of regression trees and the CART algorithm (Breiman et al., 1984), their extensions such as oblique trees (Murthy et al., 1994), conditional inference trees (Hothorn et al., 2006), treed regression (Alexander and Grimshaw, 1996) and evolutionary trees (Grubinger et al., 2011), random forests (Breiman, 2001), their extensions such as extremely randomized trees (Geurts et al., 2006) and reinforcement learning trees (Zhu et al., 2015), and Bayesian additive regression trees (Chipman et al., 2010).

Chapter 2 presents some pitfalls for which the utilization of regression trees and random forests gives erroneous information about the data generating process. These pitfalls are due to the greedy search of the CART algorithm.

In Chapter 3 the Semilinear regression trees (SRT) is presented to overcome these pitfalls. SRT are able to treat both linear and non linear relationships, and are easy to interpret prediction rules. SRT integrates a linear and a tree component in a semiparametric model, whose parameters can be estimated via two new algorithms. The first proposed algorithm is a two-stage estimation procedure based on a backfitting algorithm (Buja et al., 1989). The second proposed algorithm is based on evolutionary algorithms (Grubinger et al., 2011).

Chapter 4 presents a Monte Carlo study and a real data study to asses the estimation and the predictive accuracy of the proposals and to compare them with the existing algorithms.

Some concluding remarks are reported in Chapter 5.

Chapter 1

Tree-based learning methods

1.1 Introduction

Regression trees (Breiman et al., 1984) are a class of predictive models used in many scientific areas, such as artificial intelligence, engineering, information technology, medicine, epidemiology and bioinformatics.

Tree-based methods are conceptually simple, useful for interpretation yet powerful for outcome predicting. They partition the variable space into a set of hyperrectangles, and fit a model within each of them. One of the most popular algorithm for tree-based regression and classification is *Classification and Regression Trees, CART* (Breiman et al., 1984). The CART algorithm recursively finds a binary partition of the explanatory variable space that leads to an accurate piecewise-constant representation of the regression function. Since classification problems are out of the scope of this dissertation, for a detailed decription of classification trees see Breiman et al. (1984) and Hastie et al. (2009).

The history of regression tree algorithms begins from the very first regression tree algorithm, which is *Automatic Interaction Detection (AID)* (Morgan and Sonquist, 1963), that chooses the split that minimize the squared deviation in two nodes. The result is a piecewise constant estimate of the regression function. Then, *THAID* (Morgan and Messenger, 1973) algorithm extends the idea of AID to categorical response. *CHAID* (Kass, 1980) employs a kind of stepwise regression for split selection. C4.5 Quinlan (2014) is an extension of the ID3 algorithm Quinlan (1986). In this case, the split are binary only for continuous explanatory variables, whereas for categorical values the number of split is equal to the number of categories of the variable involved in the split. FACT (Loh and Vanichsetakul, 1988) performs a recursive linear discriminant analysis, which generates linear splits. As a result, the algorithm splits each node into as many nodes as the number of classes. To obtain splits, FACT uses the F-tests of ANOVA to rank the explanatory variables. Then the LDA is applied to the most significant

variable to split the node. QUEST (Loh and Shih, 1997) uses an F-tests only for continuous variables, while for categorical variables it employs a contingency table chi-squared tests. CRUISE (Kim and Loh, 2003) is a descendent of QUEST. The difference is in the splitting in multiple nodes, which number depends on the number of distinct values of the response variable.

Even if regression trees do not boast the maximum predictive performance, they are nevertheless appreciated for being easily interpretable (see, among others, Wolfson and Venkatasubramaniam, 2018), because of the accompanying recursive diagram depicting as a tree the partition selected by the algorithm.

Random forests (Breiman, 2001) is an ensemble algorithm based on an average of regression trees each of which is based on randomly selected explanatory variables and bootstrapped units. Random forests exhibit higher predictive performance and lower predictive variability than regression trees, but the interpretation of the results is less straightforward, being based on variable importance measures.

Bayesian Additive Regression Trees, BART (Chipman et al., 2010), are a sum of tree models where each tree is constrained by a regularization prior to be a weak learner. Fitting and inference are done via an iterative Bayesian backfitting MCMC algorithm. Actually, BART reach excellent empirical performances that are considerably superior with respect to other tree-based algorithm. Moreover, another strength is the uncertainty quantification in the model.

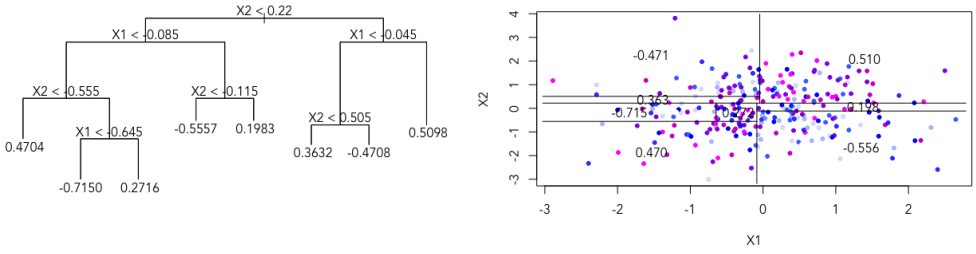
In this Chapter, three tree-based methods will be presented: regression trees, random forests and BART. After a brief review of the original models proposed respectively by Breiman et al. (1984), Breiman (2001) and Chipman et al. (2010), more recent development in these methods will be presented.

Throughout this thesis, I will consider $\boldsymbol{X} = (X_1, \ldots, X_p)$ as a vector of explanatory variables taking values in $\mathcal{X} \equiv \mathbb{R}^p$ and Y a response variable with values in $\mathbb{R}$. I suppose also to observe an *iid* sample of n units from this population, and $(\boldsymbol{X}_i, Y_i)^{\texttt{train}}$, with $i = 1, \ldots, n_{\texttt{train}}$, will be the *training set* or *learning sample*, with which the prediction rule is constructed, and $(\boldsymbol{X}_i, Y_i)^{\texttt{test}}$, with $i = 1, \ldots, n_{\texttt{test}}$, will be the *test set*, with which the prediction rule is validated. Later I will call the training and the test set $(\boldsymbol{X}, \boldsymbol{Y})^{\texttt{train}}$ and $(\boldsymbol{X}, \boldsymbol{Y})^{\texttt{test}}$.

1.2 Regression trees

Regression trees are a class of non parametric predictive models that leads to a piecewise-constant representation of the regression function. One of the most popular algorithm to find a partition of the explanatory variable space is the CART algorithm (Breiman et al., 1984).

Definition 1.2.1. Given a training set, a *regression tree* $T(\boldsymbol{X})$ is a tree structured



(a) (b)

Figure 1.1: A graphical representation of a regression tree by a tree on the left (a) and by the corresponding partition on the right (b).

predictor that, trough the *recursive partitioning* in binary split of the space $\mathcal{X}$ of explanatory variables, predicts the response Y as a constant.

A regression tree model can be represented by a binary tree, see Figure 1.1(a). The tree is constructed by repeated split of $\mathcal{X}$, starting from $\mathcal{X}$ itself, called *root node*. Then *root node* is divided in two disjoint subsets, called *nodes*. After the recursive partitioning, the *terminal nodes* or *leaves* form a partition of $\mathcal{X}$. The construction of the tree moves around three elements: the splitting rule, the stopping rule, and the prediction rule.

Definition 1.2.2. The *splitting rule* refers to the choice of the split variable X_j at the split point s, which is the variable X_j that most decrease the Mean Square Error (MSE) of the tree $MSE = \mathbb{E}(Y - \hat{T}(\boldsymbol{X}))^2$.

Definition 1.2.3. The *stopping rule* is the criterion with which a node is declared terminal.

Definition 1.2.4. The *prediction rule* for a regression trees with splitting rule defined in 1.2.2 is the average of Y falling into each terminal node.

The algorithm stops the recursion when a minimum value of observations (generally 5) is in the node.

The regression tree model is

$$T(\boldsymbol{X}) = T(\boldsymbol{X}; \mathcal{R}_Y, \boldsymbol{\mu}) = \sum_{m=1}^{M} \mu_{R_m} \mathbb{I}_{(\boldsymbol{X} \in R_m)} \tag{1.1}$$

where $\mathcal{R}_Y = (R_1, \ldots, R_M)$ is a binary recursive partition of $\mathcal{X}$: $\mathcal{X} = \bigcup_{m=1}^{M} R_m$, $\mu_{R_m} = \mathbb{E}(Y|\boldsymbol{X} \in R_m)$, the means of Y within the terminal regions (or nodes), are the values that minimize the MSE of the tree, and $\mathbb{I}_{(\boldsymbol{X} \in R_m)}$ is the indicator variable of the elements within the partition.

CART algorithm starts from the root node and divides the explanatory variable space in two half-plains: $R_1(j, s_j) = \{\mathcal{X} : X_j \leq s_j\}$ and $R_2(j, s_j) = \{\mathcal{X} : X_j > s_j\}$. Then, the variable X_j at the split point s^* that most decrease the MSE of the tree is selected by solving

$$\min_{j,s} \left[\min_{\zeta_1} \sum_{\boldsymbol{X}_j \in R_1} (Y - \zeta_1)^2 + \min_{\zeta_2} \sum_{\boldsymbol{X}_j \in R_2} (Y - \zeta_2)^2 \right] \tag{1.2}$$

where the inner minimization gives $\hat{\zeta}_1 = \hat{\mu}_{R_1}$ and $\hat{\zeta}_2 = \hat{\mu}_{R_2}$. In that way, the best split and variable is the one that most successfully separates the high and the low response.

However, exploring all the possible partitions is computationally infeasible. For this reason, the CART, a top-down greedy approach, has been proposed by Breiman et al. (1984). The CART algorithm is summarized in Algorithm 1. It begins from the top of the tree, where all the units belong to the same node, and then successively splits the predictor space in two new nodes, from which the splitting procedure is repeated. The search is named greedy because at each step the procedure is conditioned to the previous step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

Algorithm 1: Pseudo Algorithm for regression tree building

```
Data: {Y_i, X_i}, i = 1, ..., n
Result: Regression tree
Initialization: all observations in the ROOT NODE;
Stopping rule = 0;
Node = 0;
for root node to nodes do
    repeat
        for j = 1 to p do
            for s = 1 to S do
                find the variable X_j at split point s* that minimize the MSE
                of the tree;
            end
        end
        partition the data in two nodes;
        compute the mean of Y within the node;
        Save → NODE
    until #i ⩽ 5 → Stopping rule = 1;
end
```

The size of the tree plays an important role: a large tree may overfit the data while a short tree may not capture important patterns in the dependence structure. The strategy proposed in the CART is to grow a large tree, and then prune it back

collapsing some node of the tree on the basis of a complexity measure.

Definition 1.2.5. The *Cost Complexity measure* C_α is defined as

$$C_\alpha(T) = Dev(T) + \alpha|M| \tag{1.3}$$

where $Dev(T)$ is the deviance at a node of the tree, $|M|$ is the number of terminal nodes of the tree T, and $\alpha \geq 0$ is the complexity parameter governing the trade-off between tree size and the goodness of fit. Small values of α lead to a large tree, instead larger values lead to smaller trees. The choice of α in order to select the *best* pruned subtree can be done adaptively by k-fold cross-validation. This approach involves randomly dividing the observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the search is made on the remaining $k-1$ folds.

Variable importance

Since one of the strengths of trees is the highly interpretability, an interesting question is which variables are the most important. Breiman et al. (1984) defined a measure, the *variable importance (VI)* , which reflects the relative importance, or contribution, of each input variable in predicting the response. If it is simple to think about the contribution of a splitting variable like the relative improvement in the deviance of the model, more difficult can be rank those variables that never occur in the tree structure. With this aim, the variable importance can be computed with the support of the surrogate split.

Let $\tilde{X}_j$ define the surrogate variable, which is a variable that most accurately predicts the action of the best split s on X_j selected by the CART algorithm.

Definition 1.2.6. The measure of VI of X_j is defined as

$$VI(X_j) = \sum_{m \in T} \Delta Dev(T)_{X_j \cup \tilde{X}_j} \tag{1.4}$$

where m are the terminal nodes of the tree T and $\Delta Dev(T)_{X_j \cup \tilde{X}_j}$ represents the decrease of the deviance done by X_j and it surrogate. If there exist more than one surrogate variable for X_j at any node, use the one with larger $\Delta Dev(T)$ in (1.4). The measure of importance of variables generally used are normalized quantities, $100VI(X_j)/\max_j VI(X_j)$, so the most important feature has measure 100, the others are in the range from 0 to 100.

Link with the single factor model

An interesting, unconventional, point of view links a regression tree to a single factor regression model and ANOVA, in case of Gaussianity.

In the regression tree model of (1.1) $\sum_{m=1}^{M} \mathbb{I}_{(\boldsymbol{x}_i \in R_m)} = 1$ as the terminal nodes of the tree are mutually exclusive and collectively exhaustive regions. Therefore it is possible to estimate the regression tree as a single factor regression model with the intercept term is set to zero in order to obtain

$$\mathbb{E}(Y|\boldsymbol{X}) = \mu_1 \mathbb{I}_{X_1 \in R_1} + \cdots + \mu_M \mathbb{I}_{X_j \in R_M} \tag{1.5}$$

where the indicator functions define a factor with M levels.

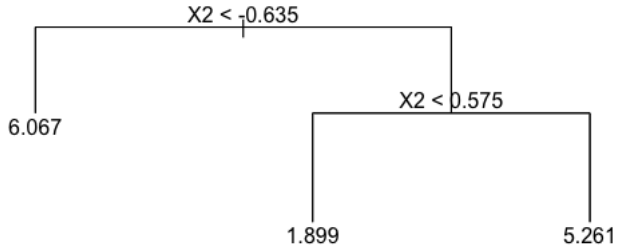


Figure 1.2: An example of regression tree with 3 terminal nodes.

As example, let us write the vector of the partitions that define the regression tree in Figure 1.2

$$\boldsymbol{R} = \left(\begin{array}{ccc} \mathbb{I}_{(\boldsymbol{X} \in R_1)}, & \mathbb{I}_{(\boldsymbol{X} \in R_2)}, & \mathbb{I}_{(\boldsymbol{X} \in R_3)} \end{array} \right).$$

With the OLS estimator $\boldsymbol{(R'R)^{-1}R'y}$ we obtain:

$$\boldsymbol{(R'R)^{-1}R'y} = \begin{pmatrix} \frac{1}{n_1} & 0 & 0 \\ 0 & \frac{1}{n_2} & 0 \\ 0 & 0 & \frac{1}{n_3} \end{pmatrix} \begin{pmatrix} \sum_{\boldsymbol{X} \in R_1} Y \\ \sum_{\boldsymbol{X} \in R_2} Y \\ \sum_{\boldsymbol{X} \in R_3} Y \end{pmatrix} = \begin{pmatrix} \frac{1}{n_1} \sum_{\boldsymbol{X} \in R_1} Y \\ \frac{1}{n_2} \sum_{\boldsymbol{X} \in R_2} Y \\ \frac{1}{n_3} \sum_{\boldsymbol{X} \in R_3} Y \end{pmatrix}$$

that are exactly the $\hat{\mu}_{R_m}$ estimates for (1.1).

As consequence of this point of view, in case we assume the Normality of the error term we have the access to a large number of statistical tests (or we can construct confidence intervals) that can be employed in trees, for example to construct a pruning procedure based on a parametric test.

1.3 Extensions to regression trees

The advantages of regression trees over many other methods are their ability to include a relatively large number of independent variables and to identify complex interactions among these variables. In addition, regression trees can easy deal with

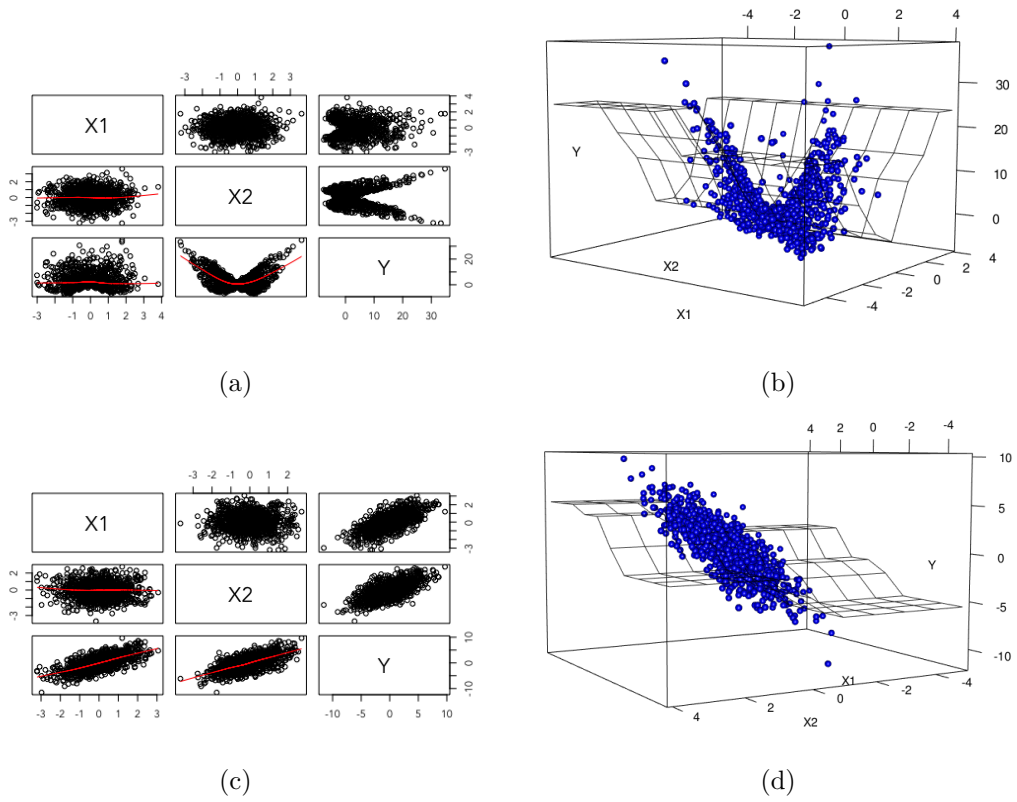


Figure 1.3: Trees struggle in modelling steep structures: (a) scatterplot of explanatory variables with a nonlinear dependence; (b) the regression tree surface relative to (a); (c) scatterplot of explanatory variables with a linear dependence; and, (d) the regression tree surface relative to (c).

missing data with trough the surrogate variables. Moreover, the simple graphic representation of the model contributes to its wide use in applications. In fact, the CART algorithm is largely employed in many scientific areas, such as medicine, genomic, social science, etc. Some example of application can be found in Crichton et al. (1997), Fan et al. (2006), Valera et al. (2007) or Lemon et al. (2003).

However, regression trees are not without drawbacks. They struggle in modelling steep structures, since they need to perform many splits to recreate a linear dependence. For example, in Figure 1.3 two different situations are illustrated: in (a) the dependence among Y and $\boldsymbol{X}$ is non linear, while in (c) the dependence is linear. The relative tree partitions in (b) for the non linear structure and (d) for the linear structure show that in the second case the tree algorithm performs much more splits than in the first one.

A further problem is the high variability of the trees. Often a small perturbation in the data can result in a very different series of splits that can compromise the utility of the trees in interpretation or prediction accuracy.

The depth of the tree is an important tuning parameter of the tree structure. Deep trees may result in very small sets for the estimates in the terminal nodes,

which may cause overfitting. In addition, the interpretability become difficult when trees are very large.

The tree structure can easy shows interactions between variables, but a large tree may give over-importance to these interactions.

Moreover, most of the algorithms for tree building are based on a greedy recursive partitioning, which is essentially a forward selection of variables. An error in the selection of the variables at the first stage will propagate in all the tree structure. This issue will be investigated in the next chapter. Another important drawback relative to the greedy search is that it is not guaranteed to find an optimal solution. Actually, in most of the cases it reaches a local solution.

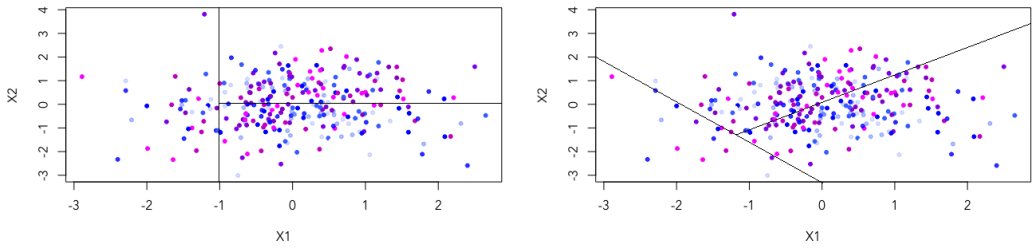
Many attempts have been made in order to try to overcome these limits, and the scientific community expanded the research on trees. In next paragraphs, some recent development on regression trees will be presented.

Trees via linear regression models

A difficulty in extending the greedy search of CART to piecewise multiple linear regression models is the increase in computational complexity. Alexander and Grimshaw (1996) avoid this difficulty by retaining the greedy search, but fitting a simple linear regression model within each terminal node. These trees are called *treed regression models*. The algorithm consists in the greedy search of the splitting variables, which minimize the MSE of a linear regression within the partition. For the linear regression, each independent variable is evaluated as regressor one at a time. According to the authors, treed regression models are more parsimonious than classical regression trees models because they result in shorter trees. Again, the disadvantage is the greedy nature of the algorithm. Moreover, these models evaluate the marginal effect of each variable rather than the conditional effect.

A related approach, suggested by Chaudhuri et al. (1994) and Loh (2002), is to avoid the use of the greedy search and fit a piecewise multiple linear regression model with standard statistical methods. See, for instance, *Smoothed and Unsmoothed Piecewise-Polynomial Regression Trees, SUPPORT* (Chaudhuri et al., 1994) and in *Generalised, Unbiased, Interaction Detection and Estimation, GUIDE* (Loh, 2002).

SUPPORT is an algorithm that fit a multiple linear regression model to the training data $(\boldsymbol{X}, \boldsymbol{Y})^{\texttt{train}}$. Each residual from this model is assigned to one class, according to its sign. Therefore two groups are identified, and two tests for differences in mean and variances of the two groups are performed for each X_j. The explanatory variable with the smallest p-value in one of the two tests is selected to be the splitting variable, and the splitting point is computed as the average of the two class means.



(a) (b)

Figure 1.4: Example of parallel (on the left) and oblique (on the right) partitions.

The *GUIDE* algorithm adopts a similar philosophy as SUPPORT. In fact, it has been proposed to solve the limitations of SUPPORT, such as the exclusion of categorical predictors and the inability to detect pairwise interactions. Therefore the algorithm proceeds first obtaining the residuals from a linear regression model, then performing a series of Chi-square tests among both continuous and categorical predictors. The most significant explanatory variable is the splitting variable. Both the sample mean and the sample median of the splitting variable can be employed as splitting point.

The use of regression models and statistical tests for the tree construction is the strength of these algorithms. Unfortunately the lack of implemented functions for statistical software as R or Pyton limited their use.

Oblique trees

As a further extension of CART, Murthy et al. (1994) defined a class of models called *oblique decision trees*. Essentially, oblique trees are a more general form of axis parallel trees. In fact, instead of axis parallel hyperplanes, the oblique trees use non parallels, or oblique, hyperplanes to partition $\mathcal{X}$.

An example of oblique partition can be find in the right panel of Figure 1.4. Here each node of the tree is delimited by an hyperplane that can take any orientation in the explanatory variable space. More precisely, the oblique splits can be defined as linear combinations of the $\boldsymbol{X}$ explanatory variables of the form

$$\sum_{j=1}^{p} \omega_j X_j + \omega_{p+1} \geq 0$$

where the $\omega_1 \dots \omega_{p+1}$ are real-valued coefficients.

Oblique trees are useful when the true decision boundaries are not aligned with the variable axes. However, one of the major drawback is the time complexity

of the algorithm, which makes the exhaustive search for the best oblique split impractical. Recently Wickramarachchi et al. (2016) proposed a new algorithm called *HHCART* to overcome this problem. Notice that this literature is specific to classification problems, whereas for regression problems the oblique trees have not been developed yet.

Trees via conditional inference

The *conditional inference trees, Ctree* have been proposed by Hothorn et al. (2006) as a method to tackle the problems of overfitting and selection bias toward explanatory variables with many possible splits or missing values. Since the recursive partitioning is done after a statistical test to asses the independence of each covariate and response variable, Ctree are so called unbiased recursive partitioning. The test uses a linear statistic to measure the association between Y and X_j, which distribution depends on the joint distribution between Y and X_j, not known in most practical circumstances. Anyway, under the null hypothesis the dependency can be recovered by fixing the covariates and conditioning on all possible permutations of the responses. This follows a test procedures known as permutation tests (Fisher, 1935). In the algorithm of *Ctree*, the conditional expectation and covariance under the null hypotesis given all permutations are computed following the theory of Strasser and Weber (1999). In addition, a multiple testing correction is implemented: if the strongest association measure passes a statistical threshold, a binary split is performed at that corresponding input variable. Otherwise the node is set as terminal node. The *Ctree* algorithm is summarized in Algorithm 2.

The advantage of Ctree lies in the use of a well-defined theory of permutation tests to select the splitting variable, which can reply to the need of a statistical approach to recursive partitioning that takes into account the distributional properties of the splitting criterion (White and Liu, 1994). However, notice that in case of continuous predictors and response, the permutation test is equivalent to a test on correlations. Therefore the splitting variable is chosen on the basis of the marginal correlation between the response and the explanatory variables, which may leads to trees that do not reflect the real structure of the data generating process (an example will be treated in the next chapter).

Trees as convex optimization problems

In recent years, the statistical community has been very interested in formulating predictive models as problems of convex optimization. Convexity constraints are familiar in many fields such as economics, statistics, operations research and financial engineering.

Algorithm 2: Pseudo Algorithm for conditional regression tree building

Data: $\{Y_i, \mathbf{X}_i\}$, $i = 1, \ldots, n$
Result: Conditional inference regression tree
Initialization: all observations in the ROOT NODE;
Stopping rule = 0;
Nodes = 0;
for *root node to nodes* **do**
  repeat
    for $j = 1$ *to* p **do**
      Test the independence between $\mathbf{X}$ and Y;
      Splitting variable $\leftarrow X_j$;
      for $s = 1$ *to* S **do**
        find split point s^* of X that minimize the MSE of the tree;
      end
    end
    partition the data in two nodes;
    compute the mean of Y within the node;
    Save $\rightarrow$ NODE
  until $\#i \leqslant 5 \rightarrow$ *Stopping rule* $= 1$;
end

Hannah and Dunson (2013) introduce the *convex adaptive partitioning*. They consider a regression model $Y = f_0(\boldsymbol{X}) + \epsilon$, where ϵ is a random variable with 0 mean, and f_0 is a convex function, that is $f_0(X_1) + (1-\delta)f_0(X_2) \geq f_0(\delta X_1 + (1-\delta)X_2)$ for every $X_1, X_2 \in \mathcal{X}$ and $\delta = (0, 1)$. Therefore, f_0 is estimated subject to the convexity (or concave, taking the negative of the convex function) constraint. The algorithm, called convex adaptive partitioning (CAP), models the convex function f_0 through a series of hyperplanes by an adaptive partitioning of $\mathcal{X}$ similarly to SUPPORT (Chaudhuri et al., 1994), described in Section 1.3. Thus, for a given partition of $\mathcal{X}$, it simply fit an OLS based on all the observations within the partition and choose the model that minimize the MSE. Then, the partitions are refined in two step. First, the set of candidate binary splits is generated within the existing partition. A linear model is fitted within the new partitions and the one that minimize the global MSE is chosen. Then, the new partition induced by hyperplanes is used to generate a new model. With these two simple rules, the adaptive partitioning and the refitting, the authors demonstrate to produce a gain comparing on treed regression models. Consistency of CAP has been demonstrated. However, the application of this method is indicated for complex problems described by functions in multiple dimensions.

Petersen et al. (2016) propose a non-greedy procedure whose fit has a block structure like CART. This method, called *CRISP*, predicts the response Y by assuming a model $Y = f(\boldsymbol{X}) + \epsilon$, where ϵ is a random variable with 0 mean. The

function f is estimated to be constant within the partition in bins of the variable space $\mathcal{X}$. The bins are constructed to be at the mean of a pair of covariates within the quantile range of the observed variables. This produces a grid of values Γ which is estimated by solving a convex optimization problem

$$\min_{\Gamma} \frac{1}{2} \sum_{i=1}^{n} (Y - \Omega(\Gamma, \boldsymbol{X}))^2 + \lambda(\Gamma)$$

where Ω extracts the element of Γ corresponding to the bin of the observation $\boldsymbol{X}$. The estimate is penalized by λ, which is chosen to encourage neighbour bins have the same values. This procedure is nice in the non-greedy estimation of the function. It would be interesting to extend the estimation of the function within the bins to other functions, for example the kernel functions. This could led to an estimated smoothed function instead of an estimated piecewise-constant function.

Trees via evolutionary algorithms

As an alternative to greedy recursive partitioning methods that estimate the regression model in a forward search, *evolutionary algorithms* (EA) can be adopted to obtain a globally optimal trees, as proposed in Grubinger et al. (2011).

EA are inspired by the principles of genetics and natural selection. In nature, individuals continuously evolving and constantly adapting to their living environment. In EA, each individual represents a candidate solution to a problem. At each generation, individuals are evaluated by a fitness function, and the best individuals have a high probability to be selected for reproduction. The operations proposed to the selected individuals are inspired by genetics, like mutation or crossover. More details on EA can be found in (Back, 1996).

Grubinger et al. (2011) propose *evtree*, an evolutionary algorithm to learning regression trees.

Recall the regression tree of (1.2) and define $\theta = (X_1, s_1, \ldots, X_{M-1}, s_{M-1})$ the vector of the splitting variables and splitting points associated with a tree with M terminal nodes, and Θ_M the product of all the possible combinations of the elements of θ. *Evtree* estimates a collection of trees which are simultaneously and iteratively modified in a stochastic way. Therefore, the overall parameter space is $\Theta = \bigcup_{M=1}^{M_{\max}} \Theta_M$, where $M_{\max}$ is the maximum number of parameters obtainable. The goal of the algorithm is to estimate the vector θ that optimize the trade-off between the prediction task and the complexity of the tree. In case of regression trees, it means to find the set of splitting variables and points which minimizes the MSE of the tree with a complexity parameter which penalize the complex tree structures.

The *evtree* algorithm starts from the initialization of each tree with a valid

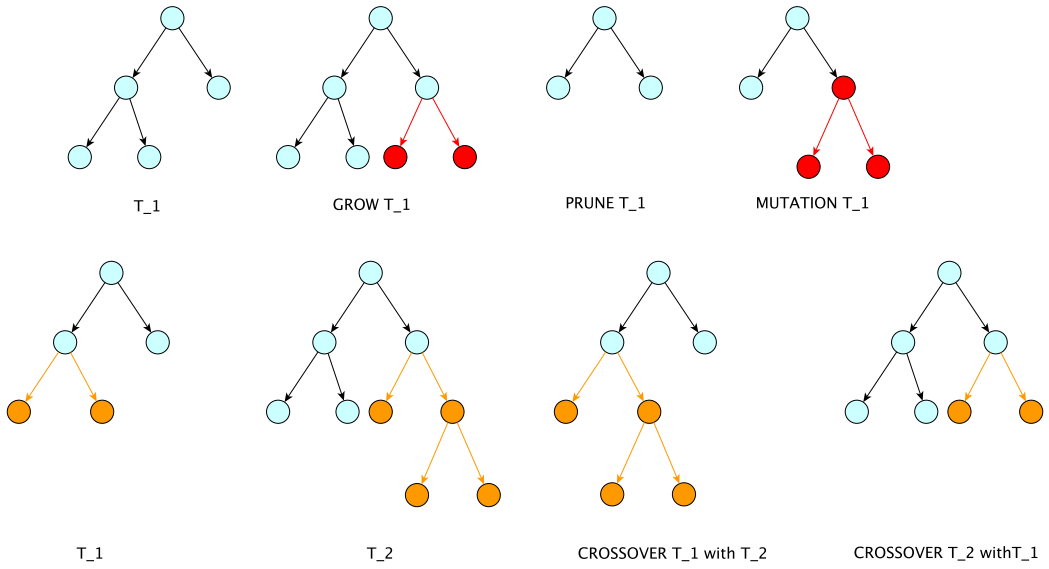


Figure 1.5: The variation operator in an *evtree*: grow, prune and mutation refers to moves that can be done to perturb a single tree, while crossover takes two trees and exchange the structure of one of their subtree.

randomly generated split rule in the root node. The first splitting variable X_1 is randomly chosen with uniform probability $\frac{1}{p}$. The split point s_1 is randomly chosen with uniform probability $\frac{1}{u-1}$, where u are distinct values of X_1. Then, at every iteration, each tree is selected one at a time to be modified by a variation operators. The variation operators are *split*, *prune*, *mutation*, and *crossover*. The first two rules are essentially similar to the splitting rule in definition 1.2.2 and the pruning rule of (1.3) for regression trees. Split selects a random terminal node and performs an additional split. As consequence, the number of terminal nodes of the tree become $M+2$. Prune selects a random non-terminal node and drop its two successor nodes. Mutation is divided in *major split rule mutation* and *minor split rule mutation*. In the first case a non-terminal node is randomly selected and its splitting variable and splitting point are changed. In the second case, only the splitting point is changed. Crossover is the unique move that select randomly two trees. Then, it exchanges two branches, or subtrees, between the selected trees. In Figure 1.5 all these possible moves are represented.

Finally, the accuracy of trees is evaluated by the MSE with a BIC-type complexity penalty

$$MSE_{BIC} = n \log(\text{MSE}) + \text{comp}_\theta \tag{1.6}$$

$$\text{comp}_\theta = \lambda \cdot 4 \cdot (M+1) \cdot \log n$$

where $M+1$ is the number of estimated parameters, taking into account the estimates of mean parameter for each of the terminal nodes and the constant

error variance term. With $\lambda = 0.25$ the criteria is equivalent to the BIC used by Fan and Gray (2005), up to a constant. A conservative value of $\lambda = 1$ is suggested by the authors due to the fact that the effective number of parameters is higher considering the selection of the splitting variables and points (see for discussion Gray and Fan (2008)).

Algorithm 3: Pseudo Algorithm for Evtree building

Data: $\{Y_i, \mathbf{X}_i\}$, $i = 1, \ldots, n$
Result: Evolutionary regression tree
nTrees= L;
niter=N;
Stopping rule: fixed % of trees stabilize over niter;
Initialization: all observations in the ROOT NODE;
construction of l trees with a randomly generated split rule in the root node;
for *niter* $= 1$ *to* N **do**
    for $l = 1$ *to* L **do**
        Alter a tree by selecting a variation operator;
        Evaluate each tree with the MSE-BIC measure;
        find split point s^* of X_j that minimize the MSE of the tree;
        Save Tree;
    end
end

Since the number of trees is decided a priori, during the evolution, only a fixed subset of trees can be kept in memory. *Evtree* employs a deterministic crowding approach, that is the comparison of each tree is limited only to its most similar successor. The algorithm stops when the quality of the 5% of the best trees stabilizes for 100 iterations, but not before 1000 iterations. The tree with the highest value of the evaluation function is returned. The process of *evtree* construction is summarized in Algorithm 3.

As a comment, notice that *Evtree* searches over all the parameter space Θ, that becomes too large even for medium size problems. A complete search would be computationally infeasible. Compared to the trees constructed by a forward stepwise search, this approach allows to search a solution in larger space of potential trees. On the other hand, the algorithm is very slow even for relative small number of explanatory variables, limiting its use.

Boosting trees

Generally, boosting algorithms combine weak learners into a single strong learner in an iterative way. When regression trees (Section 1.2) are chosen as weak learner, a boosted regression tree (Friedman, 2001) is defined as

$$T_{\texttt{Boost}}(\mathbf{X}) = \sum_{k=1}^{K} T(\mathbf{X}; \mathcal{R}_{k,Y}, \boldsymbol{\mu}_k) \tag{1.7}$$

where K is the number of trees fitted, each one with associated regions $\mathcal{R}_{k,Y}$ and mean parameters $\boldsymbol{\mu}_k$. The boosted tree model in (1.7) is induced in a forward stagewise manner (Hastie et al., 2009). At each step, the algorithm solves

$$\underset{(\mathcal{R}_{k,Y},\mu_k)}{\arg\min} \sum_{i=1}^{N} \Psi(Y_i, f_{k-1}(X_i) + \lambda T_k(\boldsymbol{X}, \boldsymbol{\mu}_k)) \tag{1.8}$$

where Ψ is the loss function. The solution to (1.8) is simply the regression tree that best predicts the current residuals $Y_i - f_{k-1}(X_i)$ plus the mean of these residuals in each corresponding region of the T_k tree. Note that each tree in (1.8) is penalized by a shrinkage factor to avoid overfitting. For more details on this procedure see Hastie et al. (2009). The boosting algorithm for regression trees is summarized in Algorithm 4.

Algorithm 4: Pseudo Algorithm of boosting for regression tree

Data: $\{Y_i, \mathbf{X}_i\}$, $i = 1, \ldots, n$
Result: Boosting for regression tree
Initialization: $T_0 = 0$;
Residuals $res_0 = Y - T_0$;
for $k = 1$ *to* K **do**
    Fit a tree to the training data $(res_{k-1}, \boldsymbol{X})$;
    Update $\hat{T}_k = \hat{T}_{k-1}(\mathbf{X}), +\lambda \hat{T}_k(\mathbf{X}; \mathcal{R}_{k,res_{k-1}}, \boldsymbol{\mu}_k)$;
    Compute the new residuals $res_k = res_{k-1} - \hat{T}_k$
end
Compute the boosted tree model estimate
$\hat{T}_{\texttt{Boost}}(\boldsymbol{X}) = \sum_{k=1}^{K} \hat{T}_k(\mathbf{X}; \mathcal{R}_{k,res_{k-1}}, \boldsymbol{\mu}_k)$

A well-known problem in the context of boosting (Freund, 2001) is the risk of overfitting. In fact, trees added at later iterations tend to impact the prediction of only a few cases, and make a minor contribution towards the prediction of all the remaining data.

Rashmi and Gilad-Bachrach (2015) propose the *Dropouts meet Multiple Additive Regression Trees, DART*, as a possible answer to the problem of overfitting in boosting trees. In fact, the most common approach employed also in Friedman (2001) or Friedman (2002) is to reduce the contribution of a new tree added by shrinkage factor. In DART the technique of droputs (utilized for example in Wager et al. (2013)) is employed. When computing the (1.8) at each iteration, only a random subset $k' < k$ of the existing ensemble of trees is considered. Moreover, when adding a new tree to the ensemble, a normalization step is performed. This

step consists in scaling the new tree by a factor $\frac{1}{k}$ such that the new tree will have the same order of magnitude as the dropped trees. This results in a balanced effect of the dropped trees together with the new tree.

1.4 Random forests

Breiman (2001) propose the random forests, an ensemble method that combines several individual regression trees such that each tree depends on a random vector of units sampled independently and with the same distribution for all trees in the forest. The rationale behind ensemble methods is that combining the prediction of many trees leads to a significant increase in the predictive performance as compared to the performance of a single tree.

The idea of random forests was inspired by *bagging*, or bootstrap aggregation (Breiman, 1996), that consists on drawing a collection of training sets from the population by a random selection without replacement, then building a separate prediction model using each training set and averaging the resulting predictions. The key concept of random split selection is find also in Ho (1998), where multiple trees are constructed in randomly chosen subspaces, and in Dietterich (2000), where at each node of the tree the split is random selected from a number s of best splits. Finally, the work that influenced substantially the ranfom forests idea was the paper of Amit and Geman (1997) about shape recognition of image data.

Definition 1.4.1. A **random forest** for regression is a predictor consisting of a collection of trees constructed on a bootstrapped sample of the original data using a random selection of the explanatory variables.

The algorithm of random forests begins with sampling with replacement n_h units from the original learning set: only the sampled units are employed for a tree construction. Each random sample reflects the same data generating process, but differs slightly from the original training sample because of random variation. Then, the growth of H trees is done as in CART, with the peculiarity that only a subset of dimension $l < p$ of variables is considered at each node of the construction of the tree. The random forests predictor is

$$T_{\mathtt{rf}}(\boldsymbol{X}) = T(\boldsymbol{X}_h^*; \mathcal{R}_{Y_{h,n^*}}, \mu_{h,n^*}) = \frac{1}{H}\sum_{h=1}^{H} T(\boldsymbol{X}_h^*; \mathcal{R}_{Y_{h,n^*}}, \boldsymbol{\mu}_{h,n^*})) \tag{1.9}$$

where H is the total number of trees in the forest, $\boldsymbol{X}_h^*$ denotes the random subset of explanatory variables sampled for each node of the h^{th} tree in the ensemble, $\mathcal{R}_{Y_h}$ and $\boldsymbol{\mu}_h$ are respectively the regions and the mean on the bootstrapped sample that defines the h^{th} tree. Therefore, the number of randomly preselected splitting variables and the total number of trees in the forest are parameters that affect

the stability of the results of random forests. The process of random forests construction is summarized in Algorithm 5.

Algorithm 5: Pseudo Algorithm for random forest building

Data: $\{Y_i, \mathbf{X}_i\}$, $i = 1, \ldots, n$
Result: Random forest
Initialization: $h = H$;
for $h = 1$ *to* H **do**
  Select n^* observations with or without replacement uniformly in $(Y, \boldsymbol{X})$;
  repeat
    Select uniformly, without replacement, a subset $l < p$ of predictors;
    Select the best split according to the regression tree procedure;
    Split the data in two nodes;
  until $\#leaves < n_{min}$;
  Compute the predicted value as the average of Y falling in each terminal node
end
Compute the random forest estimate $\hat{T}_{\texttt{rf}}(\boldsymbol{X})$

An important feature of random forests is their use of out-of-bag samples, the bootstrapped data which are not used to fit the trees, for the out-of-bag error estimates.

Definition 1.4.2. The **out-of-bag estimate, OOB** is calculated by predicting the real value for each observation in the train set $(X_i, Y_i)^{\texttt{train}}$ by using only the trees for which this observation was not included in the bootstrap sample.

As proved by Breiman (1996), the out-of-bag estimates are as accurate as using a test set with size equal to the training set. Therefore the use of the out-of-bag estimates removes the need of splitting the data in training and test set. The out-of-bag estimates are important also for their use in variable importance measures, described in the next section.

Variable importance

As regression trees, random forests can be used to rank the predictive importance of the variables. Two measures are available: the *Mean Decrease Impurity* (MDI, Breiman and Cutler (2003)) and the *Mean Decrease Accuracy* (MDA, Breiman (2001)).

Let X_j the variable for which the importance has to be evaluated. The MDI for X_j is defined by

$$MDI(X_j) = \frac{1}{H} \sum_{h=1}^{H} \sum_{t \in \mathcal{T}_h} q_t \Delta Dev(T)$$

where q_t is the fraction of observation that fall in the node t, $\mathcal{T}_h$ with $1 \leq h \leq H$ is the collection of trees in the forest and $\Delta Dev(T)$ represents the maximum decreasing in the deviance obtained by the best split, according to (1.2). The MDI computes the weighted decrease of MSE from splitting on the variable X_j, and average that quantity over the trees of the forest.

The second measure, the MDA, relies to the permutation principle and involves the out-of-bag estimates of definition 1.4.2. Let $\mathcal{OOB}_h$ be the out-of-bag sample of the h^{th} tree. Let $\mathbf{X}_{j,\texttt{perm}}$ be the input variable vector where the j^{th} variable has been permuted, and let $\mathcal{OOB}_{h,\texttt{perm}}$ be its corresponding out-of-bag sample. The MDA for X_j is defined by:

$$\begin{aligned} MDA(X_j) = \frac{1}{H}\sum_{h=1}^{H}\Bigg[\frac{1}{|\mathcal{OOB}|_{h,\texttt{perm}}}(Y - T_{\texttt{rf}}(\boldsymbol{X}_{j,\texttt{perm}}))^2\Bigg] - \\ - \Bigg[\frac{1}{|\mathcal{OOB}|_h}(Y - T_{\texttt{rf}}(\boldsymbol{X}))^2\Bigg] \end{aligned} \tag{1.10}$$

that is the average difference in accuracy of the out-of-bag versus permuted out-of-bag observations over the H trees is the variable importance measure for X_j.

1.5 Extensions of Random Forests

The popularity of random forests, beside their predictive power, is that splitting variables are chosen on random subsets: this make them applicable also in *small n large p* problems. Moreover, random forests are capable in fitting high-dimensional signals, both in terms of their stability and computational efficiency. As for regression trees, random forests have proven to be effective across many application areas. See for example Díaz-Uriarte and De Andres (2006), Lunetta et al. (2004) or Bureau et al. (2005). Interestingly, random forests have been extended to regression trees not based on the CART algorithm. For example, Menze et al. (2011) introduce random forests based on Oblique Trees, and Strobl et al. (2007) utilize conditional inference trees for the random forests construction. However, even if the performance of random forests is good in many contexts, the consistency has been demonstrated only in case of independent variables, see for example Biau et al. (2008), Ishwaran and Kogalur (2010), and Scornet et al. (2015). Moreover, the overfitting problem may be mitigated by the use of random forests (as well as every ensemble methods), but the difficulty to exploit smoothness in the surface they are estimating still persists.

In next paragraphs, some recent development on random forests will be presented.

Extra Trees

Geurts et al. (2006), propose the extremely randomized trees, *Extra-Trees (ET)*, a tree-based ensemble method that introduces randomization on both variable choice and cut-point during the splitting and tree construction. The trees of the ensemble are a collection of regression trees constructed by a top-down greedy search, as in CART. The innovation of this algorithm is that the selection of the split points is made fully at random. The choice of the explanatory variables is at random, as in random forests. ET builds trees whose structures are independent of the response variable of the learning sample. The ET model is

$$T_{\texttt{et}}(\boldsymbol{X}) = T(\boldsymbol{X}^*; \mathcal{R}^*_{Y_h}, \mu_h) = \frac{1}{H}\sum_{h=1}^{H} T(\boldsymbol{X}^*; \mathcal{R}^*_{Y_h}, \boldsymbol{\mu}_h)) \tag{1.11}$$

where the $\mathcal{R}^*_{Y_h}$ denotes the regions induced by the random split. The parameters that identify the ET are l, the number of variables randomly selected at each node, n_{min} the minimum sample size for splitting a node, and H the total number of trees. They play different roles: l determines the strength of the variable selection process, n_{min} controls the strength of averaging output noise, and the total number of trees controls the strength of the variance reduction of the ensemble model aggregation. The algorithm proceeds as follows. First, a random subsample η of dimension l of explanatory variable is drawn. Then, for each variable selected, a random split point is picked. Then, the splitting variable X_j is chosen on the basis of a score computed for each variable in η as

$$Score_j = \frac{Var(Y) - \frac{n_{X_j \geq s_j}}{n} Var(Y|X_j \geq s_j) - \frac{n_{X_j < s_j}}{n} Var(Y|X_j < s_j)}{Var(Y)} \tag{1.12}$$

that is the amount of variance reduction due to the split. The algorithm is summarized in Algorithm 6.

According to the authors, the rationale behind the ET is to reduce variance more strongly than the weaker randomization schemes used by other methods such as bagging or random forests. Moreover, in order to minimize the bias, the full original training set is used, rather than a bootstrap replicate as in random forests.

The idea of the double randomization in the splitting procedure is very interesting and could be insert in other kinds of models.

Reinforcement learning

Zhu et al. (2015) propose a new approach for the random forests construction called *reinforcement learning trees (RLT)*.

Algorithm 6: Pseudo Algorithm for Extra Trees building

Data: $\{Y_i, \mathbf{X}_i\}$, $i = 1, \ldots, n$
Result: Extra Trees
Initialization: $h = H$;
for $h = 1$ *to* H **do**
  repeat
    Select uniformly, without replacement, a subset $l < p$ of predictors, η;
    Select l split point randomly, one for each variable in η;
    Choose the split variable and split point that give $Score_{min}$ (1.12);
    Split the data in two nodes;
  until $\#leaves < n_{min}$;
  Compute the predicted value as the average of Y falling in each terminal node
end
Compute the extra tree estimate $\hat{T}_{\texttt{et}}(\boldsymbol{X})$

The proposed model is similar to random forests, with a special procedure of splitting variable selection and noise variable muting, done by a reinforcement learning mechanism (Sutton et al., 1998) at each internal node.

RLT starts with a first tree using a slight modification of the Extra Tree model (1.5). The modification consists in applying the model to a bootstrapped sample of the learning data as in random forests. Then, the split variable is chosen by the MDA variable importance measure in (1.10). Since searching for a strong variable becomes increasingly difficult moving down the nodes of the tree, a procedure of variable muting is implemented to prevent some noise variables from being considered as splitting variables. The variable muting consists of the allocation of variables in the muted set or the protected set. The muted set includes the variables with a low ranking of the variable importance. The muting rate is an important tuning parameter of the method, which allow to control the sparsity towards terminal nodes. The protected set includes variables used as splitting rules. Both sets can be updated during the process. The final splitting rule is a linear combination of the strongest variables. Therefore the reinforcement procedure consists in growing trees excluding noise variables moving down the depth of the tree.

According to the authors, this approach allows to concentrate the splitting rule search process only on the strong variables at the early stage of the tree construction while also reducing the number of candidate variables gradually towards terminal nodes. This results in a more sparse tree structure, that focuses on a smaller number of variables than a traditional tree-based model. In addition, the authors proved the consistency of the model in case of independent explanatory

variables.

Kernel regression forests

Scornet (2016) demonstrate how random forests can be rewritten as kernel estimators to obtain forests more interpretable and easier to analyse in terms of properties. Since several properties of random forests remain unexplained, kernel regression forests (KeRF) are shown to provide estimates more amenable to mathematical analysis. The KeRF model can be written as

$$T_{\texttt{KeRf}}(\boldsymbol{X}) = T(\boldsymbol{X}^*, \mathcal{R}_{Y_{h,n^*}}, \mu_{h,n^*}) = \frac{1}{\sum_{h=1}^{H} N(\mathcal{R}_{Y_{h,n^*}})} \sum_{h=1}^{H} T(\boldsymbol{X}^*; \mathcal{R}_{Y_{h,n^*}}, \boldsymbol{\mu}_{h,n^*})) \tag{1.13}$$

where the weights $N(\mathcal{R}_{Y_{h,n^*}})$ represents the number of observation in each region of the tree in the forest. The key idea in (1.13) lies in the weights applied to each observation; these weights are the number of times that an observation appears in the trees of the forests. As a consequence observations never sampled do not contribute to the prediction.

Another example of random forests with a weighting function applied to the trees of the ensemble can be find in Athey et al. (2018). They propose the *Generalized Random Forests*, which are based on the model in (1.4), except for the weights. In this case the forests are treated as a type of adaptive nearest neighbor estimator.

1.6 Bayesian Additive Regression Trees

Bayesian Additive Regression Trees, or BART (Chipman et al., 2010), are bayesian non-parametric regression models based on a sum of tree. The authors find their inspiration among the aforementioned ensemble methods as boosting (Section 1.3), random forests and bagging (Section 1.4). The essential idea is to regularize the fit by imposing a prior to each tree. This leads to small effects of individual trees as in the boosting models.

The BART model can be expressed as:

$$\boldsymbol{Y} = f(\boldsymbol{X}) + \boldsymbol{\epsilon} \approx T_1(\boldsymbol{X}; \mathcal{R}_Y, \boldsymbol{\mu}_1) + \cdots + T_h(\boldsymbol{X}; \mathcal{R}_Y, \boldsymbol{\mu}_m) + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim N(0, \sigma^2 \boldsymbol{I})$$

where there are h single regression trees, each composed by its structure defined by the partitions $\mathcal{R}_Y$ and the parameter at the terminal nodes μ_m. The total number of leaves in a tree is represented by m. The model fitting and the inference

are accomplished by a Bayesian backfitting MCMC algorithm that generates samples from the posterior distribution. Essentially the single regression trees that compose the BART have the same structure for the splitting rules as the trees described in CART. The sum of m terminal nodes value becomes the predicted value of the tree. For simplicity of notation, in the next paragraph I will refer to the tree structure with T instead of $T_h(\boldsymbol{X}; \mathcal{R}_Y, \boldsymbol{\mu}_m)$

The crucial point that distinguish BART from other ensemble methods is the underlying probability model assumed in BART. In fact, as a Bayesian model, BART consists of a set of priors for the trees structure and for the means of the leaves of the trees, and a likelihood for the observations in the terminal nodes.

The priors for the BART have three components: the tree structure itself, the parameters of the terminal nodes given the tree structure and the error variance of the error term, which is independent from the other two priors. Therefore the priors can be expressed as

$$\mathbb{P}(T_1, \ldots, T_h, \sigma^2) = \left[\prod_{h=1}^{H} \mathbb{P}(T_h)\right] \mathbb{P}(\sigma^2) = \left[\prod_{h=1}^{H} \prod_{m=1}^{M} \mathbb{P}(\mu_m | T_h) \mathbb{P}(T_h)\right] \mathbb{P}(\sigma^2) \quad (1.14)$$

In (1.14), the prior component $\mathbb{P}(T_h)$ affects the locations of the nodes within the trees, thus the depth of the tree and the splitting rules associated to the internal nodes of a tree. Nodes at depth d are nonterminal with a probability $D(1+d)^{-Q}$, where $D \in (0,1)$ and $Q \in [0, \text{inf}]$. This prior has the function to limit complexity structures of singles trees. For the splitting rules, the splitting variable is randomly selected, and its splitting value is randomly chosen among the available values via a discrete uniform distribution.

The prior component $\mathbb{P}(\mu_m | T_h)$ controls the leaf parameter. Typically it is assumed that $\mu_m | T_a \sim N(\mu_\mu, \sigma_\mu^2)$, where the expectation μ_μ is picked to be the range center $(Y_{min} + Y_{max})/2$ and the σ_μ is chosen empirically, so that the range center plus or minus 2 variances cover the 95% of the response values in the training set.

The final prior on the error variance is chosen to be $\sigma^2 \sim InvGamma(\varsigma/2, \varsigma(\varpi/2))$. The choice of hyperparameters ϖ and ς is data-driven to assign substantial probability to the entire region of plausible values of σ, in order to avoid overconcentration or overdispersion.

Along with the set of priors, BART specifies the likelihood of responses in the terminal nodes, that is assumed normal with the mean being the best guess in the leaf at the moment (i.e. in the current MCMC iteration) and variance being the best guess of the variance at the moment.

BART algorithm now has to generate draws from the posterior distribution

of $f(T_1, \ldots, T_H, \sigma^2 | \boldsymbol{Y})$. It employs a Gibbs sampler (Geman and Geman, 1984) with a form of *Bayesian backfitting* (Hastie et al., 2000) where the h^{th} tree is fitted iteratively, holding all other trees constant by using only the partial residuals:

$$Y_{Res_h} = \boldsymbol{Y} - \sum_{t \neq h} T_t(\boldsymbol{X}) \tag{1.15}$$

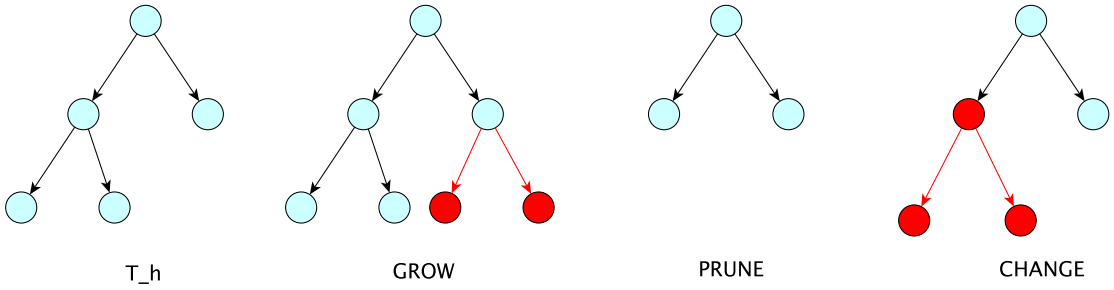


Figure 1.6: The alterations in a *BART*: grow consists in adding two child nodes at a node, prune consists in dropping two child nodes, and change consists in changing a split rule.

The Gibbs sampler proceeds iteratively drawing from (1.15) first the tree structure, then the parameter of the leaves. Finally, it sample the variance from the full conditional. For the tree structure, the algorithm proceeds by proposing a new tree from a perturbation of the current tree structure, and a Metropolis-Hastings step (Gelman et al., 1995) accept or reject the new tree. The possible alterations of the tree structure are growing terminal nodes by adding two child nodes at a node, pruning two child nodes or changing a split rule. Notice that change rule is allowed only on singly internal nodes of trees, that are nodes which both children nodes ere terminal. In Figure 1.6 these moves are illustrated.

For a detailed description of the Metropolis-Hastings step see Kapelner and Bleich (2016). In Algorithm 7, the pseudo-code for the BART building is reported.

Algorithm 7: Pseudo Algorithm for BART building

Data: $\{Y_i, \mathbf{X}_i\}$, $i = 1, \ldots, n$

Result: Bayesian Additive Regression Tree

Initialization: Residuals $Y_{Res_h} = Y$;

Tree: two terminal nodes;

for $u = 1$ *to* U **do**

  for $h = 1$ *to* H **do**

    1. Generate a proposal tree $T^*$by choosing from one of the following moves:
       - GROW;
       - PRUNE;
       - CHANGE.
    2. Accept/Reject the proposal by the Metropolis ratio and set T_h;
    3. Update the leaves parameters by drawing from $(\boldsymbol{\mu}_m | T_h, S_h, \sigma^2)$;
    4. Update residuals Y_{Res_h}

  end

  Update σ^2 by drawing from $(\sigma^2 | T_1, S_1, \ldots, T_H, M_H, \boldsymbol{\epsilon})$;

  Set $\hat{Y}_u = \sum_{h=1}^{H} T_h(\boldsymbol{X}, \boldsymbol{\mu}_m)$

end

Estimate $E(f(\boldsymbol{X})|Y) = \frac{1}{U} \sum_{u=1}^{U} \hat{Y}_u$

Chapter 2

Pitfalls in variable selection for tree–based models

2.1 Introduction

The models described in Chapter 1 are appealing as they can deal with both non linear relationships and interactions and have, apparently, an easy interpretation.

Regression trees and random forests are sometimes also used to discover the relevant variables, determinants or risk factors, which are sometimes interpreted as the variables directly influencing the response. This is encouraged by the measures of variable importance, where predictive importance is misinterpreted as a generative importance. Some examples of a misleading interpretation of variable importance are found in Karaolis et al. (2010); Ma et al. (2007); Fan et al. (2006).

This raises the question of whether an essentially predictive model can give also information concerning the data generating process. This problem has been also addressed in the discussion of Breiman et al. (2001) and by Shmueli et al. (2010). In the literature (see for instance Wolfson and Venkatasubramaniam, 2018), the discrepancy between a learning algorithm and the underlying *true* data generating process is often ascribed to overfitting. The findings here presented discredit this hypothesis detecting another source of incongruity. In this chapter, I discuss some data generating processes based on directed acyclic graphs for continuous variables and I show that the claimed important variables can be systematically incorrectly identified. This issue occurs especially when there are background variables strongly influencing intermediate variables with a direct effect on the response. This situation may lead to an inaccurate interpretation of variable importance in applications.

In next sections some examples of data generating process for which the variable importance can give erroneous information about the true relationship among variables are described.

2.2 Identification of relevant variables: an example

As mentioned above, regression trees can provide predictions where it is possible to identify which of the explanatory variables are most relevant to predict a response. Consider now data generated by the directed acyclic graph (DAG) in Figure 2.1.

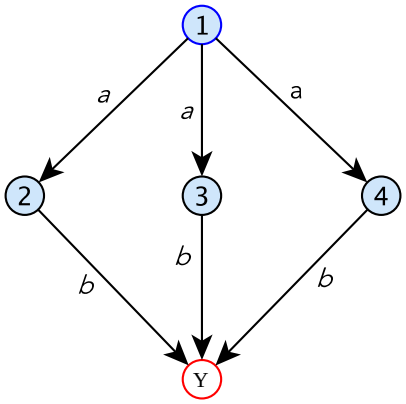


Figure 2.1: The generating process according to a DAG with three paths from X_1 to Y.

Specifically, the variables (X_1, X_2, X_3, X_4, Y) are generated by the equation

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -a & 1 & 0 & 0 & 0 \\ -a & 0 & 1 & 0 & 0 \\ -a & 0 & 0 & 1 & 0 \\ 0 & -b & -b & -b & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ Y \end{bmatrix} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_Y \end{bmatrix} \tag{2.1}$$

where $(\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_Y)' \sim N(0, I_5)$. The (2.1) corresponds to the following regression models

$$\begin{aligned} X_j &= aX_1 + \epsilon_j \qquad \text{for } j = 2, \ldots, 4 \\ Y &= bX_2 + bX_3 + bX_4 + \epsilon_Y. \end{aligned}$$

As a consequence, model (2.1) implies that X_2, X_3, X_4 have a direct influence on Y, while X_1 has only an indirect influence on it. Later the matrix of negative regression coefficients on the left side of (2.1) will be denoted by L.

Interestingly, for data generated by model (2.1), for certain values of a and b, both the CART algorithm and the random forests indicate the background variable X_1 as the most relevant to predict the response Y. As a matter of fact, the predictive strategy departs from the generative model. In fact, an explanatory variable which has an indirect effect on the response can sometimes be the most relevant to predict the response variable. Differently, if the researcher is interested

in discovering the variables that have a direct effect on the response, in these cases CART and random forests variable importance could end up with misleading conclusions. It seems evident that variable importance provides a measure of relevance summing up both direct and indirect effects. This confirms that interpreting the variable importance as representative of the direct effects in a generative model sense, as commonly done in some research fields, is incorrect. See also Breiman et al. (2001) and Shmueli et al. (2010). In the following paragraph, it is provided a further discussion of this issue in the case of Gaussian distributions.

First split criterion in the CART algorithm

Let $(X_1, \ldots, X_p, Y)$ be a random vector with a joint Gaussian distribution such that (Y, X_j), $j = 1, \ldots, p$, have a bivariate Gaussian distribution with zero means and correlations ρ_{Yj}. The basic CART algorithm for regression trees is based on a greedy search of the best piecewise constant function on a binary partition of the explanatory variable space. At the first step, the algorithm considers all the variables $X_j, j = 1 \ldots, p$ and searches the variable with the minimal MSE after a dichotomization. This corresponds to finding which choice of j and s_j minimizes the MSE of the one-factor regression model

$$Y_i = \beta_1 \mathbb{I}_{\{X_{ij} \leq s_j\}} + \beta_2 \mathbb{I}_{\{X_{ij} > s_j\}} + \varepsilon_i.$$

Now, under the assumption of a joint Gaussian distribution, it can be shown that the MSE, is an increasing function of the marginal correlation ρ_{Yj} between Y and X_j for each possible cut point s. At this aim, consider the two following results.

Proposition 2.2.1. *Let $(X_1, \ldots, X_p, Y)$ be a random vector with a joint mean zero multivariate normal distribution, and let ρ_{Yj} the marginal correlations between Y and X_j for $j = 1, \ldots, p$. Given the dichotomized variable*

$$W = \begin{cases} 1 & X_j > s_j \\ 0 & X_j \leq s_j \end{cases}$$

then the conditional expected value of $Y \mid W$ is given by

$$\mathbb{E}[Y \mid W = w] = \rho_{YX_j} \, \phi(s_j) \left(\frac{w}{\Phi(-s_j)} + \frac{w-1}{\Phi(s_j)} \right) \tag{2.2}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ are the density and distribution function, respectively, of a standard normal distribution.

Proof. As (X, Y) are bivariate Gaussian distribution with zero means, unit vari-

ances and correlation ρ_{YX_j}. Then, one can write that

$$Y = \rho_{YX_j} X + \varepsilon_{Y.X}, \tag{2.3}$$

where is $\varepsilon_{Y.X}$ is has a gaussian distribution with zero mean and variance $1 - \rho^2_{YX_j}$ and it is uncorrelated with X. According to Cox and Wermuth (1992), the conditional expectations of X given A can be written as

$$\begin{aligned} \mathbb{E}\left[X \mid W = 1\right] &= \mu_X^1(s) = \phi(s)/\Phi(-s) \\ \mathbb{E}\left[X \mid W = 0\right] &= \mu_X^0(s) = -\phi(s)/\Phi(s) \end{aligned}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ are the standard Gaussian density and cumulative function respectively. Now, for (2.3), $\mathbb{E}[Y \mid W] = \rho_{YX_j}\mathbb{E}[X \mid W]$. Consequently,

$$\mathbb{E}[Y \mid W = w] = \rho_{YX_j}\mu_X^1 \cdot w + \rho_{YX_j}\mu_X^0 \cdot (1-w) = \rho_{YX_j}\,\phi(s)\left(\frac{w}{\Phi(-s)} + \frac{w-1}{\Phi(s)}\right). \qquad \square$$

Figure 2.2 reports the behaviour of this expected value for several values of s and ρ_{YX_j}.

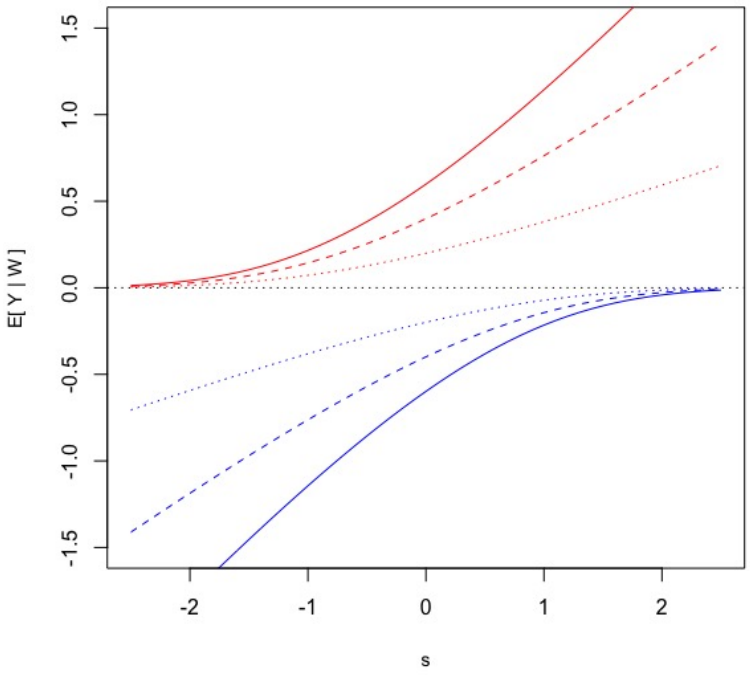


Figure 2.2: Plot of $\mathbb{E}[Y \mid W]$ when $w = 0$ (blue) and $w = 1$ (red), for $\rho_{YX_j} = 0.75$ (solid line), 0.5 (dashed line) and 0.25 (dotted line).

Proposition 2.2.2. *The MSE of the prediction of Y given the binary variable W is*

$$\text{MSE} = \mathbb{E}\left[(Y - \mathbb{E}[Y \mid W])^2\right] = \mathbb{E}(Y^2) - \rho^2_{Yj}\,\frac{\phi(s_j)^2}{\Phi(s_j)\Phi(-s_j)}. \tag{2.4}$$

As $\mathbb{E}(Y^2)$ is a constant and the factor multiplying ρ^2_{Yj} is a function only of s, the minimum of the MSE for a fixed s corresponds to the maximum of ρ^2_{Yj}. The variable X_j with the largest ρ^2_{Yj} is selected and dichotomized, thus forming two classes.

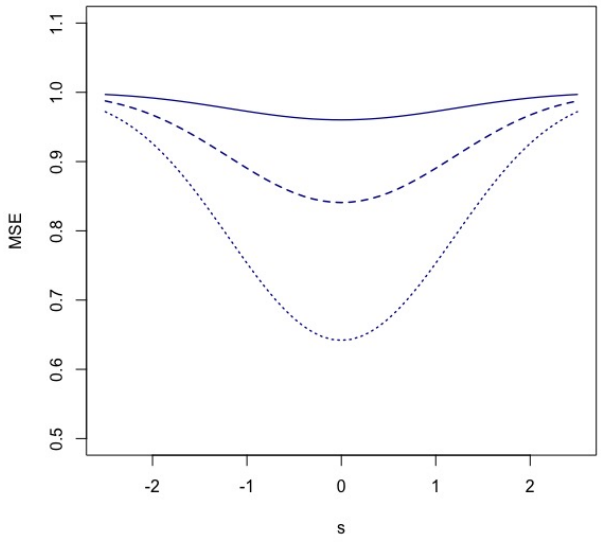


Figure 2.3: Plot of MSE at the first split for different values of s, when ρ_{YX_j} is 0.25 (solid line), 0.5 (dashed line) and 0.75 (dotted line).

As can be noticed in Figure 2.3, the marginal correlation between Y and an explanatory variable uniformly determines the minimum value of the MSE, ruling the choice of the first split.

At the second step, a similar criterion is used to select a variable and an optimal split within the two previous regions. A high squared correlation between the response and a specific dichotomized variable is sometimes persistent within the classes in further steps and this may lead to an increase in variable importance.

When the data generating process follows that described in (2.1), depicted in Figure 2.1, the correlation matrix of $(X_1, \ldots, X_4, Y)$ can be obtain by the matrix of the regression coefficients as

$$R = \begin{pmatrix} 1 & \frac{a}{\sqrt{a^2+1}} & \frac{a}{\sqrt{a^2+1}} & \frac{a}{\sqrt{a^2+1}} & \frac{3ab}{\sqrt{c}} \\ \frac{a}{\sqrt{a^2+1}} & 1 & \frac{a^2}{a^2+1} & \frac{a}{\sqrt{a^2+1}} & \frac{3a^2b+b}{\sqrt{c}\sqrt{a^2+1}} \\ \frac{a}{\sqrt{a^2+1}} & \frac{a^2}{a^2+1} & 1 & \frac{a^2}{a^2+1} & \frac{3a^2b+b}{\sqrt{c}\sqrt{a^2+1}} \\ \frac{a}{\sqrt{a^2+1}} & \frac{a^2}{a^2+1} & \frac{a^2}{a^2+1} & 1 & \frac{3a^2b+b}{\sqrt{c}\sqrt{a^2+1}} \\ \frac{3ab}{\sqrt{c}} & \frac{3a^2b+b}{\sqrt{c}\sqrt{a^2+1}} & \frac{3a^2b+b}{\sqrt{c}\sqrt{a^2+1}} & \frac{3a^2b+b}{\sqrt{c}\sqrt{a^2+1}} & 1 \end{pmatrix},$$

where $c = 9a^2b^2 + 3b^2 + 1$. The corresponding matrix of the partial correlation

coefficient is

$$K = \begin{pmatrix} 3a^2+1 & -a & -a & -a & 0 \\ -a & b^2+1 & b^2 & b^2 & -b \\ -a & b^2 & b^2+1 & b^2 & -b \\ -a & b^2 & b^2 & b^2+1 & -b \\ 0 & -b & -b & -b & 1 \end{pmatrix}.$$

where it results that $Y \perp\!\!\!\perp X_1 \mid X_2, X_3, X_4$ is the only conditional independence in the undirect graph obtained by moralizing the diamond graph. On the other side, from the correlation matrix R, one can derive that $Y \not\perp\!\!\!\perp X_1$ marginally whenever both a and b are different from zero. The marginal correlations are

$$\rho_{Y1} = \frac{3ab}{\sqrt{c}}, \quad \rho_{Yj} = \frac{3a^2b+b}{\sqrt{c}\sqrt{a^2+1}}, \quad j = 2,3,4 \tag{2.5}$$

where $c = 9a^2b^2 + 3b^2 + 1$. Then, the CART algorithm at the first step chooses the variable X_1 when $\rho_{Y1} > \rho_{Yj}$, $j = 2,3,4$ or, equivalently, when

$$(b < 0 \text{ and } a < 1/\sqrt{3}) \;\text{ or }\; (b > 0 \text{ and } a > 1/\sqrt{3}). \tag{2.6}$$

Therefore, in these situations the algorithm chooses for the first split the variable X_1, which, as a matter of fact, does not directly influence Y.

Now, at the second step the CART algorithm looks for the value of k, with $k = 1, \ldots, p$ and s_k such that one of the two following one-factor regression model

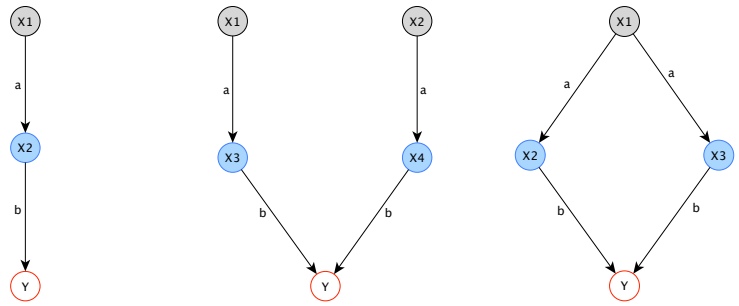
$$Y_i = \beta_1 \mathbb{I}_{\{X_{ij} \le s_j\}} \mathbb{I}_{\{X_{ik} \le s_k\}} + \beta_2 \mathbb{I}_{\{X_{ij} \le s_j\}} \mathbb{I}_{\{X_{ik} > s_k\}} + \beta_3 \mathbb{I}_{\{X_{ij} > s_j\}} + \varepsilon_i,$$

$$Y_i = \beta_1 \mathbb{I}_{\{X_{ij} \le s_j\}} + \beta_2 \mathbb{I}_{\{X_{ij} > s_j\}} \mathbb{I}_{\{X_{ik} \le s_k\}} + \beta_2 \mathbb{I}_{\{X_{ij} > s_j\}} \mathbb{I}_{\{X_{ik} > s_k\}} + \varepsilon_i$$

provides the minimum MSE. Consequently, the first split choice has an important role in the final tree as the first dichotomized variable is multiplied to each other subsequent splitting variables. The final tree is therefore substantially depending on the first split, which is chosen (see Proposition 2.2.2) on the most marginally (and not conditionally) explanatory variable. A similar behaviour can be found in a generating process involving only binary variables. This argument applies to the trees forming a random forest, where the first splitting variable selected is the most marginally explanatory within the randomly selected variables. This is confirmed in the Monte Carlo simulations in Section 2.3.

Interesting situations can be found in data generating processes similar to those presented in Figure 2.4. Both the graph Fig. 2.4(a)-(b) represents the case of a single path from the background variables to the response Y. With these data generating process, the CART selects as most important variable the one with the direct effect on the response if b is not too weak, as no values of a and b guarantees a

dominant marginal correlation between the background variables and the response. The graph Fig. 2.4(c) represents the case of two paths from the ancestor X_1 to the response Y. With this data generating process, the CART algorithm tends to pick at random (uniformly) the three explanatory variables for the first split, selecting about the 25% of the times the variable with an indirect effect on Y. However, in all the three data generating processes, the variable importance measure will give misleading results as a increases, given to the background variables about the same amount of importance as the variables with direct effect on the response.



(a) (b) (c)

Figure 2.4: The generating process according to DAGs with one or two paths from X_1 to Y.

2.3 Monte Carlo study

Variable importance measures are defined specifically for each tree-based algorithm and typically measure the predictive importance of variables. To compare the variable importance measures I chose the following algorithms (with in parentheses the R package used):

- CART, Breiman et al. (1984) (`rpart`)
- Random forests (RF), Breiman (2001) (`randomForest`)
- Conditional random forests (CRF), Hothorn et al. (2006) (`party`)
- Reinforcement Learning Trees (RLT), Zhu et al. (2015) (`rlt`)
- Bayesian additive regression trees (BART), Chipman et al. (2010) (`bartMachine`)

To show how the variable importance measures can be misleading, I conducted a minimal simulation with $N = 1000$ data sets with size $n = 1000$ generated from model (2.1). The data sets are sampled from a multivariate normal distribution with zero mean and precision matrix $\Sigma^{-1} = L^T L$ taking $a = b = 3$. Table 2.1

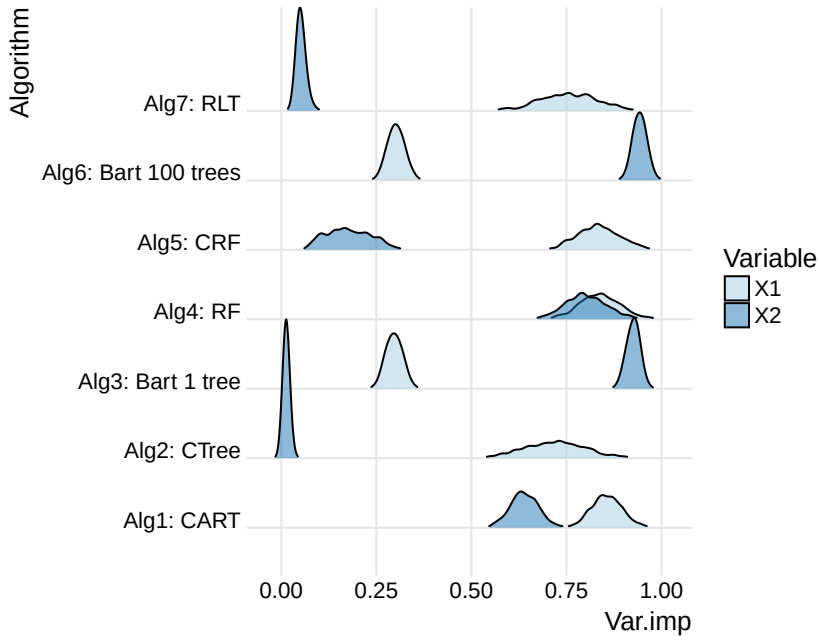


Figure 2.5: Monte Carlo distributions of rescaled variable importance measures of X_1 and X_2 in the 5 algorithms (CART, Ctree, RF, CRF, BART) for simulated data generated according to the DAG of Figure 2.1.

reports the averages and standard deviations of the raw variable importance measures obtained by growing trees with default settings and no pruning.

Table 2.1: Monte Carlo averages and standard deviations (in parentheses) of variable importance measures for simulated data generated according to the DAG of Figure 2.1.

	X_1	X_2	X_3	X_4
CART	704 091.2	525 987.4	526 636.9	526 591.6
	(33801.7)	(31162.3)	(31258.9)	(30479.0)
RF	194 582.0	186 209.6	186 190.0	186 160.5
	(12571.1)	(12816.4)	(12046.2)	(12221.4)
CTREE	998.9	18.3	18.4	18.2
	(116.3)	(5.8)	(6.0)	(5.6)
CRF	412.8	87.7	90.9	89.3
	(27.8)	(28.5)	(29.3)	(30.4)
RLT	41.8	2.9	2.9	2.9
	(4.4)	(0.6)	(0.6)	(0.6)
BART	0.097	0.301	0.301	0.301
(1 TREE)	(0.007)	(0.005)	(0.006)	(0.006)
BART	0.096	0.301	0.301	0.301
(100 TREES)	(0.007)	(0.006)	(0.006)	(0.005)

In Figure 2.5 are shown the results of the simulation as Monte Carlo distribu-

tions of the variable importance measures for the six algorithms. The comparison is limited to X_1 and X_2 as X_3, X_4 have a similar behaviour. To ensure comparability the measures are rescaled between 0 and 1 by dividing by the maximum observed value of the variable importance within each algorithm.

Figure 2.5 highlights that all the algorithms based on a greedy search fail to discover the direct influence of variables X_2, X_3, X_4 on Y and select as the most important variable X_1. With a difference: the classical random forest algorithm essentially gives the same importance to X_1 and X_2 while the conditional variable importance of CRF algorithm sharply separates X_1 from X_2 in the wrong direction. Moreover, also RLT selects as the most important variable X_1, even if its variable importance has been proven to be consistent for independent covariates.

On the other hand, BART correctly separates the variables in the right direction and gives more importance to variable X_2 as an explanatory variable.

Another simulation study has been conducted with data set generated from the the DAG in Figure 2.4(c), as a less adverse example. Table 2.2 reports the results according to this data generation process of the variable importance measures for the five algorithms: CART, RF, Ctree, CRF and RLT. Also in this case both CART and random forests fail to discover the direct influence of variables X_2, X_3 on Y and give the same importance to X_1, X_2 and X_3. Conversely, Ctree, CRF and RLT in this case select correctly $X2$ and X_3 as most important variables. Again, BART correctly separates the variables in the right direction and gives more importance to variables X_2, X_3 as explanatory variables.

It is therefore evident how dangerous is to interpret the variables selected by algorithms based on greedy search as representative of the generative model. In addition, the CART algorithm is also used to identifies mutually exclusive and exhaustive subgroups of population. The correct interpretation of these subgroups is that units in the same subgroup share the same prediction of the response value. However, the variables driving the individual stratification can be somehow linked to the true risk factors, while not being a risk factor by themselves.

The pitfalls described here are probably due to the greedy search used in most of the tree-based algorithms. The algorithms I am proposing in the next chapter will avoid this pitfalls.

Table 2.2: Monte Carlo averages and standard deviations (in parentheses) of variable importance measures for simulated data generated according to the DAG of Figure 2.4(c).

	X_1	X_2	X_3
CART	256 166.3	256 753.1	260 588.9
	(22 979.3)	(26 802.8)	(26 944.4)
RF	112 092.3	114 561.6	114 573.3
	(6 881.6)	(6 626.8)	(17 000.8)
CTREE	38.5	125.4	128.3
	(77.7)	(56.3)	(57.5)
CRF	50.8	128.7	129.4
	(38.5)	(31.7)	(31.8)
RLT	0.3	40.2	40.2
	(0.2)	(5.6)	(5.5)
BART	0.115	0.443	0.443
(1 TREE)	(0.008)	(0.007)	(0.007)
BART	0.114	0.443	0.443
(100 TREES)	(0.001)	(0.001)	(0.001)

Chapter 3

Semilinear regression trees

3.1 Introduction

In Chapter 1, the difficulty of modelling linear structures for regression trees has been mentioned. Indeed, the algorithms handle linear relationships between the response and the covariate by approximating the structure of dependence with long trees. In other words, to recover a simple type of relation among variables as the linear dependence, a regression tree will perform lots of splits which result in a complicated tree representation.

Chapter 2 dealt with the problem of the variable selection in tree-based models, highlighting how the variable importance measures of these kinds of model can give erroneous information about the generating process underlying the data structure. This issue relates to the different goal in analysing data, the generative and the predictive modelling as stated by Breiman et al. (2001) and Shmueli et al. (2010). When analysing data with prediction goal, a model would be able to accurately predict the response variable for new statistical units. When the goal is generative, a model should be able to capture the information underlying the data generating process.

Out of the ordinary *technological* usage, machine learning techniques are increasingly applied in several scientific domains. Alongside this spread, an interesting debate has arisen on predictive machine learning models transparency and interpretability. See for instance Vellido et al. (2012), Doshi-Velez and Kim (2017), Guidotti et al. (2018), among others. As noted by Doshi-Velez and Kim (2017), in machine learning the need of a definition and rigorous evaluation of interpretability is urgent and it represents an open scientific challenge. The European Union's General Data Protection Regulation (AA.VV., 2016) introduces the right for all individuals to obtain meaningful explanations of the logic involved in an algorithm when this algorithm makes decisions automatically.

In this chapter I study a class of tree-based models that can balance the pre-

dictive and the generative point of view. The interest is to model jointly linear and non linear relationships between the response and the covariates, taking into account the requirement of interpretability. For these reasons I propose a model that is a sum of a linear component and a tree, here called *Regression Tree model (SRT)*.

In the literature, examples on this kind of models can be found in Dusseldorp et al. (2010) and Liu et al. (2014). Dusseldorp et al. (2010) propose a new algorithm called *STIMA* to estimate a model called *regression trunk model* (Dusseldorp and Meulman, 2004), which integrates a multiple regression model with a regression tree. In Liu et al. (2014), the authors developed a new graphical model, called *sparse tree-embedded graphical model*, which is able to capture both linear and non linear associations among predictors. The construction of the associations of the graphical model is based on an integration of a generalized linear model with a regression tree.

I propose two different kinds of model estimation procedures: a two-stage estimation procedure based on a backfitting algorithm (Buja et al., 1989) and an estimation procedure based on an evolutionary algorithm (Grubinger et al., 2011).

The chapter is organized as follows: in Section 3.2 I propose the Semilinear Regression Tree model, while Sections 3.3 and 3.4 depict the two model estimation procedures for SRT.

3.2 Semilinear Regression Tree model

In this section I am going to study a class of model, here called Semilinear Regression Tree model $T_{SRT}(\boldsymbol{X})$, that is characterized by a the sum of a linear regression and a regression tree. Let $\boldsymbol{X} = (X_1, \ldots, X_p)$ be a vector of explanatory variables and Y be a response variable, and suppose to observe an *iid* sample of n units from this population. I consider modelling $\mathbb{E}(Y|\boldsymbol{X})$ by $T_{SRT}(\boldsymbol{X})$, that is

$$\begin{aligned}\mathbb{E}(Y|\boldsymbol{X}) = T_{SRT}(\boldsymbol{X}) = T(\boldsymbol{X}; \boldsymbol{\beta}, \mathcal{R}_Y, \boldsymbol{\mu}) = \boldsymbol{\beta}_1\boldsymbol{X}_1 + \cdots + \boldsymbol{\beta}_p\boldsymbol{X}_p + T(\boldsymbol{X}; \mathcal{R}_Y, \boldsymbol{\mu}) = \\ = \boldsymbol{\beta}_1\boldsymbol{X}_1 + \cdots + \boldsymbol{\beta}_p\boldsymbol{X}_p + \sum_{m=1}^{M} \mu_{R_m}\mathbb{I}_{(\boldsymbol{x}_i \in R_m)}.\end{aligned} \tag{3.1}$$

The model is characterized by a first part that is a linear component without intercept, with $\beta_1, \ldots, \beta_p$ unknown parameters associated with the X_p explanatory variables. The second part of the model is a regression tree, with μ and $\mathcal{R}_Y$ unknown parameters. I propose two estimation procedures of the unknown parameters, one iterative and one simultaneous. These procedures will be presented

in the next sections.

Notice that the model in (3.1) is a special case of the partial linear model (Härdle et al., 2012) that employs a tree model for the nonparametric part. The partial linear model is a semiparametric model largely employed in the econometric field. Suppose to separate the vector of the explanatory variables $\boldsymbol{X} = (X_1, \ldots, X_p)$ into $\boldsymbol{U} = (U_1, \ldots, U_p)$ and $\boldsymbol{W} = (W_1, \ldots, W_p)$. The regression of Y on $\boldsymbol{X} = (\boldsymbol{U}, \boldsymbol{W})$ is assumed to be

$$\mathbb{E}(Y|\boldsymbol{U}, \boldsymbol{W}) = \boldsymbol{U'\beta} + f(\boldsymbol{W}) \tag{3.2}$$

where $f(\cdot)$ is an unknown function of the vector $\boldsymbol{W}$. Thus the partial linear model is a sum of a purely parametric part $\boldsymbol{U'\beta}$ and a purely nonparametric part $f(\boldsymbol{W})$. By assuming a regression tree $T(\boldsymbol{X}; \mathcal{R}_Y, \boldsymbol{\mu})$ for the nonparametric part, the SRT model of (3.1) is obtained. This specification of the model will be useful for the two-stage estimation procedure of Section 3.3.

Model (3.1) differs from a regression tree because of the inclusion of the linear part. Instead of a piecewise constant estimate, a piecewise-linear approximation of the conditional expectation is obtained, with slopes capturing the main direction of the dependence. This aspect makes this model particularly useful for quasi-linear dependence shapes. Two examples in three dimensions of the surface estimated by the classical regression trees and the Semilinear regression trees on two different simulation data are illustrated in Figure 3.1.

The SRT model differs also from treed regression model of Section 1.3, because here there is only one model estimated for all the regions, while in treed regression model several regressions, one for each region, have to be estimate.

The rationale behind this model is to capture separately the linear and the non linear effect of the covariates through the estimate of a single model. The linear part will capture the main direction of dependence, while the tree part will capture interactions and non linearity. This is the crucial aspect of my proposal, and not only important to recover the dependence structure of the data generation process. In fact, the choice to sum an additive linear part to a tree component helps to keep small the tree, with few splits describing interactions and non linear terms. This point is particularly important for the interpretability of the proposed model. The estimates of the β parameters are simply interpreted as in the linear regression model, and the tree structure gives an immediate representation of the interaction terms, which are not complicated to interpret given that the tree will be possibly shorter.

In the literature, some particular case of the more general model proposed in (3.1) can be found in Dusseldorp et al. (2010) and Liu et al. (2014). In the

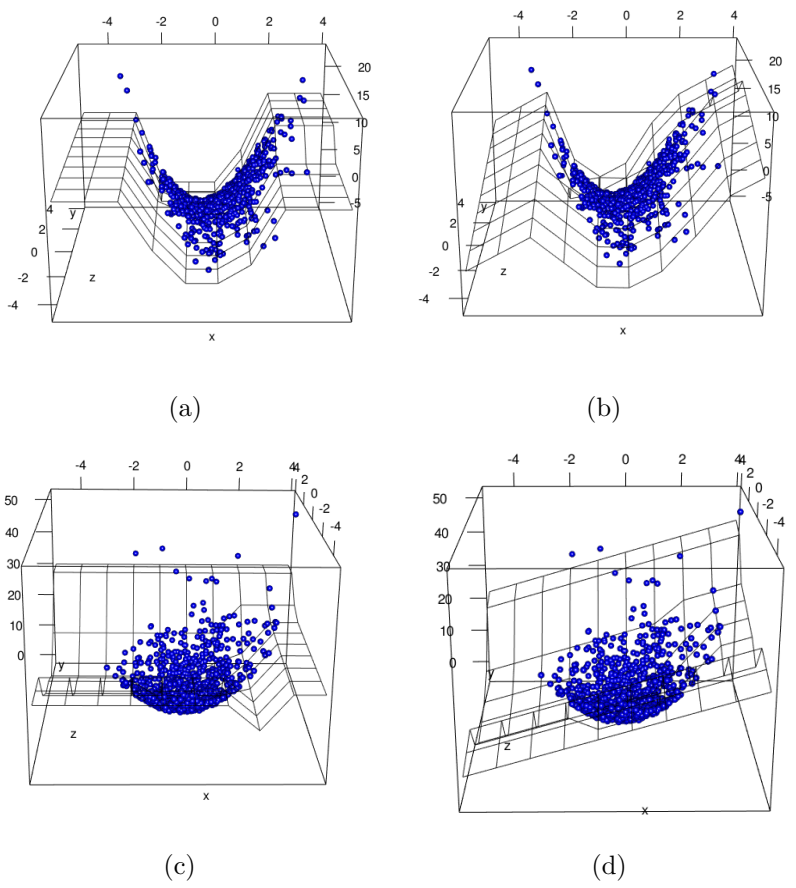


Figure 3.1: Two examples on two different simulation data of regression functions in three dimensions: in (a) and (c), piecewise-constant surfaces obtained with a regression tree; in (b) and (d), piecewise-linear surfaces obtained with the Semilinear regression tree.

remaining of this Section, I am going to introduce these models along with the estimation algorithm.

The STIMA algorithm

STIMA, is an algorithm to estimate a regression trunk model, that is a model which integrates a regression model and a regression tree. The goal of STIMA is to search for higher order interaction effects that can be added to a linear regression model with main effects of the predictors (Doove et al., 2014).

Let Y be a continuous response and X_j, with $j = 1, \ldots, p$ a set of continuous predictors. The model can be defined as

$$\mathbb{E}[Y|\boldsymbol{X}] = \beta_0 + \sum_{j=1}^{p} \beta_j X_j + \sum_{m=1}^{M-1} \beta_{p+m} I((X_1, \ldots, X_p) \in R_m).$$

where M denotes the total number of terminal nodes. Notice that one region serves as reference group. Consequently, $M - 1$ is the total number of regions included

in the model. The intercept parameter for the reference region R_M is β_0, while $\beta_0+\beta_{p+m}$ are the intercept coefficients for the other regions R_m, $m = 1, \ldots, M-1$. The slopes are denoted by β_j, and allow different predicted values for units in a same region, in contrast to regression trees where all subjects in the same region receive the same predicted value.

The learning procedure works as follows. At the first step, the model considered is the linear model with $M = 1$

$$\mathbb{E}[Y|\boldsymbol{X}] = \beta_0 + \sum_{j=1}^{p} \beta_j X_j.$$

The model parameters are estimated via OLS. In the successive step, STIMA determines the first split variable and the split point. This is done estimating for all the covariates X_j and all over each split points s^*

$$\mathbb{E}[Y|\boldsymbol{X}] = \beta_0 + \sum_{j=1}^{p} \beta_j X_j + \beta_{j+1} I(X_{j*} > s^*).$$

where the indicator function represents a binary split of the data on the basis of predictor X_{j*} at the point s^*. The chosen variable is the one that induces the highest increment in the explained variance (minimum MSE). Call X_1 the first X_{j*} chosen. The splitting procedure is repeated within the binary partition defined by X_1 at s_1 with exhaustively searching among all the covariates. Again, the reference model is updated by choosing the model that gives the highest increase in the explained variance. The splitting continues until no further improvement in the explained variance is possible. To avoid overfitting, at each step, a procedure of V-fold cross validation is implemented to obtain the predicted values. Once a large model has been constructed, the tree is pruned using a V-fold cross-validation. Specifically, choosing V, the number of subsets for the cross-validation (CV) and a parameter $\varrho = [0, 1]$ used in ϱ-SE rule to select final regression trunk used, the size of the best pruned tree is chosen by selecting the subtree that minimize the sum of the relative cross-validated error RE_m^{CV} and the standard error of a regression with m splits, that are

$$RE_m^{CV} = \frac{\sum_{v=1}^{V} \sum_{i \in (Y,\boldsymbol{X})^{\text{test}}} (Y_i^v - \hat{Y}_i^v)^2}{\sum_i^N (Y_i - \bar{Y})^2} \tag{3.3}$$

$$SE_m^{CV} = \frac{\sqrt{\sum_{i=1}^{N} [(Y_i - \hat{Y}_i^{cv})^2) - N^{-1} \sum_{i=1}^{N} (Y_i - \hat{Y}_i^{cv})^2)]^2}}{\sum_i^N (Y_i - \bar{Y})^2}. \tag{3.4}$$

In (3.3), the sum is over all the V cross-validated subsets and all over the i units of the test set in the corresponding cross-validated subset. Therefore Y_i^v are the observations in the $v - th$ subsets, and $\hat{Y}_i^v$ are the predicted values in this

subset. In (3.4), the $\hat{Y}_i^{cv}$ is the vector of the sums of the predicted values from all test sets $\hat{Y}_i^v$. Let $RE_{m^*}^{CV}$ be the size of the model with lowest RE_m^{CV}. The size of the pruned model m^{**} corresponds to the minimum value of m such that $RE_{m^{**}}^{CV} \leq RE_{m^*}^{CV} + \varrho SE_{m^*}^{CV}$.

In summary, STIMA creates a sequence of nested regression models with the purpose of reducing the bias by capturing the interactions among variables. The innovation of STIMA is in the simultaneous estimation of the linear part and the tree model by a linear regression model.

The STIMA algorithm is summarized in Algorithm 8.

Algorithm 8: Pseudo Algorithm for regression trunk building by STIMA algorithm

Data: $\{Y_i, \mathbf{X}_i\}$, $i = 1, \ldots, n$
Result: Regression Trunk model
Initialization: Fit a linear regression with main effect $\mathbb{E}[Y|\boldsymbol{X}] = \beta_0 + \sum_{j=1}^{p} \beta_j X_j$;
Exhaustive search among all X_j of the first split variable and split point;
First model: $\mathbb{E}[Y|\boldsymbol{X}] = \beta_0 + \sum_{j=1}^{p} \beta_j X_j + \beta_{j+1} I(X_{j*} > s^*)$;
repeat
    Search among all X_j of the split variable and split point within the binary partition;
    Select the model that gives the highest increase in the explained variance;
    Predict $\mathbb{E}[Y|\boldsymbol{X}]$ by a V-fold cross validation.
until *no further improvement in explained variance*;
Obtain the final model $\mathbb{E}[Y|\boldsymbol{X}] = \beta_0 + \sum_{j=1}^{p} \beta_j X_j + \sum_{m=1}^{M-1} \beta_{p+m} I((X_1, \ldots, X_p) \in R_m)$.

Sparse tree-embedded graphical models

Liu et al. (2014) define a graphical model able to detect both linear and non linear associations. They propose an algorithm based on a combination of a regression model and a regression tree. As my interest is in how the authors construct the algorithm for the associations detection, I will focus the attention on this part rather than in the graphical model. Moreover, in next paragraphs I am explaining the estimation procedure as described in Liu et al. (2014) and the modified version implemented in the R functions made available by the authors.

Let Y be a continuous response and X_j, with $j = 1, \ldots, p$ be the continuous predictors. The conditional expectation of the response is assumed

$$\mathbb{E}[Y|\boldsymbol{X}] = g(\boldsymbol{\beta}^T \boldsymbol{X} + \gamma T(\boldsymbol{X}; \mathcal{R}_Y, \mu)) \tag{3.5}$$

where $\boldsymbol{\beta}^T\mathbf{X}$ is a the linear part, $T(\mathbf{X}; \mathcal{R}_Y, \mu)$ is the regression tree part and γ is a parameter that measures the effect of the tree. The link function g depends on the type of $\mathbf{X}$, here assumed to be the identity.

The aim is to model separately linear and non linear effects. To discover the association between Y and $\boldsymbol{X}$ the set of parameters to estimate are the nonzero $\boldsymbol{\beta}$ and the $\mathcal{R}_Y$ and μ that identify the tree. This is done by an iterative fitting algorithm that combine the LASSO estimate of the linear part and the CART algorithm for the non linear part of the model in (3.5). The authors employ the existent package `RPART` and `glmnet` in their algorithm. The procedure starts with an estimation of the tree on Y. The β parameters and the tree T are then estimated alternately. The predicted values from the tree enter in the LASSO as an offset. Then the tree is estimated via the `RPART` routine. In the function implementation, the usual CART splitting procedure is replaced by a splitting procedure proposed by the authors (Therneau, 2018). Hence, the search of the splitting point is done along the direction of maximum reduction of the negative likelihood of a linear model of Y on the predicted values of the LASSO model as offset. The algorithm proceeds until no further change are observed on the tree structure. The estimation process is summarized in Algorithm 9.

Algorithm 9: Pseudo Algorithm to discover association in STGM

Data: $\{Y_i, \mathbf{X}_i\}$, $i = 1, \ldots, n$
Result: $\hat{\beta}^t$, $\hat{\gamma}^t$, $\hat{T}^{t-1}$

Initialization: Convergence= $false$;
t=0;
Fit a tree via `RPART` to obtain $\hat{Y}_{tree}$;
repeat
    Fit a LASSO regression $\mathbb{E}(Y|\boldsymbol{X}) = \text{offset}(\hat{Y}_{tree}) + \boldsymbol{X}\boldsymbol{\beta}$;
    Obtain $\hat{Y}_{LASSO}$;
    Fit a tree via `RPART` (authors splitting function);
    If $T^t = T^{t-1} \rightarrow$ Converge=TRUE;
until *Convergence= true*;

3.3 A new two-stage estimation procedure

The model of (3.1) could be useful to capture both linear and non linear association. Moreover, the employment of well-known estimation methods can lead, without a stronger computational effort, to performances competitive with most of the algorithms described in Chapter 1. My estimation proposal is to combine the least square estimate and the regression tree estimate in a backfitting two-stage iterative algorithm.

Let $\tau = 0, \ldots, t$ be the iteration number. The algorithm works as follows.

1. Initialize the tree at the mean of the response variable

$$\hat{T}^{(0)}(\boldsymbol{X}; \hat{\mathcal{R}}_Y, \hat{\mu}) = \bar{Y}.$$

2. Compute the centred values of Y

$$Y_{ct}^{(1)} = Y - \hat{T}^{(0)}(\boldsymbol{X}; \hat{\mathcal{R}}_Y, \hat{\mu}). \tag{3.6}$$

3. Fit the model $\hat{Y}_{ct}^{(1)} = \boldsymbol{X'\beta} + \epsilon$ by the OLS as $\boldsymbol{(X'X)^{-1}X'\hat{y}_{ct}^{(1)}}$ to obtain $\hat{Y}_{ols}^{(1)}$.

4. Compute the residuals given the linear model

$$\hat{Y}_{res}^{(1)} = Y - \hat{Y}_{ols}^{(1)}. \tag{3.7}$$

5. Estimate the tree model to obtain

$$\hat{T}^{(1)}(\boldsymbol{X}; \hat{\mathcal{R}}_{Y_{res}^{(1)}}, \hat{\mu}) = \sum_{m=1}^{M} \hat{\mu}_{\hat{R}_m} \mathbb{I}_{(\boldsymbol{x}_i \in \hat{R}_m)}.$$

6. Use the predicted values from step 5 $\hat{Y}_{tree}^{(1)} = \hat{T}^{(1)}(\boldsymbol{X}; \hat{\mathcal{R}}_{Y_{res}^{(1)}}, \hat{\mu})$ to compute the new centred values of Y, $Y_{ct}^{(2)}$.

7. Repeat from step 2 to step 6 until the end of iterations t or convergence is reached.

8. Obtain $\mathbb{E}(Y|\boldsymbol{X}) = \boldsymbol{X'\hat{\beta}}^{(t)} + \hat{T}^{(t)}(\boldsymbol{X}; \hat{\mathcal{R}}_{Y_{res}^{(t)}}, \hat{\mu})$

The tree estimate can be performed with four different algorithms (with in parentheses the R package used): CART (`rpart`), conditional inference tree (`party`), evolutionary trees (`evtree`), and random forests (`randomForest`). Some details on the setting will be given in Chapter 4. Note that in step 5 of the two-stage algorithm, if `randomForest` method is employed, the estimate changes in $\hat{T}_{rf}(\boldsymbol{X}; \hat{\mathcal{R}}_{Y_{res}}, \hat{\mu}, B)$ as from (1.4). The pseudo-algorithm is reported in Algorithm 10.

Essentially, the proposed estimation procedure is similar to that of the partial linear model in (3.2) with a backfitting approach (Härdle et al., 2012). The backfitting (Buja et al., 1989) was originally proposed as an iterative algorithm for fitting additive models. The key idea of backfitting is to regress the additive components separately on partial residuals. Here this idea is followed, computing

Algorithm 10: Pseudo Algorithm of backfitting for an iteratively two-stage estimator of the Semilinear regression tree

Data: $\{Y_i, \mathbf{X}_i\}$, $i = 1, \ldots, n$
Result: Two-stage Semilinear regression tree
1 Initialization step: Tree fit initialized to the mean of Y:$\hat{T}(\boldsymbol{X}; \hat{\mathcal{R}}_Y, \hat{\mu}) = (\bar{Y})$;
2 **for** *t=1 to niter* **do**
3 Compute centred values $Y_{ct} = Y - \hat{T}(\boldsymbol{X}; \hat{\mathcal{R}}_Y, \hat{\mu})$;
4 Fit the least square regression of Y_{ct} on $\boldsymbol{X}$ to obtain $\hat{Y}_{ols} = \boldsymbol{X'}\hat{\boldsymbol{\beta}}$;
5 Compute residuals: $Y_{res_1} = Y - \hat{Y}_{ols}$;
6 Fit the regression tree of Y_{res_1} on $\boldsymbol{X}$ and obtain $\hat{Y}_{tree} = \hat{T}(\boldsymbol{X}; \hat{\mathcal{R}}_{Y_{res_1}}, \hat{\mu})$;
7 Update $\hat{T}(\boldsymbol{X}; \hat{\mathcal{R}}_Y, \hat{\mu}) = \hat{Y}_{tree}$
8 **end**
9 Compute $\hat{Y} = \hat{Y}_{ols} + \hat{Y}_{tree}$.

iteratively the partial residuals from one of the components of the SRT model to obtain the estimates of the other. Specifically, let us rewrite the Semilinear regression tree in (3.1) in the form of the partial linear model of (3.2)

$$Y = \boldsymbol{X'}\boldsymbol{\beta} + T(\boldsymbol{X}_{Tree}; \mathcal{R}_Y, \boldsymbol{\mu}) + \epsilon \tag{3.8}$$

where $\boldsymbol{X}$ and $\boldsymbol{X}_{Tree}$ play the roles of $\boldsymbol{U}$ and $\boldsymbol{W}$ respectively, and the regression tree function $T(\cdot)$ plays the role of the nonparametric function $f(\cdot)$. The error component ϵ is assumed to have zero mean and finite variance. By taking the expectation conditioned on $\boldsymbol{X}_{Tree}$

$$\mathbb{E}(Y|\boldsymbol{X}_{Tree}) = \mathbb{E}(\boldsymbol{X'}\boldsymbol{\beta}|\boldsymbol{X}_{Tree}) + \mathbb{E}\{T(\boldsymbol{X}_{Tree})|\boldsymbol{X}_{Tree}\} + \mathbb{E}(\epsilon|\boldsymbol{X}_{Tree}) \tag{3.9}$$

and by subtracting (3.9) from (3.8) on obtain

$$Y - \mathbb{E}(Y|\boldsymbol{X}_{Tree}) = \{\boldsymbol{X} - \mathbb{E}(\boldsymbol{X}|\boldsymbol{X}_{Tree})\}'\boldsymbol{\beta} + \epsilon - \mathbb{E}(\epsilon|\boldsymbol{X}_{Tree})$$

since $\mathbb{E}\{T(\boldsymbol{X}_{Tree})|\boldsymbol{X}_{Tree}\} = T(\boldsymbol{X}_{Tree})$. By definition, $\mathbb{E}(\epsilon|\boldsymbol{X}, \boldsymbol{X}_{Tree}) = 0$, and it can be shown that $\mathbb{E}(\epsilon - \mathbb{E}(\epsilon|\boldsymbol{X}_{Tree})) = 0$ holds by applying the law of iterated expectations.

Therefore, once one of the two components of the SRT model is obtained, that is $\boldsymbol{\beta}$ or T, the computation of the other can be carried out by the backfitting approach. This is accomplished by subtracting $\boldsymbol{X}'\boldsymbol{\beta}$ from Y to obtain

$$\mathbb{E}(Y - \boldsymbol{X}'\boldsymbol{\beta}|\boldsymbol{X}_{Tree}) = T(\boldsymbol{X}_{Tree}). \tag{3.10}$$

Hence the $\boldsymbol{\beta}$ parameters can be estimated by OLS. This occurs in step 3 of the

proposed algorithm. Plugging $\boldsymbol{X}'\hat{\boldsymbol{\beta}}$ in (3.10) yields to the regression tree estimated model $\hat{T}(\boldsymbol{X}_{Tree})$. Then, as

$$\mathbb{E}(Y - \hat{T}(\boldsymbol{X}_{Tree}|\boldsymbol{X})) = \boldsymbol{X}'\boldsymbol{\beta}$$

the $\boldsymbol{\beta}$ are updated given $\hat{T}(\boldsymbol{X}_{Tree})$ and so on. These are the steps 4 and 5 of the proposed algorithm.

Therefore, the SRT model

$$\mathbb{E}(Y|\boldsymbol{X}, \boldsymbol{W}) = \boldsymbol{X'\beta} + T(\boldsymbol{X}_{Tree})$$

consists of only two additive components. Denoting by $\boldsymbol{P}$ the projection matrix $P = \boldsymbol{X}(\boldsymbol{X'X})^{-1}\boldsymbol{X'}$ and a $\boldsymbol{S}$ a smoother matrix, backfitting mean to solve

$$\boldsymbol{X\beta} = \boldsymbol{P}(Y - T)$$

$$T = \boldsymbol{S}(Y - \boldsymbol{X\beta})$$

As a consequence, the estimation procedure proposed is a backfitting algorithm for an iteratively two-stage estimator. Because of its iterative nature, theoretical results for backfitting have not been well explored and for my proposal are under investigation. However, Opsomer and Ruppert (1999) provides the asymptotic properties of semiparametric additive models both when the nonparametric component is univariate ad multivariate. In case of univariate nonparametric model, they give the asymptotic approximation of the conditional bias and variance of $\hat{\beta}$ in case of local linear regression as smoother. In addition, Mammen et al. (1999) proved the consistency and calculated the asymptotic properties under weaker conditions when using kernel regression smoothers for the estimation of the nonparametric component. The properties of my proposal are under study noticing that the regression trees can be seen as a particular method for constructing regressograms (Tukey, 1947), that is a particular linear smoother, see Wasserman (2006).

3.4 A new evolutionary estimation procedure

The evolutionary estimation procedure is proposed with the aim to treat simultaneously the linear and the tree part of the model in (3.1). Moreover, this innovative estimation method merges several notions described in Chapter 1, and tries to overcome the pitfall presented in Chapter 2.

Recall that (see Section 1.2) a regression tree can be written as a single factor regression model. Hence, the proposed model is viewed as a general linear model

with continuous predictors and a factor with M-levels, that is an ANCOVA in case of Gaussianity assumption. The model (3.1) in case of p continuous variables and one single factor with 2-levels (one split tree) can be written as

$$\mathbb{E}(Y|\boldsymbol{X}) = \boldsymbol{\beta}_1 X_1 + \cdots + \boldsymbol{\beta}_p X_p + \mu_1 \mathbb{I}_{(X_j \geq s_1)} + \mu_2 \mathbb{I}_{(X_j < s_1)} \tag{3.11}$$

or, in matrix form

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ \vdots \\ Y_i \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} X_{11} & \dots & X_{1j'} & \dots & X_{1p} & 1 & 0 \\ X_{21} & \dots & X_{2j'} & \dots & X_{2p} & 1 & 0 \\ X_{31} & \dots & X_{3j'} & \dots & X_{3p} & 1 & 0 \\ \vdots & \dots & \vdots & \dots & \vdots & 1 & 0 \\ X_{i1} & \dots & X_{ij'} & \dots & X_{ip} & 0 & 1 \\ \vdots & \dots & \vdots & \dots & \vdots & 0 & 1 \\ X_{np} & \dots & X_{nj'} & \dots & X_{np} & 0 & 1 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \dots \\ \beta_p \\ \mu_1 \\ \mu_2 \end{bmatrix}$$

In (3.11) X_j represents the splitting variable and s_1 represents the splitting point. The two indicator functions represent the partitions $\mathcal{X}$ of the tree. An additional split corresponds to the inclusion multiplicatively of a further indicator variable, such as

$$\mathbb{E}(Y|\boldsymbol{X}) = \boldsymbol{\beta}_1 X_1 + \cdots + \boldsymbol{\beta}_p X_p + \mu_1 \mathbb{I}_{(X_j \geq s_1)} + \mu_2 \mathbb{I}_{(X_j < s_1)} \cdot \mathbb{I}_{(X_j \geq s_2)} + \mu_3 \mathbb{I}_{(X_j < s_1)} \cdot \mathbb{I}_{(X_j < s_2)}.$$

This model specification makes it possible to estimate the model parameters $\boldsymbol{\beta}$ and $\boldsymbol{\mu}$ via the ordinary least square estimator $\boldsymbol{(X'X)^{-1}X'y}$.

The algorithm proceeds as follows. First, the model is initialized to a linear regression model which includes all the X_p covariates and an M-level factor randomly chosen. The parameters are estimated with OLS. After that, the current model is perturbed in the tree part to obtain the new estimated model parameters, again, with OLS. At each step, a comparison between the old and the new models is performed, and the one with minimum MSE is retained. The evolution continues until the last iteration or convergence is reached. At the end of the evolution, the final model is returned and it is used to estimate the $E(Y|\boldsymbol{X} = \boldsymbol{x})$. As a further extension, I implemented an *honest* version of the algorithm as suggested in Wager and Athey (2018). Here, at each iteration, the tree is grown using two non-overlapping subsamples of the training data, called $(\boldsymbol{X}_i, Y_i)^{\texttt{train}_1}$ and $(\boldsymbol{X}_i, Y_i)^{\texttt{train}_2}$. The tree structure is chosen only using the data in $(\boldsymbol{X}_i, Y_i)^{\texttt{train}_1}$, while the model is estimated only using the data in $(\boldsymbol{X}_i, Y_i)^{\texttt{train}_2}$. Wager and Athey (2018) note that for random forests predictors this kind of double subsampling can improve the MSE of prediction. In the next chapter, I will compare the predictive performance

between an honest and a non-honest version of the proposed algorithm.

Algorithm 11: Pseudo Algorithm for Evolutionary Semilinear regression tree building

Data: $\{Y_i, \mathbf{X}_i\}$, $i = 1, \ldots, n$
Result: Evolutionary Semilinear regression tree
Initialization step: Model specification as in (3.1);
Estimation of the model with OLS: SAVE→ Current model;
for *t=1 to niter* **do**
    Perturbation step: GROW, PRUNE or CHANGE the tree;
    Evaluation step: estimates of the model perturbed: SAVE → New model;
    Comparison between current model and new model and choice of the best model;
    SAVE: Best model → Current model
end
Compute $\hat{Y} = E(Y|\boldsymbol{X} = \boldsymbol{x})$.

The entire process is summarized in Algorithm 11. In the next section, I will give more details on the three phases that identify the process.

Initialization step

The starting model (3.1) is defined by the sum of a linear model and a tree model. The tree forms an M-level factor, and the procedure is composed by the following steps, where the superscript will denote the sampled splitting variable. For example, X_j^1 will be the first splitting variable, X_j^2 will be the second splitting variable and so on. The splitting variable may be the same all time, for example X_1 at each sample procedure, or may differ.

STEP 1: SPLITTING

- Sample from $\boldsymbol{X}$ the first splitting variable X_j^1 with probability $\frac{1}{p}$
- Discard from the empirical distribution of X_j^1 10% of observations from the two tails, 5% from the right tail and 5% from the left tail. Call X_j^{1*} the resulting vector of values of dimension n^*
- Sample from X_j^{1*} the splitting point s_1 with probability $\frac{1}{n^*}$
- Create two dummies $\mathbb{I}_{(X_j^1 \geq s_1)}$ and $\mathbb{I}_{(X_j^1 < s_1)}$

STEP 2 Repeat STEP 1 within the partitions $\mathbb{I}_{(X_j^1 \geq s_1)} = 1$ and $\mathbb{I}_{(X_j^1 < s_1)} = 1$

STEP 3 Create the final dummies that identify the tree regions, such as,
if $M = 4$,

$$\boldsymbol{R} = \left(\mathbb{I}_{(X_j^1 \geq s_1)} \cdot \mathbb{I}_{(X_j^2 \geq s_2)}, \quad \mathbb{I}_{(X_j^1 \geq s_1)} \cdot \mathbb{I}_{(X_j^2 < s_2)}, \quad \mathbb{I}_{(X_j^1 < s_1)} \cdot \mathbb{I}_{(X_j^3 \geq s_3)}, \quad \mathbb{I}_{(X_j^1 < s_1)} \cdot \mathbb{I}_{(X_j^3 < s_3)} \right)$$

STEP 4 Obtain the starting model: $Y = \boldsymbol{\beta X} + R + \epsilon$.

The first model that will be evaluated is a linear regression model with all the $\boldsymbol{X}_p$ covariates and an M-level factor. An example of the tree part of the model is illustrated in Figure 3.3, left side. The idea is to keep the depth of the tree small whenever the functional form of the dependence is supposed to be quasi-linear. A short tree can help to avoid overfitting and to improve efficiency. The exploration of a space of the trees is achieved by a perturbation step.

In order to choose the split point, a modification of the CART splitting procedure is proposed. The first modification is the random selection of the splitting variable. Moreover, since the choice of the split point among all the observations of the empirical distribution is computationally intensive, the proposed algorithm implements the strategy used in *Extra-trees* (Section 1.5) and in *purely* random forests (Breiman, 2000), by introducing the random choice of the split point selection. Note that for random forests, it has been shown that the random choice of the split point can improve the performance of the MSE in presence of noise variable (Geurts et al., 2006). Moreover, Biau et al. (2008) prove the consistency of *purely* random forests for the case of independent explanatory variables.

The choice of leaving out 10% of the observations from the tails avoids the case of a small number of observations in the leaves if the split point is on the boundaries of the empirical distribution.

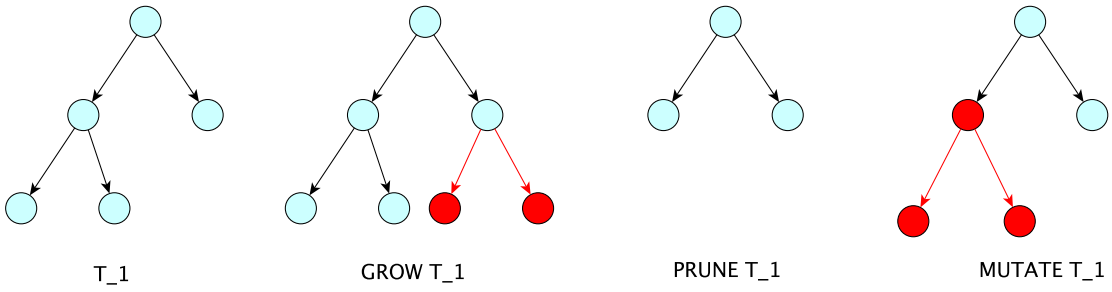
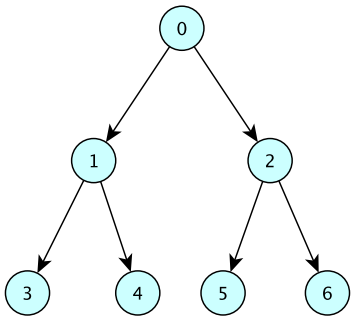


Figure 3.2: Moves of the perturbation step: GROW consists in the random selection of a node and its split. PRUNE consists in the random selection of a node and its deletion from the tree. MUTATE consists in the random selection of a node and its change in terms of splitting variable and splitting rule.



Tree depth	Possible moves
0	GROW
1	GROW, CHANGE
2, ..., maxdepth−1	GROW, PRUNE, CHANGE
maxdepth	PRUNE, CHANGE

Figure 3.3: On the left, an example of the tree structure of the SRT model in (3.4). The tree nodes are labelled from 1 to 6, and $d = 0, 1, 2$ represents the depth of the tree at each node. On the right, the possible moves of the perturbation step depending from the depth of the tree.

Perturbation step

The idea of a perturbation step of the model arises from the perturbation step of the *BART* model (Section 1.6) and from the modification by the variation operators of the *Evtree* algorithm (Section 1.3). Starting from the current model, the tree structure is modified in an evolutionary way, while the linear part of the model is not involved in the modification. Three possible moves are allowed to modify the tree structure: *grow*, *prune*, and *mutate*. They are illustrated in Figure 3.2. At each iteration of the algorithm, a valid move is randomly selected from a Uniform distribution. The valid rules of perturbation depend on the position of the node involved. Specifically, given the depth of the tree, the possible moves are summarized in Figure 3.3, right side. For example, from the maximum depth of the tree, *grow* move would not be possible. In next paragraphs details on the three perturbation steps will be given.

Grow The GROW move is possible at each depth of the tree, except for the maximal depth of the tree. The move consists in the random selection of a node and its split into two leaves. The split provide two indicator functions to be multiplied for the parent node. Taking as an example the tree T_1 of Figure 3.2, the dummy vector representing T_1 is

$$\boldsymbol{R}_{T_1} = \left(\mathbb{I}_{(X_j^1 \geq s_1)} \cdot \mathbb{I}_{(X_j^2 \geq s_2)}, \quad \mathbb{I}_{(X_j^1 \geq s_1)} \cdot \mathbb{I}_{(X_j^2 < s_2)}, \quad \mathbb{I}_{(X_j^1 < s_1)} \right).$$

The second tree of Figure 3.2 shows the tree T_1 after the growing. For example, by selecting $X_j^1 < s_1$ in $\boldsymbol{R}_{T_1}$ as node to grow, $\boldsymbol{R}$ becomes

$$\boldsymbol{R}_{T_1} = \left(\mathbb{I}_{(X_j^1 \geq s_1)} \cdot \mathbb{I}_{(X_j^2 \geq s_2)}, \quad \mathbb{I}_{(X_j^1 \geq s_1)} \cdot \mathbb{I}_{(X_j^2 < s_2)}, \quad \mathbb{I}_{(X_j^1 < s_1)} \cdot \mathbb{I}_{(X_j^3 \geq s_3)}, \quad \mathbb{I}_{(X_j^1 < s_1)} \cdot \mathbb{I}_{(X_j^3 < s_3)} \right).$$

Prune PRUNE consists in deleting a node of the tree randomly selected. This is not possible at depth 0, since the root node cannot be pruned. The third tree of Figure 3.2 shows the tree T_1 after the pruning. For example, by selecting $X_j^2 \geq s_2$ or $X_j^2 < s_2$ in $\boldsymbol{R}_{T_1}$ as node to prune $\boldsymbol{R}$ becomes

$$\boldsymbol{R}_{T_1} = \left(\begin{array}{cc} \mathbb{I}_{(X_j^1 \geq s_1)}, & \mathbb{I}_{(X_j^1 < s_1)} \end{array} \right).$$

Mutate MUTATE is the hardest move in terms of computational intensity, because in some situation it involves the entire redefinition of the tree. It consists in the random selection of a node and its change in terms of splitting variable and splitting point. In the proposed algorithm MUTATE is possible for all the nodes of the tree. The fourth tree of Figure 3.2 shows the tree T_1 after the mutation. For example, by selecting $X_j^1 \geq s_1$ in $\boldsymbol{R}_{T_1}$ as node to mutate $\boldsymbol{R}$ becomes

$$\boldsymbol{R}_{T_1} = \left(\begin{array}{ccc} \mathbb{I}_{(X_j^* \geq s_*)} \cdot \mathbb{I}_{(X_j^2 \geq s_2)}, & \mathbb{I}_{(X_j^* \geq s_*)} \cdot \mathbb{I}_{(X_j^2 < s_2)}, & \mathbb{I}_{(X_j^* < s_*)} \end{array} \right)$$

where $X_j^* \geq s_*$ denotes the new variable and the new split point sampled in the mutation step.

Evaluation step

The evaluation step involves the model parameters estimation and the computation of a measure on which two models will be compared. As already mentioned, the specification of the model allow the use of the OLS estimator for both $\boldsymbol{\beta}$ and $\boldsymbol{\mu}$ parameters. Once the model parameters are estimated, its goodness of fit is usually computed by the MSE. The proposed algorithm utilizes the measure proposed by Grubinger et al. (2011) and also adopted in Fan and Gray (2005), that is a MSE with a BIC-type complexity penalty. This choice is made since in traditional regression model building, the use of residual sum of square or R^2 for the tree fitness criterion results in large models and may lead to overfitting. Therefore the MSE-BIC here adopted is

$$\begin{aligned} \text{MSE}_{\text{SRT}} &= n \log(\text{MSE}) + \text{comp}_{\text{SRT}} \\ \text{comp}_{\text{SRT}} &= \lambda \cdot 4 \cdot (G+1) \cdot \log n \end{aligned} \tag{3.12}$$

where $G+1 = M+p+1$ is the number of estimated parameters, the $\boldsymbol{\mu}$ parameters for each of the terminal nodes, the $\boldsymbol{\beta}$ parameters for the linear part and the residual variance term. This measure has been also adopted in *evtree* (1.6).

After the perturbation step, the current model is compared with his successor. The best model is the one with minimum MSE_{SRT}. It is kept as current model and employed in the subsequent iteration.

3.5 Some remarks

In the previous sections, I proposed two algorithm for the estimation of the SRT model. As already pointed out in Section 3.2, such model is essentially a semi-parametric model, precisely a partial linear model, where the nonparametric part is a regression tree. The SRT model is also is linked to an ANCOVA model in the sense that it is equivalent to an ANCOVA with parallel line parametrization, which implies a common slope and a specific intercept for each level of the factor.

The two-stage estimation procedure is to some extent related to the algorithm proposed by Liu et al. (2014) (see Section 3.2). Both these algorithms estimate separately the linear and the tree components of the model and rely on the CART algorithm for the non linear part. The main difference is in using the residuals during the iterations. This makes possible to extract the information about the relationship between the response and the covariates by subtracting each time the variability explained by both components. In (3.6) the variability of Y captured by the tree is subtracted to the total variability of Y. Then, in (3.7), the variability captured by the linear regression is subtracted to the total variability of Y. Therefore, my proposed algorithm is in fact a proper backfitting algorithm. The coefficients associated to the linear terms are the partial effects of a predictor on the response, while the tree parameters will handle the interaction and non linear partial effects.

It is worth to notice that the functional form of the regression function is assumed to be quasi-linear. In the presence of strong non linearities that affect the data a quasi-linear specification could be not adequate. A possible development of the SRT model is to replace the linear component with the regression splines. In this way, the linear and non linear relationships would be handled by the splines, and the tree component would handle the interactions.

The splitting procedure here adopted is the CART splitting rule, with the recommendation to keep small the maximum depth of the tree. However, the tree algorithm can be constructed using other tree-based algorithms. For instance, one can choose among random forests, evolutionary tree or conditional inference trees. These alternative formulations will be used in the next chapter, and their differences in performance will be outlined.

The second algorithm here proposed, the evolutionary estimation procedure, is based on a different construction of the SRT model. Note that the regression trunk model in STIMA (Dusseldorp et al., 2010) is similar to my proposal. The difference between these two models is in the specification of the intercept term, which is omitted in SRT because it is included through the tree component. Looking at the estimation process, the splitting procedure in STIMA is performed

by a greedy exhaustive search among all possible variables and splitting points, while in my evolutionary procedure I introduce the randomizing component in the search. This implies a significant gain in the computational time. Moreover, in this estimation procedure a higher number of tree configurations is explored thanks to the perturbation step. This choice reinforces the suggestion to keep small the depth of the tree to avoid overfitting. Finally, in STIMA a procedure of V-fold cross validation is implemented to estimate the MSE of the test set, while I introduce the honest subsampling, which, again, is reflected by a considerable gain in computational time.

The key point of the evolutionary estimation procedure proposed is the use of the OLS estimator. This allows to avoid the greedy search algorithm and instead obtain the parameter estimates of the tree and the linear components of the model simultaneously. The parameter estimates can therefore be interpreted exactly as those of a regression model with particular interaction terms. Specifically, the interaction terms concern the product of dichotomized versions of the main effects. An example of the parameters estimation via OLS of the evolutionary algorithm for SRT model on simulated data is reported in Table 3.1. This model is without the intercept term, 4 predictors and a two-level factor. The factor represents the tree component of the model, that in this case it is a tree with one split and two terminal nodes. The coefficients associated to the linear terms represent the partial effects of a predictor on the response, while the coefficients associated to the factor represent the averages of the units belonging to that partition.

Table 3.1: An example of OLS estimates of the evolutionary algorithm for the Semilinear regression tree model on simulated data via Scenario 5 presented in Chapter 4, Section 4.2.

	Coefficient Estimate	s.e.	t-test	p-value
X1	0.2063	1.1583	0.18	0.8587
X2	-0.4082	1.1678	-0.35	0.7268
X3	-17.0082	1.1916	-14.27	0.0000
X4	11.2645	1.9819	5.68	0.0000
X5	4.3135	1.1690	3.69	0.0002
X4 $\geq$ 0.058	21.2648	2.4078	8.83	0.0000
X4 $<$ 0.058	25.4495	2.0572	12.37	0.0000

However, by avoiding the greedy search in the algorithm construction, it is generally possible to avoid the problems highlighted in Chapter 2 of methods which gives the same or major importance to indirect effects than direct effects. Looking at the tree part, the proposed tree specification could be an innovative way to construct trees that move away from the complete nonparametric form to a semiparametric model. As mentioned in Section 1.2, if the assumption of

Normality of the error term ϵ is added, as in the BART algorithm, it is possible to perform statistical hypothesis testing and to construct reliable confidence intervals for parameter estimates. In my opinion, a powerful improvement can be the employ of a statistical test for pruning the tree. In fact, a pruned tree is obtained by constraining some parameters of the tree to be equal. The pruned model is therefore nested in the unpruned one. Moreover, the normality assumption of the error term entail the ML estimator for the parameters. In fact, note that in principle the number of parameters in SRT model can increase with n, if the tree is left free to grow. Therefore, it is relevant to fix the tree depth in advance in order to achieve the ML estimator properties when the evolutionary algorithm reach the convergence. Finally, the evolutionary estimate can easily be extended to a regularized estimate of parameters as LASSO or adaptive LASSO (Zou, 2006) in high dimensional frameworks.

Chapter 4

Comparison of the proposed methods

4.1 Introduction

In Chapter 3, I studied the Semilinear regression tree model (SRT), an interpretable tree-based regression model which can balance the trade-off between the predictive and the generative modelling. Moreover, I proposed two estimation procedures: a two-stage estimation procedure based on a backfitting algorithm and an estimation procedure based on an evolutionary algorithm. In this chapter, two comparisons of the proposed estimation methods via Monte Carlo study and real data study are presented.

The chapter is organized as follows. In Section 4.2 are illustrated a comparison on the proposed algorithms, and a study on the prediction and the estimation accuracy of the proposed methods and some of the estimation methods presented in Chapter 1. In Section 4.3 are described two applications on *Boston housing* and *Carseats* data available in R software.

4.2 Monte Carlo study

4.2.1 Comparison of the proposed methods

The two-stage estimator proposed in Section 3.3 and the evolutionary estimator proposed in Section 3.4 for the SRT model are compared by considering the following Scenario.

Scenario 1: independent linear regressors and two-level factor
Let $X_j \sim \mathcal{N}(0,1)$, for $j = 1, \ldots, 4$ and let

$$Y = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \mu_1 \mathbb{I}_{(X_3 \geq s_3)} + \mu_2 \mathbb{I}_{(X_3 < s_3)} + \epsilon,$$

with $\epsilon \sim \mathcal{N}(0,1)$. The coefficients are set to $\beta_1 = 3$, $\beta_2 = 2$, $\beta_3 = 0$, $\beta_4 = 0$,$\mu_1 = 3$, and $\mu_2 = 4$. The splitting point for X_3 is $s_3 = 0.5$. In Figure 4.1 the pairwise associations and correlations among variables on simulated data are represented. I will call this scenario *LR1*.

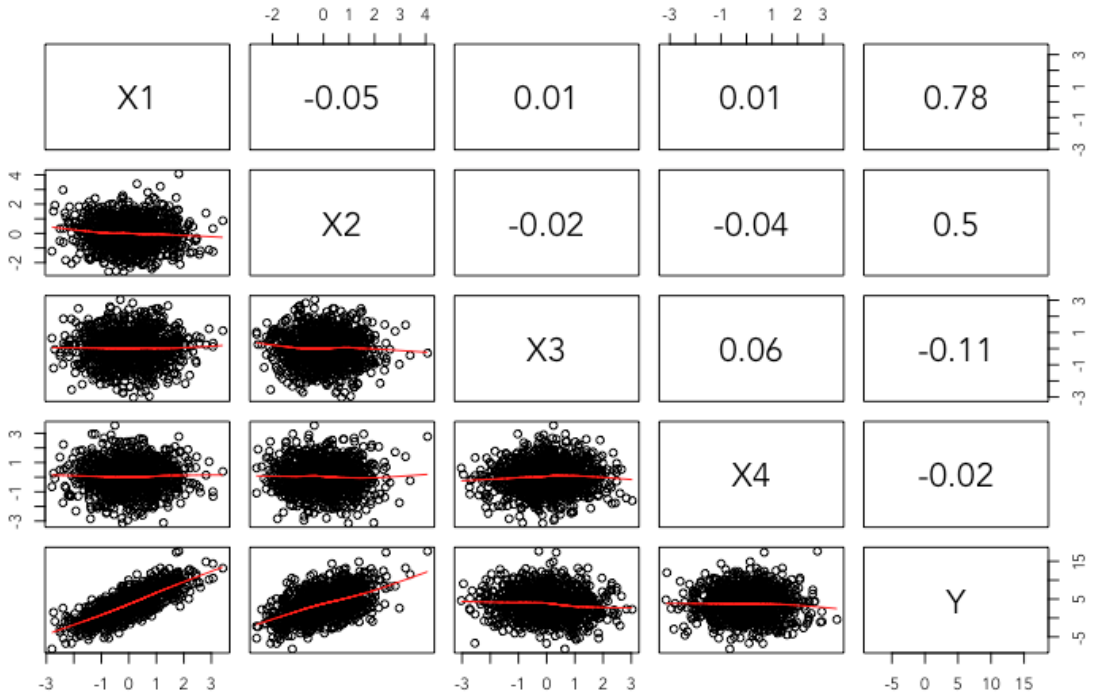


Figure 4.1: Scenario 1: in the upper part, sample pairwise correlations. In the lower part, plot of pairwise associations on simulated data according to LR1.

To compare the performances of the two estimation algorithms, I conducted a minimal simulation study with $N = 1000$ data sets with size $n_{\texttt{train}} = 1000$ and $n_{\texttt{test}} = 1000$ generated from scenario LR1.

The model has been fitted with all the $p = 4$ explanatory variables, both for the two-stage estimator and the evolutionary estimator. The settings that I employed for the two-stage algorithm are growth of the tree choosing among CART trough the R package `rpart` (*TS rpart*), conditional inference trees trough the R package `party` (*TS ctree*), and random forests trough the R package `randomForest` (*TS rf*). The comparison with evtree (`evtree`) is not reported here due to the infeasible computational time of the evtree function. I am reporting the results with maximum depth set to 2 for trees, while for random forests with maximum number of trees set to 10. For the evolutionary algorithm proposed (*EV*, or *HEV* for the honest version), the settings are $N_{iter} = 1000$ and estimation based on OLS.

Table 4.1 shows the proportions of splitting for each variable, the averages and

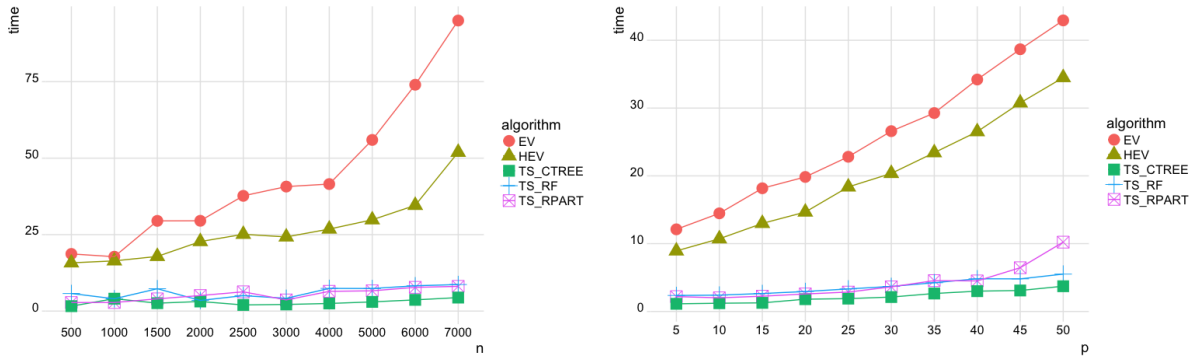
Table 4.1: Summary of proportions of splitting for each variable, Monte Carlo averages and standard errors of the splitting points and variable importance, and Monte Carlo averages and standard errors of the $\hat{\beta}$ parameters with data generation *LR1*.

Algorithms	X_1	X_2	X_3	X_4	*No split*
EV					
Depth=1					
Proportion of splits	0.007	0.007	0.978	0.008	0.000
MC average splitting point			0.523		
MC s.e. splitting point			0.007		
Depth=2					
Proportion of splits	0.000	0.000	0.178	0.000	0.822
MC average splitting point			0.499		
MC s.e. splitting point			0.003		
$\hat{\beta}$	2.999	1.999	-0.006	0.000	
$s.e.(\hat{\beta})$	0.001	0.001	0.002	0.001	
HEV					
Depth=1					
Proportion of splits	0.003	0.004	0.984	0.009	
MC average splitting point			0.511		
MC S.e. splitting point			0.006		
Depth=2					
Proportion of splits	0.000	0.000	0.090	0.000	0.910
MC average splitting point			0.494		
MC s.e. splitting point			0.005		
$\hat{\beta}$	2.999	2.000	-0.005	-0.001	
$s.e.(\hat{\beta})$	0.001	0.001	0.003	0.001	
TS rpart					
Proportion of splits	0.000	0.000	1.000	0.000	0.000
MC average splitting point			0.500		
MC s.e. splitting point			0.001		
$\hat{\beta}$	2.998	2.000	-0.089	-0.001	
$s.e.(\hat{\beta})$	0.001	0.001	0.003	0.001	
TS ctree					
Proportion of splits	0.000	0.000	0.000	0.000	1.000
$\hat{\beta}$	2.999	1.999	-0.352	-0.001	
$s.e.(\hat{\beta})$	0.001	0.001	0.001	0.000	
TS rf					
Variable importance	7480.084	3957.764	1218.313	1077.464	
MC s.e. Variable importance	12.900	8.194	2.564	1.556	
$\hat{\beta}$	2.99	1.998	-0.273	-0.001	
$s.e.(\hat{\beta})$	0.001	0.001	0.002	0.001	

standard errors of the splitting points (in case of *TS rf*, the averages and standard errors of the variable importance), and the averages and standard errors of the $\hat{\beta}$ parameters. While *EV* and *HEV* algorithms raise similar performances, the two-stage algorithm shows slightly different behaviour. Despite the greedy search employed in the tree construction, *TS rpart* has performances comparable with *EV* and *HEV*. Differently, *TS ctree* estimates trees with no split. This is due

to the *ctree* algorithm itself, that in case of continuous predictors and response performs a test on correlations, as noted in Section 1.3. Therefore, combine a linear component with the *ctree* estimation process for the tree is unhelpful, because of the tree search is done on the residual variability of Y, after removing from the total variability of Y the variability captured by the linear regression. With *TS rf*, the comparison can be made looking at the variable importance. In this case, the major variable importance is on X_1, even if the true splitting variable is X_3. Therefore, the differences highlighted among the different estimation procedures proposed are due to the different nature of the tree construction: as explained in Chapter 3, while the two-stage perform a greedy search, the evolutionary algorithm search for the best split in a non-greedy search, exploring a largest number of tree configurations.

The time consumed for the model of Scenario 1 by the proposed algorithms are showed in Figure 4.2. In the right panel, the time (in seconds) is function of the number of parameters included in the model, with $p = 4$ parameters of interest and the remaining noise parameters. In the left panel, the time (in seconds) is function of the number of observations n. *EV* and *HEV* algorithms show a higher computational burden, since their functions are completely implemented in *R* software.



(a) (b)

Figure 4.2: Computational time of the proposed algorithms for the data generation of Scenario 1. On the right, the time (in second) versus the number of parameters included in the model, with $p = 4$ parameters of interest and the remaining noise parameters. On the left, the time (in second) versus the number of observations n.

4.2.2 Prediction and estimation accuracy

In this Section, the proposed algorithms are compared with respect to some algorithms described in Chapter 1. In these Monte Carlo studies, the Scenario 1 of Section 4.2.1 will be considered, along with the following Scenario.

Scenario 2: independent linear regressors and four-level factor

Let $X_j \sim \mathcal{N}(0,1)$, for $j = 1, \ldots, 4$ and let

$$Y = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \mu_1 \mathbb{I}_{(X_2 \geq s_2)} \cdot \mathbb{I}_{(X_1 \geq s_1)} + \mu_2 \mathbb{I}_{(X_2 \geq s_2)} \cdot \mathbb{I}_{(X_1 < s_1)} + \\ + \mu_3 \mathbb{I}_{(X_2 < s_2)} \cdot \mathbb{I}_{(X_1 \geq s_{11})} + \mu_4 \mathbb{I}_{(X_2 < s_2)} \cdot \mathbb{I}_{(X_1 < s_{11})} + \epsilon,$$

with $\epsilon \sim \mathcal{N}(0,1)$. Note that at depth 2 the splitting variable is X_1 in both branches of the tree. In order to distinguish the two split points, they have been labelled s_1 and s_{11}. The parameter setting is $\beta_1 = 3$, $\beta_2 = 2$, $\beta_3 = 0$, $\beta_4 = 0$, $\mu_1 = 3$, $\mu_2 = 4$, and $\mu_3 = -1$. The splitting points are $s_2 = 1$, $s_1 = 0$ and $s_{11} = -1$. In Figure 4.4 the pairwise associations and correlations among variables on simulated data are represented. I will call this scenario *LR2*.

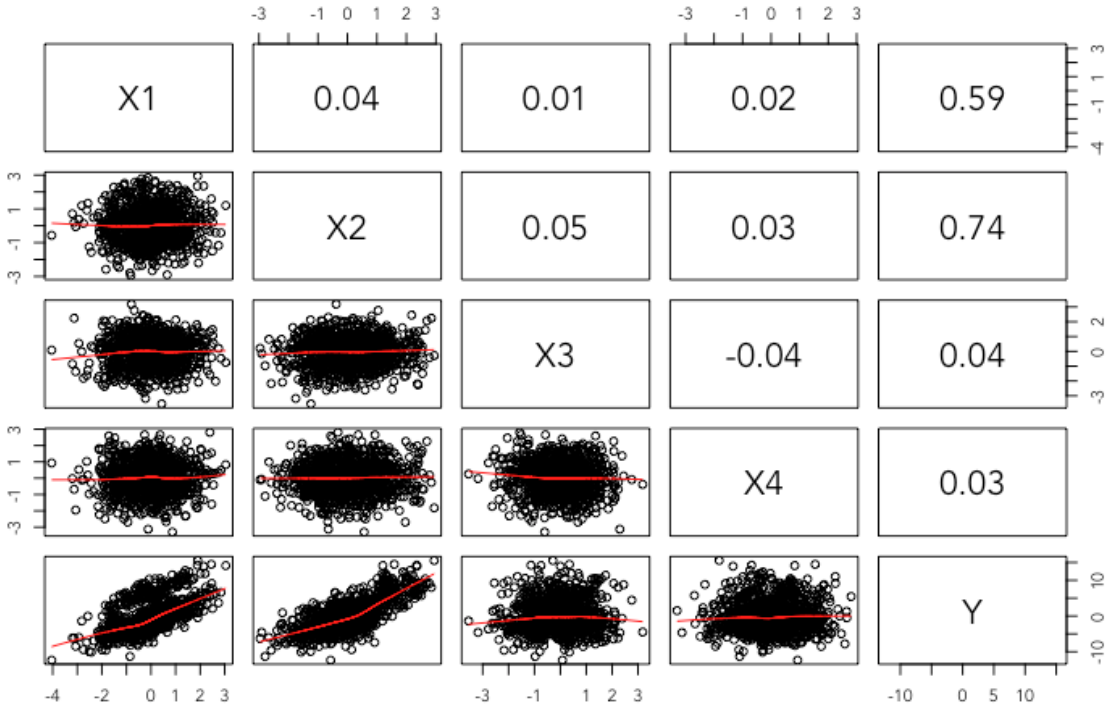


Figure 4.3: Scenario 2: in the upper part, sample pairwise correlations. In the lower part, plot of pairwise associations on simulated data according to LR2.

Scenario 3: diamond graph, linear associations

Let $X_1 \sim \mathcal{N}(0,1)$, and $X_j = aX_1 + \epsilon$ for $j = 2, \ldots, 4$, and $\epsilon \sim \mathcal{N}(0,1)$. Let

$$Y = bX_2 + bX_3 + bX_4 + \epsilon,$$

with $\epsilon \sim \mathcal{N}(0,1)$. This data generation corresponds to the one described in Section 2.3, whose graph is depicted in Figure 2.1. The parameters a and b are set to 3. In Figure 4.4 the pairwise associations and correlations among variables on simulated data are represented. I will call this scenario *DL*.

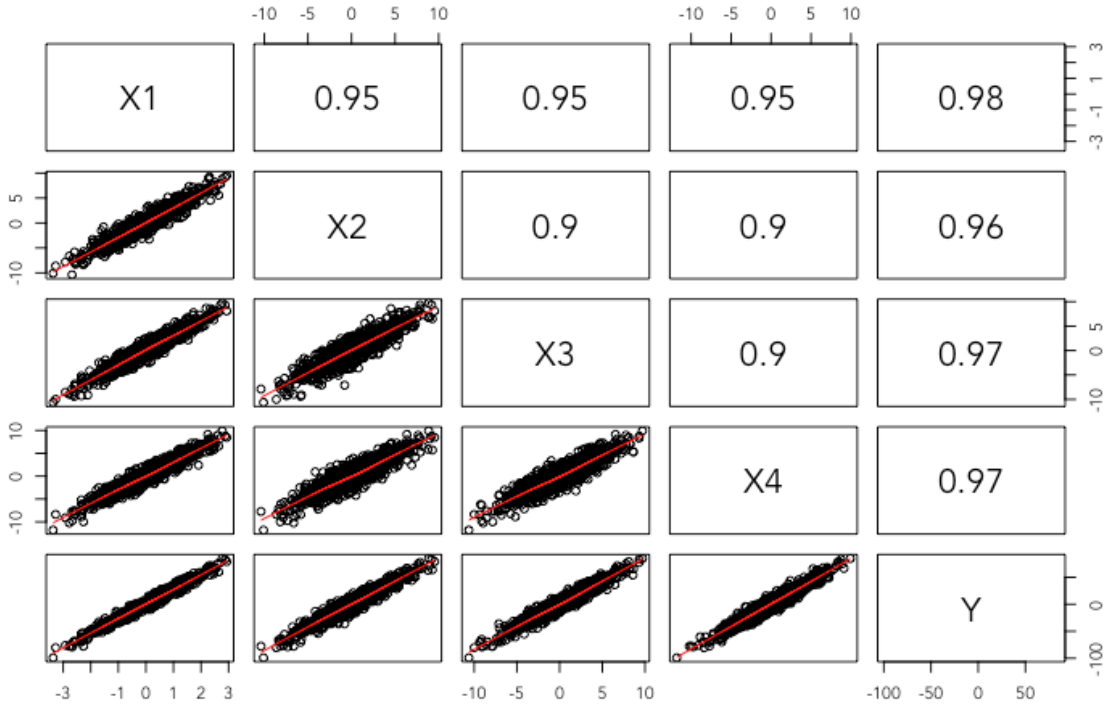


Figure 4.4: Scenario 3: in the upper part, sample pairwise correlations. In the lower part, plot of pairwise associations on simulated data according to DL.

Scenario 4: diamond graph, non linear associations

Let $X_1 \sim \mathcal{N}(0,1)$, and $X_j = aX_1 + \epsilon$ for $j = 2, \ldots, 4$, and $\epsilon \sim \mathcal{N}(0,1)$. Let

$$Y = bX_2^2 + bX_3^2 + bX_4 - bX_2X_3 + \epsilon,$$

with $\epsilon \sim \mathcal{N}(0,1)$. Again, this data generation corresponds to the one described in Section 2.3, but the associations between the response and their direct influencing variables X_2, X_3, X_4 are both linear and non linear. Also in this case, the parameters a and b are set to 3. In Figure 4.5 the pairwise associations and correlations among variables on simulated data are represented. I will call this scenario *DNL*.

Scenario 5: Friedman

Let $X_j \sim \mathcal{U}(0,1)$ for $j = 1, \ldots, 5$. Let

$$Y = 10\sin(\pi X_1 X_2) + 20(X_3 - 0.5)^2 + 10X_4 + 5X_5 + \epsilon,$$

with $\epsilon \sim \mathcal{N}(0,1)$. The relevant predictors are associated with the response in both linear and non linear way, with an interaction term. This benchmark data generation is suggested by Friedman (1991) as a very challenging situation where standard parametric models struggle to recover the associations. Friedman (1991) proposed the data generation with the inclusion of five irrelevant predictors, which

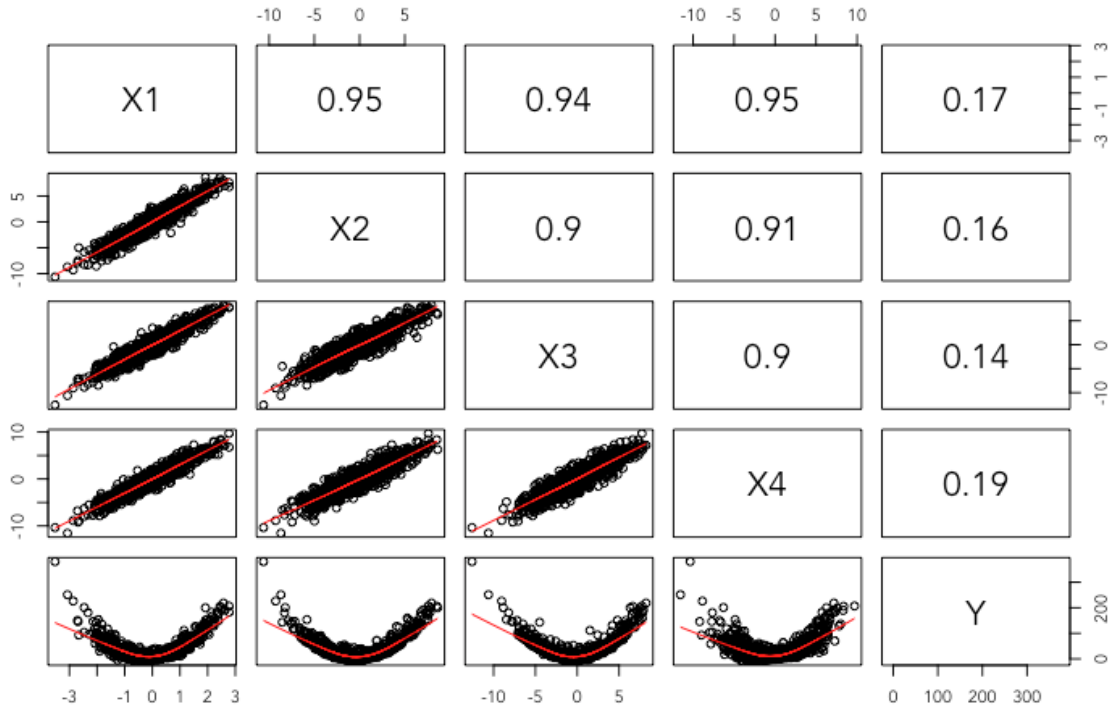


Figure 4.5: Scenario 4: in the upper part, sample pairwise correlations. In the lower part, plot of pairwise associations on simulated data according to DNL.

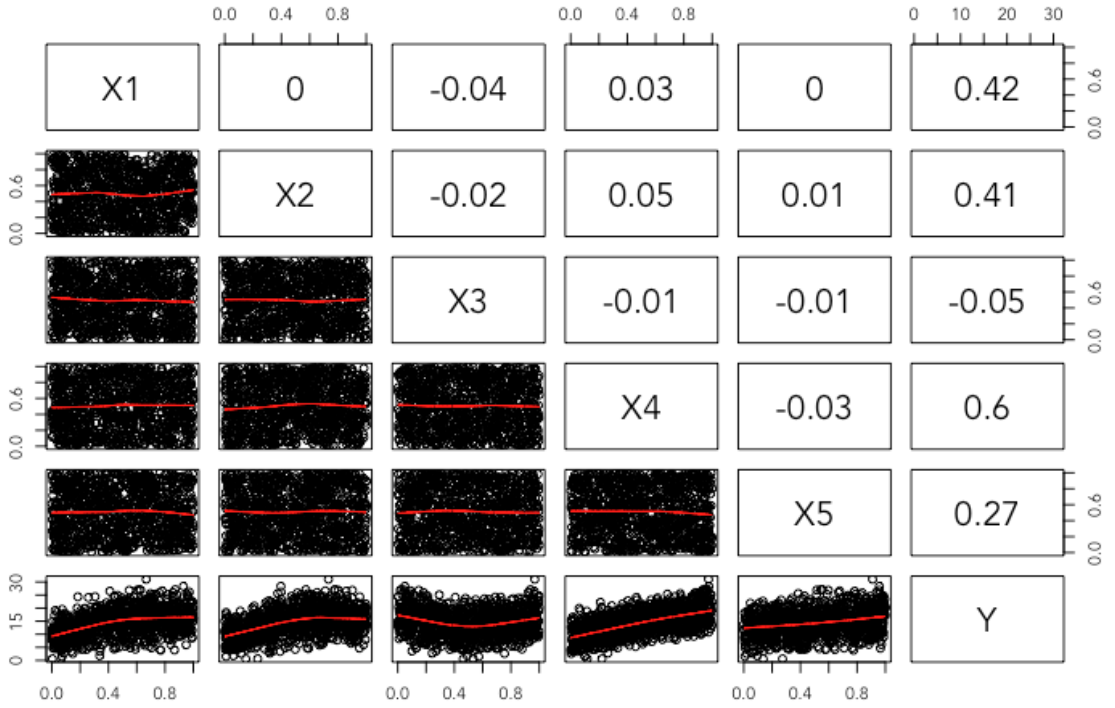


Figure 4.6: Scenario 5: in the upper part, sample pairwise correlations. In the lower part, plot of pairwise associations on simulated data according to FR.

are not included in this scenario. In Figure 4.6 the pairwise associations and

correlations among variables on simulated data are represented. I will call this scenario *FR*.

In next simulations studies, the models have been fitted with all the $p = 4$ explanatory variables of Scenario 1, 2, 3 and 4, and $p = 5$ explanatory variables of Scenario 5.

Prediction accuracy simulation study

In this case the prediction performances of the proposed algorithms are compared with respect to some algorithms described in Chapter 1. Therefore, the accuracy of predictions is evaluated on the basis of the following algorithms (with in parentheses the R packages used):

- Evolutionary estimation procedure (*EV*)
- Evolutionary estimation procedure, honest version (*HEV*)
- Two-stage estimation procedure with `rpart` for the tree estimation (*TS rpart*)
- Two-stage estimation procedure with `Ctree` for the tree estimation (*TS ctree*)
- Two-stage estimation procedure with `randomForest` for the tree estimation (*TS rf*)
- Bayesian additive regression trees (*Bart*), Chipman et al. (2010) (`bartMachine`)
- CART (*Rpart*), Breiman et al. (1984) (`rpart`)
- Random forest (*RF*), Breiman (2001) (`randomForest`)
- Conditional inference tree (*Ctree*), Hothorn et al. (2006) (`party`)
- Conditional random forest (*Cforest*), Hothorn et al. (2006) (`party`)
- Reinforcement Learning Trees (*RLT*), Zhu et al. (2015) (`rlt`)
- Extra Trees (*ExT*), Geurts et al. (2006) (`extraTrees`)

For the proposed algorithms, I employed the settings described in Section 4.2.1. For the others, the growth of trees has been performed with default settings and no pruning.

Table 1.2 summarizes the averages and the standard errors of the Monte Carlo distribution of the MSE on test sets for the 12 algorithms evaluated. The Monte Carlo distributions of the estimates of the MSE are illustrated in Figures 4.7, 4.8, 4.9, 4.10, and 4.11.

Scenario 1 represents a linear regression with a one-level factor. In this case, *EV* algorithm reaches the the minimum average of MSE estimates on test sets. The performances of the two-stage algorithms are comparable with the performance of the evolutionary algorithms. The worst MSE is obtained with *Rpart*.

Scenario 2 represents a linear regression with a four-level factor. Also in this case, *EV* algorithm reaches the the minimum average of MSE estimates on test sets. The averages estimated with he two-stage algorithms are slightly larger than those computed by the evolutionary algorithms and better than *Cforest*, *Ctree*, *RF*, *ExT*, *RTL*, and *Rpart*.

Scenario 3 represents the diamond graph of Figure 2.1, where all the associations of the relevant variables to the response are linear. In this case, *TS ctree* reaches the minimum averages of MSE estimates on test sets. However, the performances of all the evolutionary and the two-stage algorithms are comparable and lower than the other algorithms considered. In these three scenarios, combining a tree and a linear component using the two-stage algorithm results in an improvement of performances with respect to *Ctree*, *RF* and *Rpart* without the linear component.

Scenario 4 represents the diamond graph of Figure 2.1, where the associations of the relevant variables to the response are both linear and non linear. In this case, the smallest average of MSE estimates on test sets for the proposed algorithm is obtained with *TS rf*. However, the averages of MSE estimates on test sets obtained with *EV* algorithm are better that those obtained with *TS rpart* and *TS ctree*. With respect the other algorithms, the averages of MSE estimates on test sets obtained with the proposed algorithms are better that the one obtained with *Cforest*. Among all the algorithms, the minimum MSE on test sets is obtained by *Bart*.

Scenario 5 represents the Friedman data generation. In this case, for the proposed algorithm, the averages of MSE estimates on test sets obtained with *EV* algorithm are better that those obtained with *TS rpart* and *Ts rf*. Also in this case, among all the algorithms, the minimum MSE on test sets is obtained by *Bart*.

In Scenario 4 and 5, the integration of the linear component and the tree component proposed in the Semilinear regression tree do not result in an improvement of prediction performances. This confirms the proposed model to be more suitable in the case of not too heavy non linearities. Nonetheless, it is worth to notice that

despite the lack of predictive performance in case of non linearities, the proposed algorithms are able to select the most relevant variables, both in cases of linear and non linear dependences. This aspect will be highlighted in the next simulation study.

Notice that the honest version of *EV* algorithm do not improve the averages of MSE estimates on test sets. This suggest that, for a single tree, honesty do not lead to an increase in MSE.

Table 4.2: Monte Carlo averages and standard errors (in parentheses) estimates of MSE on test sets for the 12 algorithms evaluated for each simulation scenarios

	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5
EV	**1.018**	**1.084**	1.011	528.708	4.990
	(0.001)	(0.002)	(0.001)	(2.411)	(0.012)
HEV	1.028	1.122	1.021	554.208	5.177
	(0.002)	(0.002)	(0.001)	(2.660)	(0.013)
TS ctree	1.095	1.609	**1.006**	1677.014	3.883
	(0.002)	(0.011)	(0.001)	(5.985)	(0.017)
TS rf	1.060	1.377	1.011	480.724	5.880
	(0.002)	(0.010)	(0.001)	(4.695)	(0.013)
TS rpart	1.025	1.156	1.025	640.591	5.389
	(0.001)	(0.013)	(0.001)	(4.210)	(0.010)
Bart	1.129	1.148	2.018	**31.873**	**1.304**
	(0.002)	(0.002)	(0.014)	(0.778)	(0.002)
Cforest	1.578	1.574	11.096	2621.696	5.243
	(0.003)	(0.003)	(0.053)	(40.276)	(0.008)
Ctree	1.989	1.801	17.467	264.128	7.788
	(0.003)	(0.004)	(0.048)	(8.227)	(0.012)
ExT	1.976	2.172	3.238	59.131	4.572
	(0.004)	(0.005)	(0.019)	(0.828)	(0.009)
RF	1.917	1.894	3.808	66.563	3.863
	(0.004)	(0.004)	(0.019)	(0.815)	(0.007)
RLT	2.634	2.929	5.016	105.827	6.357
	(0.006)	(0.006)	(0.039)	(1.161)	(0.012)
Rpart	3.247	3.081	56.674	272.952	9.712
	(0.007)	(0.008)	(0.123)	(1.488)	(0.016)

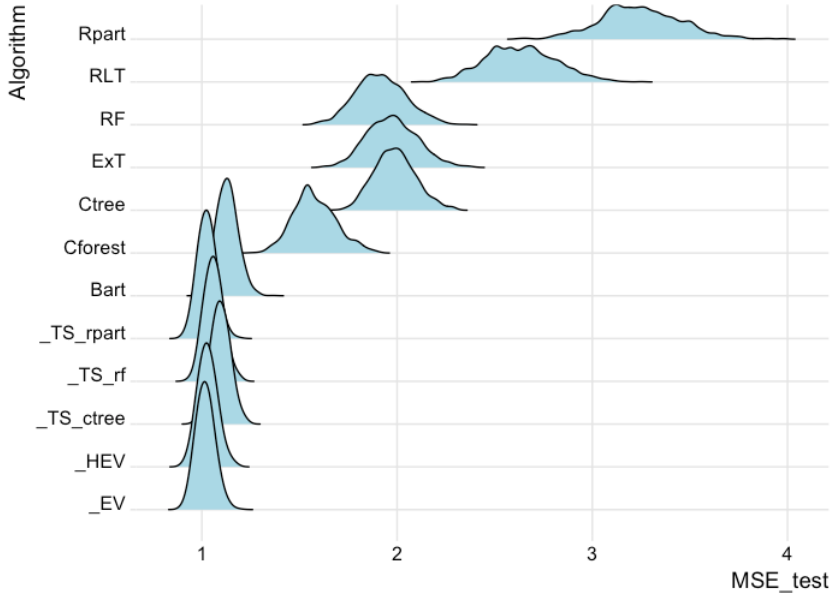


Figure 4.7: Monte Carlo distributions of MSE on test set for the 12 algorithms evaluated, data generation *LR1*.

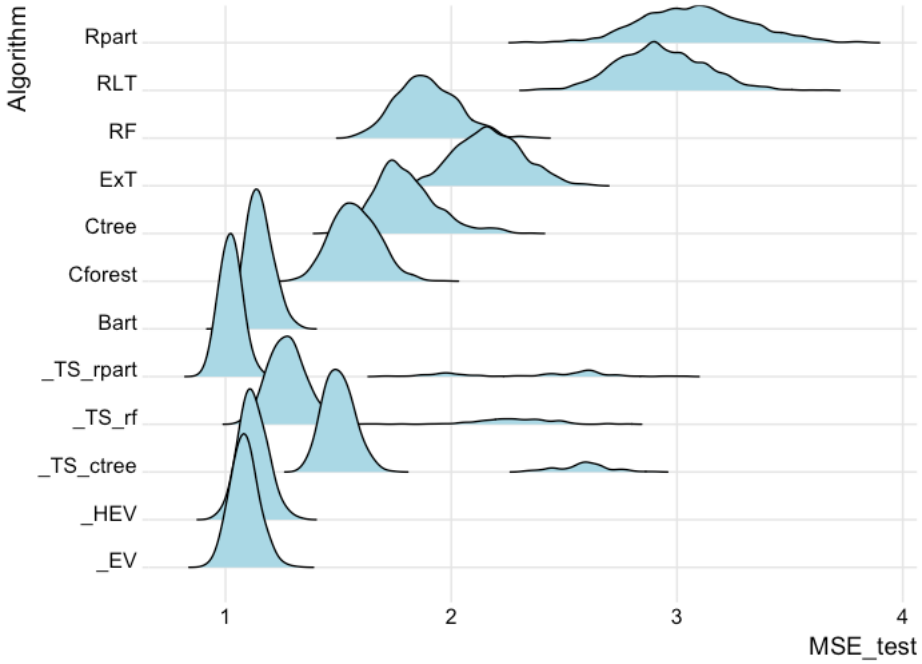


Figure 4.8: Monte Carlo distributions of MSE on test set for the 12 algorithms employed, data generation *LR2*.

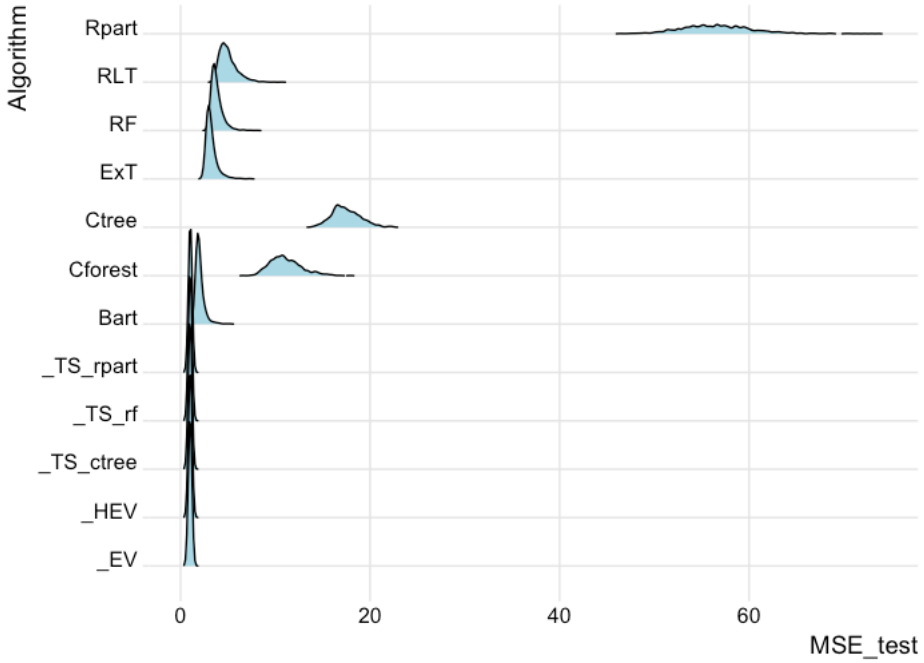


Figure 4.9: Monte Carlo distributions of MSE on test set for the 12 algorithms evaluated, data generation *DL*.

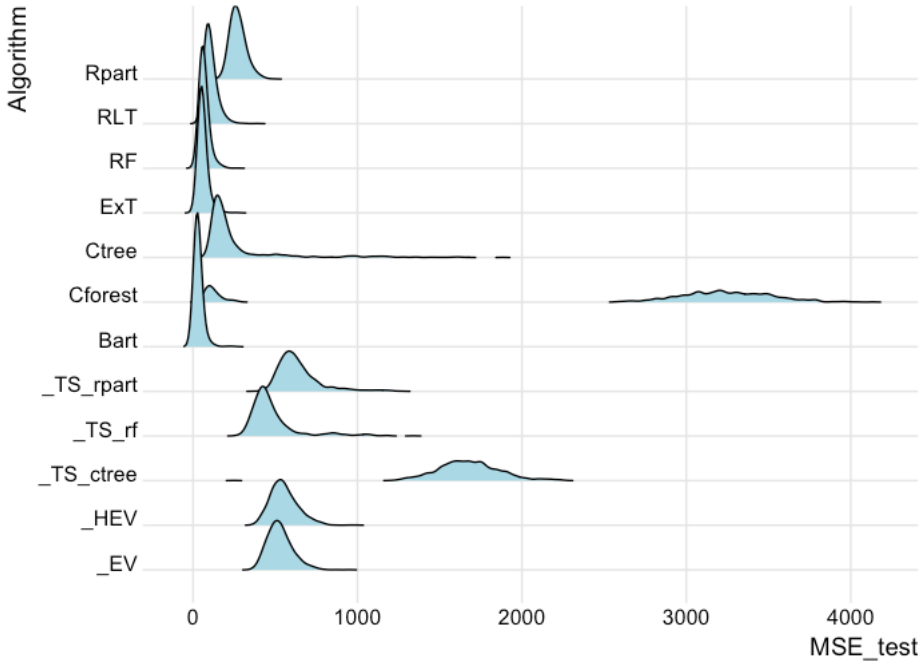


Figure 4.10: Monte Carlo distributions of MSE on test set for the 12 algorithms evaluated, data generation *DNL*.

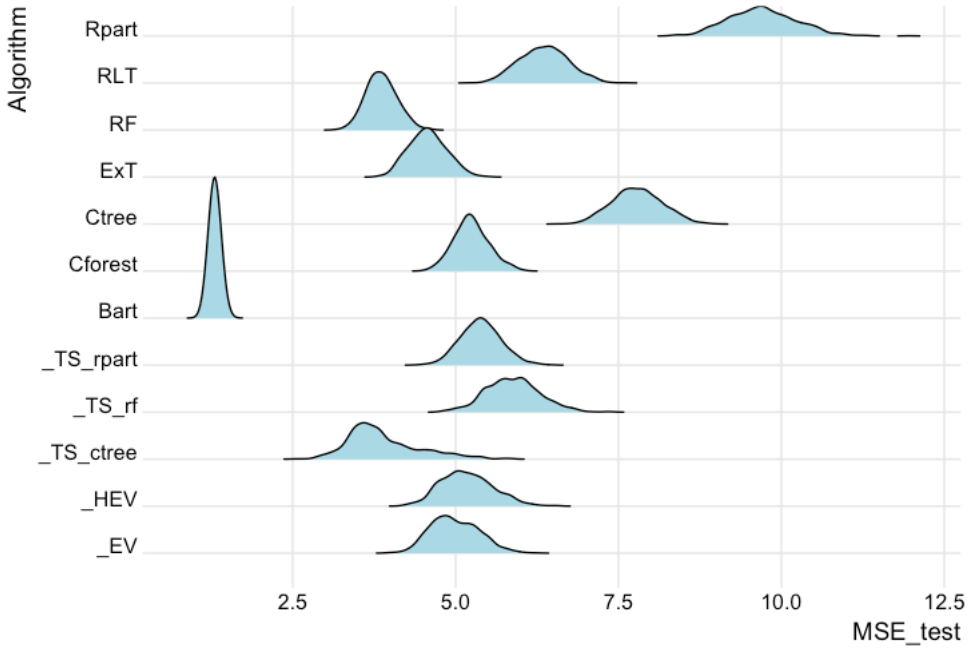


Figure 4.11: Monte Carlo distributions of MSE on test set for the 12 algorithms evaluated, data generation *FR*.

Estimation accuracy simulation study

The estimation accuracy is evaluated on the basis of the following algorithms (with in parentheses the R package used):

- Evolutionary estimation procedure (*EV*)
- Evolutionary estimation procedure, honest version (*HEV*)
- Two-stage estimation procedure with `rpart` for the tree component (*TS rpart*)
- CART (*Rpart*), Breiman et al. (1984) (`rpart`)
- Conditional inference trees (*Ctree*), Hothorn et al. (2006) (`party`)

For *Rpart* and *Ctree* trees the growth of trees has been performed with default settings and no pruning.

The comparison is limited to the two-stage estimation procedure with `rpart`, since with `party` for some scenarios the splitting is never performed, as already noted in Section 4.2.1. Moreover, the comparison of the two-stage estimation procedure with `randomForest` can be done only in terms of variable importance, since the splitting points and splitting variables are many and depend on a random sampling of variables and units. Similarly for BART and other ensemble algorithms. Therefore, these algorithms are excluded in this part of the Monte Carlo study.

Scenario 1: independent linear regressors and two-level factor

Recall that the true regression function is $Y = 3X_1 + 2X_2 + 0X_3 + 0X_4 + 3\mathbb{I}_{(X_3 \geq 0.5)} + 4\mathbb{I}_{(X_3 < 0.5)} + \epsilon$.

In Table 4.3 the proportions of splits, the averages and standard errors of the splitting points, and the averages and standard errors of the $\hat{\beta}$ parameters are shown. The proposed *EV*, *HEV*, and *TS rpart* algorithms correctly select X_3 as splitting variable. The mean of the Monte Carlo distribution of the split point is near the true value with *EV* and *HEV*, while *TS* gets exactly the true value on average. In contrast, *Rpart* and *Ctree* algorithms never select the true splitting variable, but they always select X_1. In fact (see Figure 4.1) X_1 has the highest marginal correlation with the response. The proposed *EV*, *HEV*, and *TS* algorithms identifies the true β values on average. This comparison is limited to these algorithms, since in *Rpart* and *Ctree* the linear parameters are not included. Table 4.4 reports as example the parameters of one Semilinear regression tree model estimated via OLS in *HEV* for this data generating process. As noted in Section 3.5, the advantage of the proposed evolutionary algorithm is in its use of the OLS estimator, for which we can obtain the coefficient estimates, the standard error estimates, and the test on the significance of the parameter.

Scenario 2: independent linear regressors and four-level factor

Recall that the true regression function is $Y = 3X_1 + 4X_2 + +0X_3 + 0X_4 + 3\mathbb{I}_{(X_2 \geq 1)} \cdot \mathbb{I}_{(X_1 \geq 0)} + 4\mathbb{I}_{(X_2 \geq 1)} \cdot \mathbb{I}_{(X_1 < 0)} - 1\mathbb{I}_{(X_2 < 1)} \cdot \mathbb{I}_{(X_1 \geq -1)} + 1\mathbb{I}_{(X_2 < 1)} \cdot \mathbb{I}_{(X_1 < -1)} + \epsilon$.

In Table 4.5 the proportions of splits, the averages and standard errors of the splitting points, and the averages and standard errors of the $\hat{\beta}$ parameters are shown. While *TS rpart*, *Rpart* and *Ctree* algorithms correctly select the splitting variables at both depth 1 and depth 2, *EV* and *HEV* show a different behaviour. At depth 1 the majority of splits is correctly done with X_2. At depth 2 the splitting variable is not for the majority of cases X_1, at least in one of the two partitions. Moreover, not all trees arrive to depth 2. The proportion of missing splits are reported in the column null. This is probably due to the evolutionary nature of the algorithm, which in this case favours less complex models in the tree part. For *EV* and *HEV*, the averages of splitting point are relative to the splitting variable where the split is performed. When the algorithms select the true splitting variable, both algorithms show bias in identifying the corrects splitting points at depth 2, differently from *TS rpart* algorithm that identifies them. With respect the β coefficients, both the proposed algorithms identifies the true values on average. Table 4.6 reports as example the parameters of one Semilinear regression tree model estimated via OLS in *HEV* for this data generating process.

Table 4.3: Summary of proportions of splitting for each variable, Monte Carlo averages and standard errors of the splitting points and variable importance, and Monte Carlo averages and standard errors of the $\hat{\beta}$ parameters with data generation *LR1*.

Algorithms	X_1	X_2	X_3	X_4	*No split*
EV					
Depth=1					
Proportion of splits	0.007	0.007	0.978	0.008	0.000
MC average splitting point			0.523		
MC s.e. splitting point			0.007		
Depth=2					
Proportion of splits	0.000	0.000	0.178	0.000	0.822
MC average splitting point			0.499		
MC s.e. splitting point			0.003		
$\hat{\beta}$	2.999	1.999	-0.006	0.000	
s.e.$(\hat{\beta})$	0.001	0.001	0.002	0.001	
HEV					
Depth=1					
Proportion of splits	0.003	0.004	0.984	0.009	0.000
MC average splitting point			0.511		
MC S.e. splitting point			0.006		
Depth=2					
Proportion of splits	0.000	0.000	0.090	0.000	0.910
MC average splitting point			0.494		
MC s.e. splitting point			0.005		
$\hat{\beta}$	2.999	2.000	-0.005	-0.001	
s.e.$(\hat{\beta})$	0.001	0.001	0.003	0.001	
TS rpart					
Proportion of splits	0.000	0.000	1.000	0.000	0.000
MC average splitting point			0.500		
MC s.e. splitting point			0.001		
$\hat{\beta}$	2.998	2.000	-0.089	-0.001	
s.e.$(\hat{\beta})$	0.001	0.001	0.003	0.001	
Rpart					
Proportion of splits	1.000	0.000	0.000	0.0000	0.000
MC average splitting point	0.003				
MC s.e splitting point	0.006				
Ctree					
Proportion of splits	1.000	0.000	0.000	0.000	0.000
MC average splitting point	-0.001				
MC s.e. splitting point	0.006				

Table 4.4: Example of OLS estimates of the evolutionary algorithm for the Semi-linear regression tree model, data generation *LR1*.

	Coefficient Estimate	s.e.	t-test	p-value
X1	3.0634	0.0447	68.58	0.0000
X2	2.0015	0.0436	45.91	0.0000
X3	0.0055	0.0651	0.08	0.9328
X4	-0.0236	0.0464	-0.51	0.6105
X3 $\geq$ 0.6303	2.8999	0.1154	25.12	0.0000
X3 $<$ 0.6303	3.9244	0.0613	64.03	0.0000

Scenario 3: diamond graph, linear associations

Recall that in this case the interest is that the algorithms would be able to discover the direct influences on Y. The true regression function is $Y = 3X_2 + 3X_3 + 3X_4 + \epsilon$, where all the associations of the relevant variables to the response

Table 4.5: Summary of proportions of splitting for each variable, Monte Carlo averages and standard errors of the splitting points and variable importance, and Monte Carlo averages and standard errors of the $\hat{\beta}$ parameters with data generation *LR2*.

Algorithms	X_1	X_2	X_3	X_4	*No split*
EV					
Depth=1					
Proportion of splits	0.155	0.845	0.000	0.000	0.000
MC average splitting point	-0.963	0.964			
MC s.e. splitting point	0.023	0.003			
Depth=2.1					
Proportion of splits	0.139	0.553	0.000	0.000	0.308
MC average splitting point	-0.002	1.002			
MC s.e. splitting point	-0.001	0.043			
Depth=2.2					
Proportion of splits	0.827	0.172	0.000	0.000	0.000
MC average splitting point	-0.292	0.995			
MC s.e. splitting point	-0.010	0.076			
$\hat{\beta}$	2.867	2.026	-0.001	-0.001	
$s.e.(\hat{\beta})$	0.003	0.002	0.001	0.001	
HEV					
Depth=1					
Proportion of splits	0.156	0.844	0.000	0.000	0.000
MC average splitting point	-0.801	0.956			
MC S.e. splitting point	-0.064	0.032			
Depth=2.1					
Proportion of splits	0.043	0.505	0.000	0.000	0.452
MC average splitting point	0.045	1.004			
MC s.e. splitting point	0.007	0.044			
Depth=2.2					
Proportion of splits	0.804	0.181	0.000	0.000	0.015
MC average splitting point	0.009	0.953			
MC s.e. splitting point	0.001	0.071			
$\hat{\beta}$	2.854	2.032	-0.002	-0.001	
$s.e.(\hat{\beta})$	0.004	0.003	0.001	0.001	
TS rpart					
Depth=1					
Proportion of splits	0.000	1.000	0.000	0.000	0.000
MC average splitting point		1.000			
MC s.e. splitting point		0.032			
Depth=2.1					
Proportion of splits	1.000	0.000	0.000	0.000	0.000
MC average splitting point	-1.000				
MC s.e. splitting point	-0.031				
Depth=2.2					
Proportion of splits	0.996	0.002	0.002	0.000	0.000
MC average splitting point	0.016	1.063	1.065		
MC s.e. splitting point	0.001	0.751	0.753		
$\hat{\beta}$	3.003	2.000	-0.002	-0.001	
$s.e.(\hat{\beta})$	0.001	0.001	0.001	0.001	
Rpart					
Depth=1					
Proportion of splits	0.000	1.000	0.000	0.000	0.000
MC average splitting point		0.991			
MC s.e. splitting point		0.031			
Depth=2.1					
Proportion of splits	1.000	0.000	0.000	0.000	0.000
MC average splitting point	0.249				
MC s.e. splitting point	0.008				
Depth=2.2					
Proportion of splits	1.000	0.000	0.000	0.000	0.000
MC average splitting point	0.008				
MC s.e. splitting point	0.001				
Ctree					
Depth=1					
Proportion of splits	0.000	1.000	0.000	0.000	0.000
MC average splitting point		0.988			
MC s.e. splitting point		0.001			
Depth=2.1					
Proportion of splits	1.000	0.000	0.000	0.000	0.000
MC average splitting point	0.248				
MC s.e. splitting point	0.005				
Depth=2.2					
Proportion of splits	1.000	0.000	0.000	0.000	0.000
MC average splitting point	-0.005				
MC s.e. splitting point	0.016				

are linear. Table 4.7 shows the proportions of the splitting variables choice and the averages and standard errors of the $\hat{b}$ parameters. *EV HEV* and *TS rpart* algorithms split for X_2, X_3 and X_4, the variables that have a direct influence on Y, around the 77% of times, while for X_1, the variable with an indirect effect on Y,

Table 4.6: Example of OLS estimates of the evolutionary algorithm for the Semi-linear regression tree model, data generation *LR2*.

	Coefficient Estimate	s.e.	t-test	p-value
X1	2.8133	0.0572	49.17	0.0000
X2	1.9891	0.0645	30.82	0.0000
X3	-0.0166	0.0447	-0.37	0.7109
X4	-0.0995	0.0467	-2.13	0.0336
X2 $\geq$ 0.9578 & X2 $\geq$ 1.0116	3.6410	0.1457	24.98	0.0000
X2 $\geq$ 0.9578 & X2 $<$ 1.0116	0.3487	0.4654	0.75	0.4540
X2 $<$ 0.9578 & X1 $\geq$ -0.9249	-0.9426	0.0614	-15.36	0.0000
X2 $<$ 0.9578 & X1 $<$ -0.9249	0.5209	0.1570	3.32	0.0010

Table 4.7: Summary of proportions of splitting for each variable and Monte Carlo averages and standard errors of the $\hat{b}$ parameters with data generation *DL*.

Algorithms	X_1	X_2	X_3	X_4	*No split*
EV					
Depth=1					
Proportion of splits	0.222	0.273	0.232	0.273	0.000
Depth=2.1					
Proportion of splits	0.000	0.000	0.000	0.000	1.000
Depth=2.2					
Proportion of splits	0.000	0.000	0.000	0.000	1.000
$\hat{b}$	0.002	2.999	2.999	2.999	
s.e.$(\hat{b})$	0.005	0.001	0.001	0.001	
HEV					
Depth=1					
Proportion of splits	0.224	0.264	0.250	0.262	0.000
Depth=2.1					
Proportion of splits	0.000	0.000	0.000	0.000	1.000
Depth=2.2					
Proportion of splits	0.000	0.000	0.000	0.000	1.000
$\hat{\beta}$	0.005	3.001	2.997	2.999	
s.e.$(\hat{\beta})$	0.008	0.002	0.002	0.002	
TS rpart					
Depth=1					
Proportion of splits	0.226	0.249	0.245	0.280	0.000
Depth=2.1					
Proportion of splits	0.208	0.241	0.212	0.235	0.000
Depth=2.2					
Proportion of splits	0.233	0.232	0.216	0.222	0.000
$\hat{b}$	-0.002	2.998	2.998	2.998	
s.e.$(\hat{b})$	0.006	0.001	0.001	0.001	
Rpart					
Depth=1					
Proportion of splits	1.000	0.000	0.000	0.0000	0.000
Depth=2.1					
Proportion of splits	0.983	0.008	0.004	0.005	0.000
Depth=2.2					
Proportion of splits	0.988	0.006	0.003	0.003	0.000
Ctree					
Proportion of splits	1.000	0.000	0.000	0.0000	0.000
Depth=2.1					
Proportion of splits	1.000	0.000	0.000	0.0000	0.000
Depth=2.2					
Proportion of splits	1.000	0.000	0.000	0.0000	0.000

around the 22% of times. Notice that in all Monte Carlo replications *EV* and *HEV* estimate trees of depth 1, while for *TS rpart* the behaviour at depth 2 is equal to

the one in depth 1. Conversely, *Rpart* and *Ctree* select wrongly X_1 as splitting variable at both depth 1 and 2. This pitfall has already been seen in Chapter 2. Both the evolutionary and the two-stage algorithms provide an unbiased estimates of the b parameter. Table 4.8 reports as example the parameters of one Semilinear regression tree model estimated via OLS in *HEV* for this data generating process.

Table 4.8: Example of OLS estimates of the evolutionary algorithm for the Semilinear regression tree model, data generation *DL*.

	Coefficient Estimate	s.e.	t-test	p-value
X1	-0.2938	0.2289	-1.28	0.2000
X2	3.0242	0.0441	68.60	0.0000
X3	3.0269	0.0440	68.80	0.0000
X4	3.0012	0.0465	64.51	0.0000
X4 $\geq$ -1.2139	0.0744	0.0668	1.11	0.2655
X4 $<$ -1.2139	-0.2769	0.1098	-2.52	0.0120

Scenario 4: diamond graph, non linear associations

This data generation process is the same of the previous in the structure, while the shape of associations between the response and their influencing variables is in this case non linear. The true regression function is $Y = 3X_2^2 + 3X_3^2 + 3X_4 - 3X_2X_3 + \epsilon$.

In Table 4.9 the proportions of the splitting variables choice and the averages and standard errors of the $\hat{b}$ parameters are shown. At depth 1, the algorithms *EV*, *HEV* and *TS rpart* split in most cases for X_2 and X_3, *Rpart* selects in quite equal proportions among X_1, X_2 and X_3, and *Ctree* selects in the majority of cases X_4. Note that also in this case, *Ctree* selects as primary splitting variable the one with highest marginal correlation (see Figure 4.5) with Y. At depth 2, the majority of splits of *EV*, *HEV* and *TS rpart* are for X_2 and X_3, while for *Rpart* and *Ctree* the splits selected are for X_1, X_2 and X_3, and only in a minimal proportion for X_4. Notice that the proposed algorithms select as splitting variables for the majority of cases the variables with a direct effect on Y. Moreover, the split is never performed on X_4 even if it has a direct influence on Y. This is thanks to the linear component included in the model, which capture the linear associations and leave the tree component to deal with the non linear associations. Both the evolutionary and the two-stage algorithms provide an unbiased estimate of b coefficients. Table 4.10 reports as example the parameters of one Semilinear regression tree model estimated via OLS in *HEV* for this data generating process.

Table 4.9: Summary of proportions of splitting for each variable and Monte Carlo averages and standard errors of the $\hat{b}$ parameters with data generation *DNL*.

Algorithms	X_1	X_2	X_3	X_4	*No split*
EV					
Depth=1					
Proportion of splits	0.031	0.480	0.489	0.000	0.000
Depth=2.1					
Proportion of splits	0.066	0.461	0.473	0.000	0.000
Depth=2.2					
Proportion of splits	0.071	0.463	0.466	0.000	0.000
$\hat{b}$	-0.036	0.029	-0.086	3.016	
$s.e.(\hat{b})$	0.173	0.156	0.158	0.023	
HEV					
Depth=1					
Proportion of splits	0.062	0.483	0.455	0.000	0.000
Depth=2.1					
Proportion of splits	0.102	0.458	0.431	0.009	0.000
Depth=2.2					
Proportion of splits	0.106	0.416	0.472	0.006	0.000
$\hat{b}$	0.002	-0.015	0.229	2.988	
$s.e.(\hat{b})$	0.229	0.154	0.155	0.033	
TS rpart					
Depth=1					
Proportion of splits	0.113	0.439	0.448	0.000	0.000
Depth=2.1					
Proportion of splits	0.101	0.460	0.439	0.000	0.000
Depth=2.2					
Proportion of splits	0.177	0.400	0.419	0.004	0.000
$\hat{\beta}$	-0.031	0.077	-0.083	2.971	
$s.e.(\hat{\beta})$	0.199	0.094	0.096	0.025	
Rpart					
Depth=1					
Proportion of splits	0.327	0.338	0.335	0.000	0.000
Depth=2.1					
Proportion of splits	0.101	0.460	0.439	0.000	0.000
Depth=2.2					
Proportion of splits	0.246	0.376	0.364	0.013	0.000
Ctree					
Proportion of splits	0.106	0.027	0.032	0.819	0.000
Depth=2.1					
Proportion of splits	0.131	0.387	0.405	0.007	0.000
Depth=2.2					
Proportion of splits	0.352	0.286	0.289	0	0.000

Table 4.10: Example of OLS estimates of the evolutionary algorithm for the Semi-linear regression tree model, data generation *DNL*.

	Coefficient Estimate	s.e.	t-test	p-value
X1	-0.4944	5.4226	-0.09	0.9274
X2	3.9598	1.0213	3.88	0.0001
X3	-5.4996	1.1107	-4.95	0.0000
X4	1.7099	1.0702	1.60	0.1107
X3 $\geq$ 2.7134 & X3 $\geq$ 6.2496	211.4511	8.5741	24.66	0.0000
X3 $\geq$ 2.7134 & X3 $<$ 6.2496	68.2177	3.4049	20.04	0.0000
X3 $<$ 2.7134 & X2 $\geq$ -4.4508	13.0641	1.2006	10.88	0.0000
X3 $<$ 2.7134 & X2 $<$ -4.4508	81.9481	4.9151	16.67	0.0000

Scenario 5: Friedman

The regression function of the Friedman data generation is $Y = 10\sin(\pi X_1 X_2)+$

$20(X_3 - 0.5)^2 + 10X_4 + 5X_5 + \epsilon$.

In Table 4.11 the proportions of the splitting variables choice and the averages and standard errors of the $\hat{\beta}$ parameters are shown. All the algorithms except for *EV* select X_3 as splitting variable at both depth 1 and depth 2. *EV* algorithm splits at depth 1 for X_3 around the 55% of times, while the remaining 45% of times splits for X_1 and X_2. At depth 2, in one branch the splits are almost equally distributed among X_1, X_2 and X_3, while in the other branch the majority of splits is again for X_3. Notice that in this case, all algorithms select as splitting variable the one with highest marginal correlation coefficient (see Figure 4.6), which has the highest β coefficient associated to the quadratic term. Only *EV* algorithm, the evolutionary algorithm without the honesty, selects less times this variable. Both the evolutionary and the two-stage algorithms provide an unbiased estimate of β parameters relative to the interactions and linear associations. Table 4.12 reports as example the parameters of one Semilinear regression tree model estimated via OLS in *HEV* for this data generating process.

Table 4.11: Summary of proportions of splitting for each variable and Monte Carlo averages and standard errors of the $\hat{\beta}$ parameters with data generation *FR*.

Algorithms	X_1	X_2	X_3	X_4	X_5	*No split*
EV						
Depth=1						
Proportion of splits	0.208	0.247	0.545	0.000	0.000	0.000
Depth=2.1						
Proportion of splits	0.311	0.277	0.290	0.000	0.000	0.000
Depth=2.2						
Proportion of splits	0.135	0.138	0.727	0.000	0.000	0.000
$\hat{\beta}$	7.583	7.604	-0.287	10.002	4.997	
$s.e.(\hat{\beta})$	0.040	0.043	0.127	0.008	0.008	
HEV						
Depth=1						
Proportion of splits	0.001	0.001	0.998	0.000	0.000	0.000
Depth=2.1						
Proportion of splits	0.000	0.000	1.000	0.000	0.000	0.000
Depth=2.2						
Proportion of splits	0.000	0.000	1.000	0.000	0.000	0.000
$\hat{\beta}$	0.011	-0.010	-20.061	9.958	4.990	
$s.e.(\hat{\beta})$	0.023	0.024	0.376	0.022	0.022	
TS rpart						
Depth=1						
Proportion of splits	0.000	0.000	1.000	0.000	0.000	0.000
Depth=2.1						
Proportion of splits	0.000	0.000	1.000	0.000	0.000	0.000
Depth=2.2						
Proportion of splits	0.000	0.000	1.000	0.000	0.000	0.000
$\hat{\beta}$	-0.015	0.0006	-19.972	9.986	4.998	
$s.e.(\hat{\beta})$	0.017	0.017	0.017	0.017	0.017	
Rpart						
Depth=1						
Proportion of splits	0.000	0.000	1.000	0.000	0.000	0.000
Depth=2.1						
Proportion of splits	0.000	0.000	0.996	0.040	0.000	0.000
Depth=2.2						
Proportion of splits	0.000	0.000	1.000	0.000	0.000	0.000
Ctree						
Proportion of splits	0.000	0.000	1.000	0.000	0.000	0.000
Depth=2.1						
Proportion of splits	0.000	0.000	1.000	0.000	0.000	0.000
Depth=2.2						
Proportion of splits	0.000	0.000	0.000	1.000	0.000	0.000

Table 4.12: Example of OLS estimates of the evolutionary algorithm for the Semi-linear regression tree model, data generation *FR*.

	Coefficient Estimate	s.e.	t-test	p-value
X1	0.5972	0.7432	0.80	0.4220
X2	0.2581	0.7251	0.36	0.7220
X3	-12.3471	1.1209	-11.02	0.0000
X4	10.0334	0.7254	13.83	0.0000
X5	3.7473	0.7388	5.07	0.0000
X3 $\geq$ -1.1158 & X3 $\geq$ 1.6425	87.9943	3.8986	22.57	0.0000
X3 $\geq$= -1.1158 & X3 $<$ 1.6425	14.7746	0.8190	18.04	0.0000
X3 $<$ -1.1158 & X3 $\geq$ -1.9662	60.9909	2.9060	20.99	0.0000
X3 $<$ -1.1158 & X3 $<$ -1.9662	168.2362	6.4756	25.98	0.0000

4.3 Real data study

In this Section, two applications based on real data are presented. Both data sets employed, *Carseats* and *Boston housing*, are available in R software. In the next paragraphs, after a brief description of the data, the SRT model estimated trough the evolutionary and the two-stage methods data will be discussed. Then, the performances of the proposed method will be compared with the performances of other estimation methods as in Section 4.2.

4.3.1 Carseats data

Carseats in the `ISLR` library is a simulated data set containing sales of child car seats at 400 different stores. The Y variable is Sales, the unit sales (in thousands) at each location, and it is measured for $n = 400$ observations. Here it has been considered $p = 5$ continuous predictors, that are

- CompPrice: the price charged by competitor at each location;
- Income: the community income level (in thousands of dollars);
- Advertising: the local advertising budget for company at each location (in thousands of dollars);
- Population: the population size in region (in thousands);
- Price: the price that company charges for car seats at each site.

In Figure 4.12 the sample pairwise correlations and the pairwise associations among variables are represented.

A random subsample of the data has been selected to obtain $n_{\texttt{train}} = 200$ and $n_{\texttt{test}} = 200$. Again, for the proposed algorithms, I employed the settings described in Section 4.2.1. For the others, the growth of trees has been performed with default settings and no pruning.

In Figure 4.13 are shown the MSE on the test set for *Carseats* data. In this case, the proposed methods outperform the others, reaching the minimum MSE on the test set.

Table 4.13 and 4.14 report the summary of the SRT parameter estimation, respectively, obtained via *EV* and *HEV* algorithms. According to the model in table 4.13, CompPrice, Income, Advertising and Price have a significant effect on the response. The tree estimated has Population as splitting variable and has two terminal nodes, both with a significant effect on the response. *HEV* algorithm estimate a model with a similar behaviour. As already noted, the advantage in the evolutionary estimation procedure is in the use of the OLS estimator also

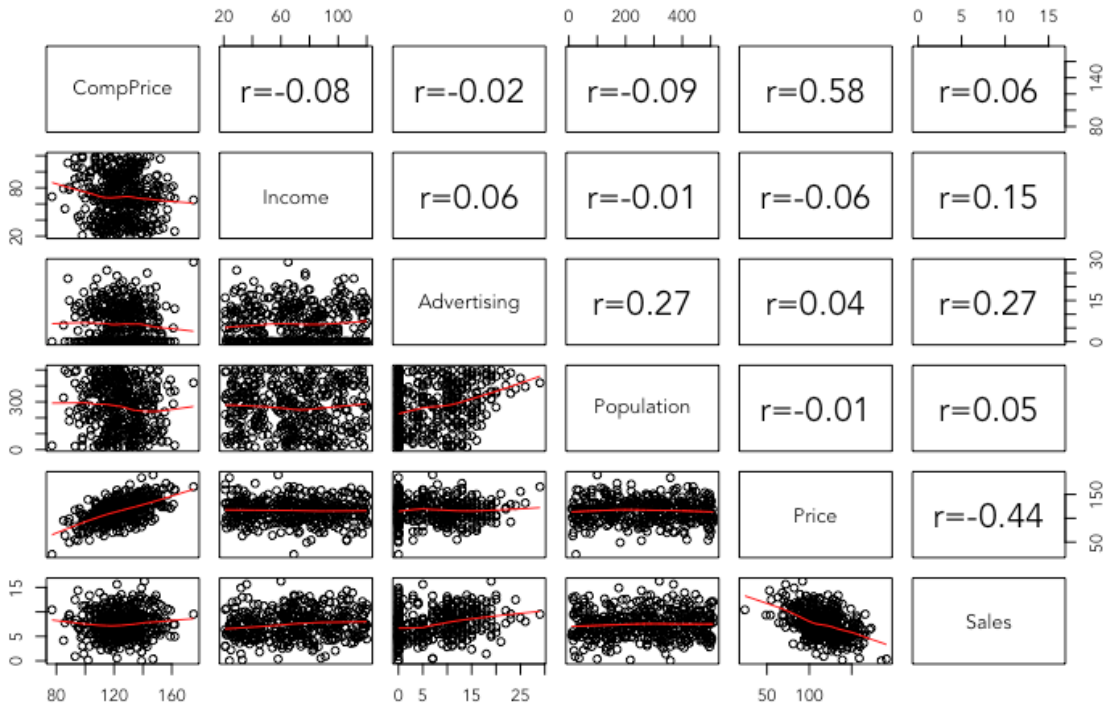


Figure 4.12: Carseats data, Sales versus 5 continuous explanatory variables: in the upper part, sample pairwise correlations. In the lower part, plot of pairwise associations.

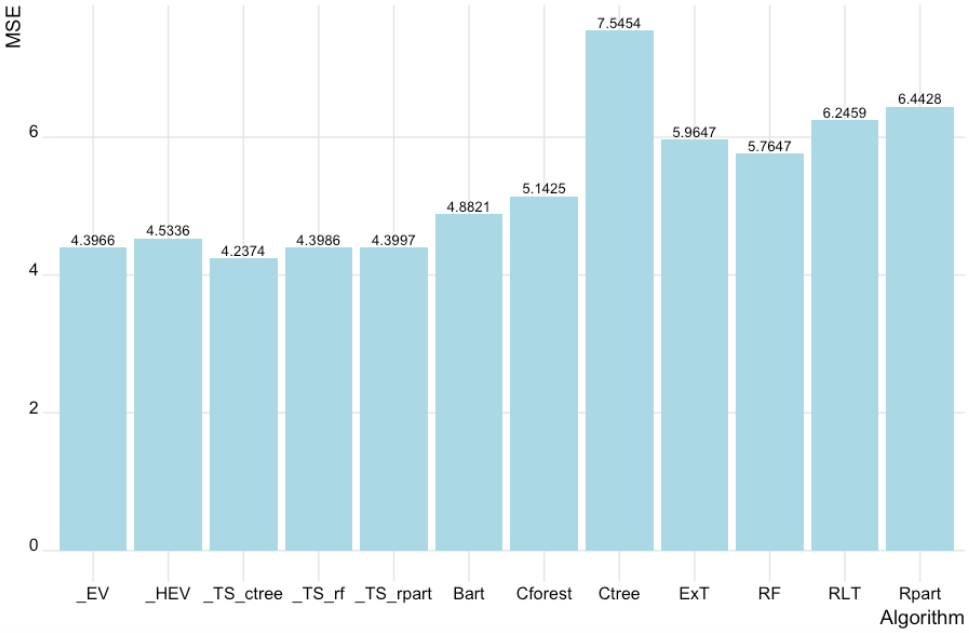


Figure 4.13: Comparison of the MSE on test set in the proposed algorithms (*EV*, *HEV*, *TS rpart*, *TS ctree*, *TS rf*) and in other 7 algorithms (*Rpart*, *Ctree*, *Bart*, *RF*, *Cforest*, *RLT*, *ExT*), data Carseats.

for the tree part, that gives the significance of the coefficients associated to the tree. Moreover, the model is interpreted as the linear regression models, with the coefficients representing the partial effect of a covariate on the response, and the

factors representing different average of the response in the subgroup defined by the factor.

In table 4.15 are reported coefficients estimated with *TS ctree*, *TS rpart* and *TS rf*. While all the coefficients estimated for the direct effect are similar among each other for these 3 algorithms, it is possible to note that with respect *EV* and *HEV* algorithms on obtain different coefficients for Population. In fact, for the evolutionary algorithms Population has a negative effect on the response, while for the two-stage algorithm it has a positive effect on the response. The tree estimated with *TS ctree* has no splits, while the tree estimated with *TS rpart* has tree splits, performed on Population at depth 1 and Price at 2. Therefore, *TS rpart* algorithm find an interaction between Population and Price. Finally, *TS rf* gives the highest variable importance to Price. Recall that, as already noted in Section 4.2.1, *TS ctree* gives tree without split because of the algorithm for the tree construction itself. As noted in the simulation study, the differences between the estimates of the evolutionary and the two-stage algorithms can be ascribed to the different nature of the estimation procedure.

Table 4.16 shows the variable importance estimated with *Rpart*, *Ctree*, *Bart*, *RF*, *Cforest*, *RLT* and *ExT* algorithms. For all these estimation procedures, the highest variable importance is reached by the variable Price.

Table 4.13: Summary of the Semilinear regression tree model, parameter estimation via OLS in *EV* algorithm, *Carseats* data.

	Coefficient Estimate	s.e.	t-test	p-value
CompPrice	1.5301	0.1866	8.20	0.0000
Income	0.4704	0.1450	3.24	0.0014
Advertising	0.8765	0.1523	5.76	0.0000
Population	-0.5022	0.2698	-1.86	0.0642
Price	-2.1415	0.1780	-12.03	0.0000
Population >= -0.4535	8.0842	0.2500	32.34	0.0000
Population < -0.4535	6.5531	0.3823	17.14	0.0000

As a final comment, for this study on Carseats data, the proposed algorithms show the best performance in the MSE on the test sets. Since the estimation via the algorithms as *Rpart*, *Ctree*, *Bart*, *RF*, *Cforest*, *RLT* and *ExT* would highlight only Price as variable most relevant to predict the response, the proposed methods have been helpful to discover a possible interaction between Population and Price. Note that the evolutionary algorithm gives the significance of the parameter estimates, that is a key point of this procedure.

Table 4.14: Summary of the Semilinear regression tree model, parameter estimation via OLS in *HEV* algorithm, *Carseats* data.

	Coefficient Estimate	s.e.	t-test	p-value
CompPrice	1.5700	0.2598	6.04	0.0000
Income	0.4270	0.2121	2.01	0.0470
Advertising	0.6684	0.2233	2.99	0.0035
Population	-0.7136	0.3637	-1.96	0.0528
Price	-2.1750	0.2602	-8.36	0.0000
Population >= -0.7792	8.3183	0.3459	24.05	0.0000
Population < -0.7792	5.9271	0.5853	10.13	0.0000

Table 4.15: Summary of proportions of splitting variables, splitting points (or variable importance) and linear parameters for *TS ctree*, *TS rpart* and *TS rf* algorithms, *Carseats* data.

	CompPrice	*Income*	*Advertising*	*Population*	*Price*	*Mean leaves*
TS CTREE						
Coefficient estimates	1.4768	0.4178	0.9275	0.1152	-2.1075	
Split: None						
TS RPART						
Coefficient estimates	1.5585	0.4010	0.9698	0.6537	-2.3164	
$\mathbb{I}_{(Population \geq 0.4082)\cdot(Price<1.1106)}$						6.4178
$\mathbb{I}_{(Population \geq 0.4082)\cdot(Price \geq 1.1106)}$						7.8137
$\mathbb{I}_{(Population<0.4082)\cdot(Price<0.2410)}$						7.6142
$\mathbb{I}_{(Population<0.4082)\cdot(Price \geq 0.2410)}$						8.6499
TS RF						
Coefficient estimates	1.5588	0.4343	0.9198	0.1107	-2.0422	
Variable importance	3.6899	14.1801	24.3866	9.5326	25.4883	

Table 4.16: Variables importance estimated with *Rpart*, *Ctree*, *Bart*, *RF*, *Cforest*, *RLT* and *ExT* algorithms, *Carseats* data.

	CompPrice	*Income*	*Advertising*	*Population*	*Price*
BART	0.2250	0.1494	0.1776	0.1443	0.3038
CFOREST	0.6917	0.2988	1.5608	-0.0256	4.1336
CTREE	0.0432	0.0728	2.5134	-0.3465	4.5745
RF	233.5010	254.2103	235.0555	240.2288	406.4031
RLT	0.0305	0.0169	0.0868	0.0055	0.1908
RPART	254.8877	154.4109	208.0568	95.8542	393.8215

4.3.2 Boston Housing data

The `MASS` library contains the Boston data set, which records medv (median house value) for $n = 506$ neighbourhoods around Boston. The explanatory variables are $p = 14$, 12 continuous variables and 2 categorical variables. In this application, the explanatory variables considered are the $p = 12$ continuous predictors, that

are

- crim: per capita crime rate by town;
- zn: proportion of residential land zoned for lots over 25000 sq.ft;
- indus: proportion of non-retail business acres per town;
- nox nitrogen oxides concentration (parts per 10 million);
- rm average number of rooms per dwelling;
- age: proportion of owner-occupied units built prior to 1940;
- dis weighted mean of distances to five Boston employment centres;
- tax: full-value property-tax rate per $10,000$ dollars;
- ptratio: pupil-teacher ratio by town;
- black: $1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town;
- lstat: lower status of the population (percent).

In Figures 4.14 and 4.15, the pairwise associations and correlations among variables on Boston data are represented. Note that the Y variable is present in both the Figures to help the reading of the correlations among the explanatory variables and the response.

A random subsample of the data has been selected to obtain $n_{\texttt{train}} = 253$ and $n_{\texttt{test}} = 253$.

In Figure 4.13 are shown the MSE on the test set for *Boston* data. In this case, among the proposed methods, *TS rf* shows the minimum MSE, and the MSE are better than those obtained with *Ctree*, *Cforest* and *Rpart*. However, the best MSE is reached by *Bart*, followed by *ExT* and *RF*. This suggest that for this application, ensemble methods are more suitable to obtain optimal predictive performances.

Table 4.17 and 4.18 reports the summary of the SRT parameter estimation, respectively, obtained via *EV* and *HEV* algorithms. According to the model in table 4.17, crim, rm, dis, rad, ptratio and lstat have a significant effect on the response. The tree estimated has rm and lstat as splitting variables and has four terminal nodes, all with a significant effect on the response. *HEV* algorithm estimate a model for which crim, ptratio and lstat have a significant effect on the response, and a shorter tree with two terminal nodes defined by rm variable.

In table 4.19 are reported coefficients estimated with *TS ctree*, *TS rpart* and *TS rf*. The tree estimated with *TS ctree* has no splits, while the tree estimated

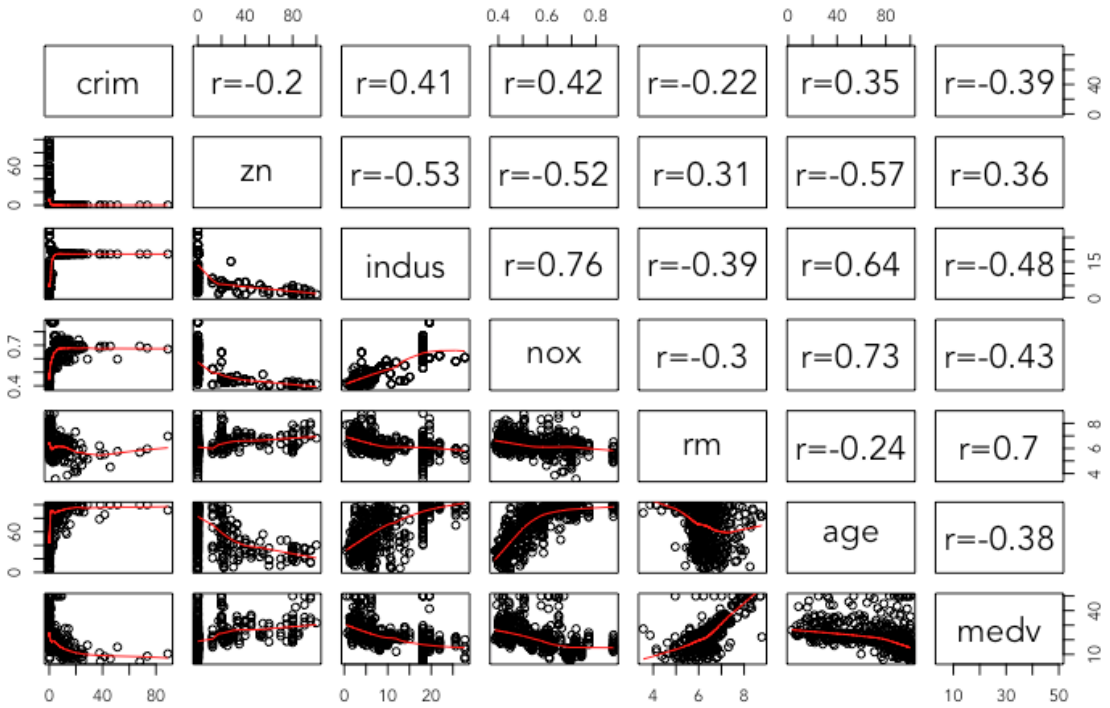


Figure 4.14: Boston data, medv versus first 6 explanatory variables: in the upper part, sample pairwise correlations. In the lower part, plot of pairwise associations.

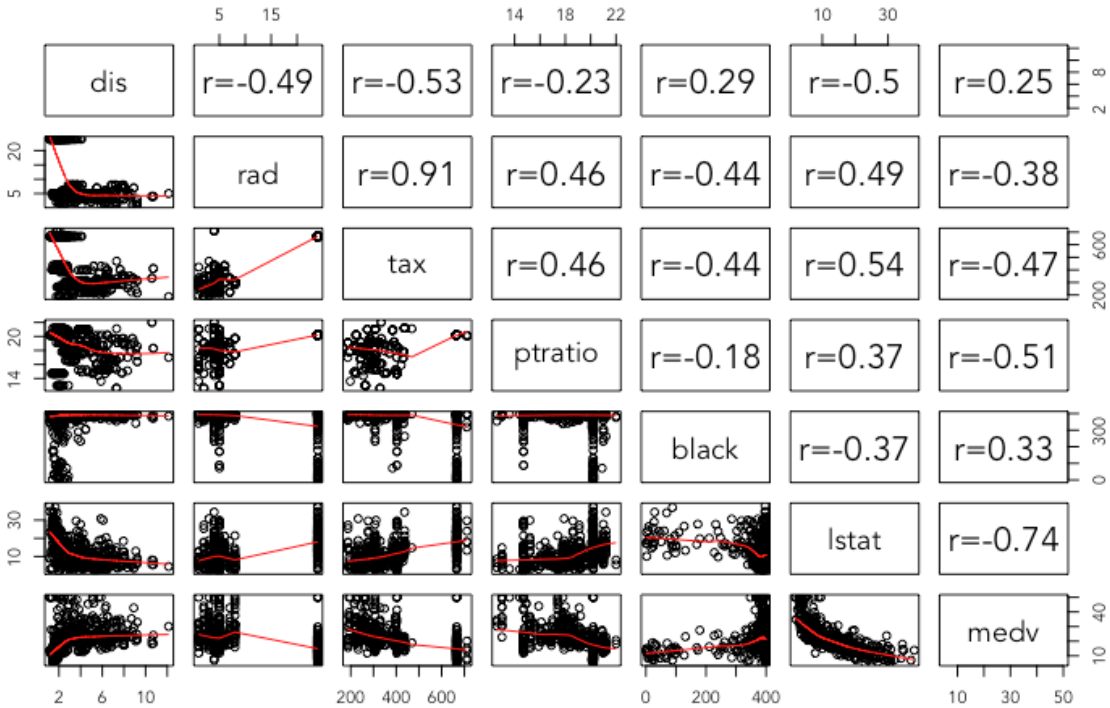


Figure 4.15: Boston data, medv versus last 6 explanatory variables: in the upper part, sample pairwise correlations. In the lower part, plot of pairwise associations.

with *TS rpart* has tree splits, performed on rm at depth 1 and age at 2. Therefore,

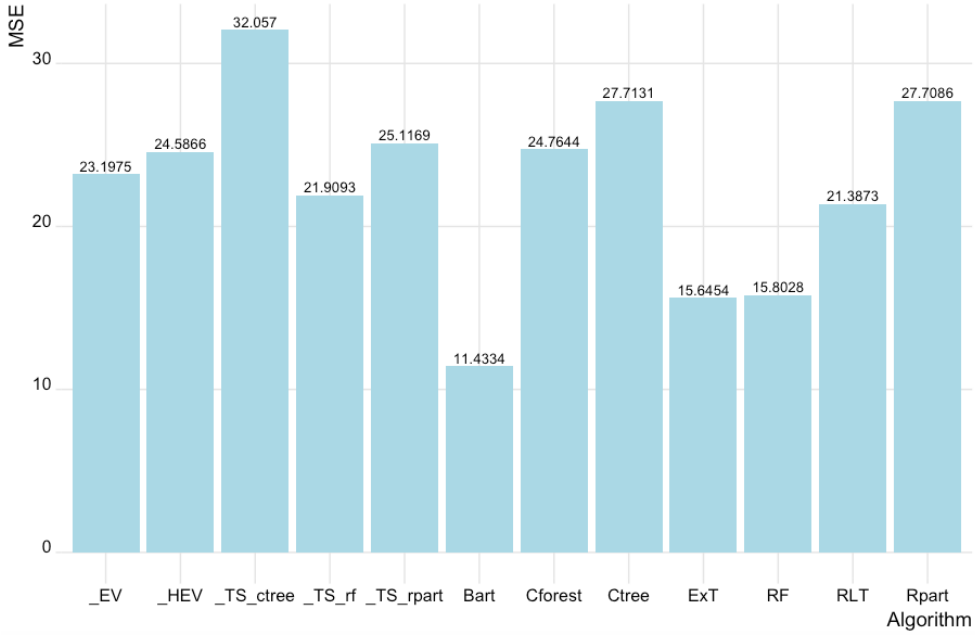


Figure 4.16: Comparison of the MSE on test set in the proposed algorithms (*EV*, *HEV*, *TS rpart*, *TS ctree*, *TS rf*) and in other 7 algorithms (*Rpart*, *Ctree*, *Bart*, *RF*, *Cforest*, *RLT*, *ExT*), data Boston.

TS rpart algorithm find an interaction between rm and age, differently from the evolutionary algorithms that find rm and lstat. Finally, *TS rf* gives the highest variable importance to rm.

Table 4.20 shows the variable importance estimated with *Rpart*, *Ctree*, *Bart*, *RF*, *Cforest*, *RLT* and *ExT* algorithms. For all these estimation procedures, the highest variable importances are reached by rm and lstat.

As a final comment, for this study on Boston data, the proposed algorithms do not show the best performance in the MSE on the test sets, while all the estimation procedures highlight rm and lstat as the variables relevant in the tree part. Therefore, as noted in the prediction accuracy simulation study, the proposed algorithms do not boast the best predictive performance in the presence of strong non linearities in the data, but they are still helpful to discover the most relevant variables that have a direct influence on the response. This happens not simply choosing the variables with the highest marginal correlation coefficients. When the variables with a direct influence on the response match the variables with the highest marginal correlation coefficients, ensemble methods possibly reach best performances.

Table 4.17: Summary of the Semilinear regression tree model, parameter estimation via OLS in *EV* algorithm, *Boston* data.

	Coefficient Estimate	s.e.	*t*-test	*p*-value
crim	-1.3010	0.2638	-4.93	0.0000
zn	0.8623	0.3389	2.54	0.0116
indus	0.3983	0.4127	0.97	0.3355
nox	-1.4435	0.4340	-3.33	0.0010
rm	3.1373	0.5088	6.17	0.0000
age	-0.4456	0.3724	-1.20	0.2327
dis	-2.0594	0.4313	-4.77	0.0000
rad	2.2755	0.5707	3.99	0.0000
tax	-1.8578	0.6082	-3.06	0.0025
ptratio	-1.2300	0.2971	-4.14	0.0000
black	0.5168	0.2377	2.18	0.0307
lstat	-3.7088	0.4484	-8.27	0.0000
rm >= -0.8221 & rm >= 1.6102	29.0539	1.3464	21.58	0.0000
rm >= -0.8221 & rm < 1.6102	21.1746	0.2565	82.54	0.0000
rm < -0.8221 & lstat >= 2.1939	31.1172	1.5627	19.91	0.0000
rm < -0.8221 & lstat <2.1939	23.7527	0.8441	28.14	0.0000

Table 4.18: Summary of the Semilinear regression tree model, parameter estimation via OLS in *HEV* algorithm, *Carseats* data.

	Coefficient Estimate	s.e.	*t*-test	*p*-value
crim	-1.3603	0.3576	-3.80	0.0002
zn	0.7605	0.6369	1.19	0.2350
indus	0.2530	0.6921	0.37	0.7153
nox	-1.9651	0.8439	-2.33	0.0217
rm	1.4604	0.8024	1.82	0.0714
age	-0.8022	0.6466	-1.24	0.2173
dis	-2.6966	0.7847	-3.44	0.0008
rad	1.9580	0.9673	2.02	0.0453
tax	-1.0368	0.9898	-1.05	0.2971
ptratio	-1.6899	0.5204	-3.25	0.0015
black	0.5621	0.4247	1.32	0.1884
lstat	-3.1315	0.7208	-4.34	0.0000
rm >= 0.9897	30.8377	1.8672	16.56	0.0000
rm < 0.9897	21.6309	0.4463	48.47	0.0000

Table 4.19: Summary of proportions of splitting variables, splitting points (or variable importance) and linear parameters for *TS ctree*, *TS rpart* and *TS rf* algorithms, *Boston* data.

	crim	*zn*	*indus*	*nox*	*rm*	*age*	*dis*	*rad*	*tax*	*ptratio*	*black*	*lstat*	*Mean leaves*
TS CTREE													
Coefficient estimates	-1.1916	0.8965	0.1738	-1.6619	4.3039	-1.0720	-2.8281	1.8492	-1.8091	-1.7745	0.7489	-1.7529	
Split point: None													
TS RPART													
Coefficient estimates	-1.4787	0.7662	0.1692	-1.6171	2.3106	-1.0678	-2.1777	1.9044	-1.5140	-1.3259	0.7598	-2.5011	
$\mathbb{I}_{(rm \geq 1.6486)}$													21.3343
$\mathbb{I}_{(rm < 1.6486) \cdot (age \geq 1.1004)}$													26.3925
$\mathbb{I}_{(rm < 1.6486) \cdot (age < 1.1004)}$													32.1167
TS RF													
Coefficient estimates	-1.2370	0.4968	0.1173	-1.5202	3.1710	-1.0171	-2.3135	1.6800	-1.7240	-1.3807	0.5988	-2.0538	
Variable importance	0.0000	0.0000	5.2840	41.1295	652.0286	179.6572	590.6338	0.0000	18.0566	61.6426	0.0000	445.6151	

Table 4.20: Variable importance measures estimated with *Rpart*, *Ctree*, *Bart*, *RF*, *Cforest*, *RLT* and *ExT* algorithms, *Boston* data.

	crim	*zn*	*indus*	*nox*	*rm*	*age*	*dis*	*rad*	*tax*	*ptratio*	*black*	*lstat*
BART	0.0795	0.0263	0.0490	0.1013	0.1410	0.0786	0.1368	0.0519	0.0565	0.0643	0.0674	0.1475
CFOREST	1.3486	0.1657	2.6203	2.4947	35.2609	1.4427	0.3182	0.2204	2.0549	1.4634	0.4759	32.7299
CTREE	0.0000	0.0000	4.2533	0.0000	46.7646	0.3969	1.6137	0.0000	1.7587	0.0000	1.9379	14.0504
RF	857.7721	162.1604	1247.1435	1424.2394	5659.7297	675.4131	856.7262	105.7809	447.8605	732.6564	338.4824	5751.9526
RLT	0.1035	0.0781	0.1101	0.1513	0.8007	0.0976	0.0855	0.0417	0.0864	0.1477	0.0428	1.1143
RPART	2080.8586	1117.3932	3956.0966	2268.1030	11477.0757	1807.7136	2218.5016	598.4113	270.4641	1364.4294	0.0000	9700.4615

Chapter 5

Concluding remarks

Regression trees are powerful and simple models that can easily handle with non linear relationship and interactions among variables. They are appreciated for being easy interpretable, because of the visual diagram that depicts as a tree the partition selected by the algorithm. In terms of prediction accuracy, regression trees do not boast great predictive performances, whereas can be done by random forests or other ensemble methods.

Since regression trees and random forests are sometimes improperly used to discover the relevant variable for the response, I presented some pitfalls that confirm that essentially predictive models can provide misleading information on the data generating process. For some data generating process, variables importance can systematically incorrectly identify direct dependencies. This pitfalls highlight that variable importance measures are not able to distinguish between background variables with indirect effects and intermediate variables with direct effects. The results of the Monte Carlo study confirm this hypothesis for the algorithms based on greedy search (CART, RF and CRF). Interestingly, the reinforcement learning trees fails on the task, even if its variable importance has been proven to be consistent for independent covariates. On the other hand, BART, which is not based on a greedy search, correctly gives higher importance to the variables having direct influence. It is therefore evident how dangerous is to interpret the variables selected by algorithms based on greedy search as representative of the generative model.

To overcome this pitfall and to propose a model capable of treat linear and non linear relationships, in this dissertation it has been proposed the Semilinear regression tree model along with two new different estimation procedures. The SRT is a semiparametric model composed of two parts: a linear component and a tree component. This idea of such a model was actually proposed as a possible extension to BART by Chipman et al. (2010). To estimate the SRT model parameters I proposed two new procedures: a two-stage estimation procedure (TS)

based on a backfitting algorithm (Buja et al., 1989), and an estimation procedure based on an evolutionary algorithm (EV) (Grubinger et al., 2011). The key point in the proposed two-stage algorithm is in the iterative use of the residuals from the two components of the model. This makes possible to extract the information about the relationship between the response and the covariate by subtracting each time the variability explained by both models. For the evolutionary algorithm, the key point is the use of the OLS estimator, which allows to simultaneously obtain the parameter estimates of the tree and the linear components of the model and release the greedy search of the partitions of the tree. Moreover, the evolutionary perturbation step of the tree component allows to quickly explore a wide space of models. This results in a highly interpretable model, being essentially an ordinary linear regression model, for which, if the assumption of Normality of the error term ϵ is added as in the BART algorithm, it is possible to perform statistical hypothesis testing and to construct reliable confidence intervals for parameter estimates.

To asses the performances of the proposed estimation procedures, I conduct a Monte Carlo study on five different data generation scenarios. The estimation accuracy has been evaluated comparing the proposals with CART and Conditional inference trees (Ctree). When the data generating process has linear and quasi-linear dependencies, *EV* and *TS* select as splitting variables the one that influence directly the response, while CART and Ctree always select as splitting variable the one with the highest marginal correlation with the response, even if, for one case, it is conditionally independent of it. Also when the data generating process has non linearities, *EV* and *TS* select as splitting variables which direct influence the response, whit a major proportion assigned to the variables associated in a non linear way with the response. Conversely, CART at first split assigns an equal proportion to variables with direct and indirect effect, while Ctree select as first splitting variable in the majority of cases the one with highest marginal correlation coefficient associated in a linear way with the response. The prediction accuracy has been evaluated on the basis of a variety of algorithms. In data generating process with linear and quasi-linear dependencies, *EV* and *TS* reach the minimum MSE among all the methods, showing better performance than ensemble methods as random forests and BART. Conversely, in the two case of data generating process with non linearity *EV* and *TS* shows performance better than *Cforest* in one case, and better than *Ctree*, *ExT* and *RLT* in the other. Moreover, the proposed algorithms have been compared in two real data study, confirming the general behaviour highlighted in the Monte Carlo simulations. Even if the proposed algorithms do not boast the best predictive performance in presence of strong non linearities in the data, they are still helpful to discover the most

relevant variables that has a direct influence on the response, not simply choosing the variables with the highest marginal correlation coefficients.

In summary, the Monte Carlo study performed for the model evaluation show that both the proposed estimation procedures are able to capture the data generating process in the case of linear and quasi-linear dependencies, while the other algorithms fail. In addition, in these cases, they show the best performances also in terms of predictive accuracy. On the contrary the proposed model seems not enough flexible in case of strong non linearities, even if it is able to capture the data generating process also in this case. Future research will focus on extensions of the SRT model to overcome this issue. A possibility is to add to the linear component some spline function. In that way, the SRT model would become a more flexible additive-type model, which would be able to handle the linearities and non linearities via the spline functions and the interactions via the tree part. In fact, as shown by the simulation study in Chapter 4, handle both non linearities and interactions with the tree can be too restrictive, and the model can show difficulties in recovering the data structure. Nonetheless, a problem that can be met with this extension is the overfitting. Another improvement in performances could be reached by an interaction term between the linear component and the tree in the case of a small number of predictors. Moreover, further extensions of the SRT model could be to consider different link functions and different kind of responses, such as binary, multinomial, Poisson or survival. For the estimation procedures, they can easily be extended to a regularized estimate of parameters as LASSO or adaptive LASSO (Zou, 2006) in high dimensional frameworks. For example, in the two-stage estimator the parameters of the linear component would be estimated via the LASSO, and the variables considered in the tree construction could be only the variables selected after the regularization.

Bibliography

AA.VV. (2016). General data protection regulation. *Official Journal of the European Union 59*(1-88), 294.

Alexander, W. P. and S. D. Grimshaw (1996). Treed regression. *Journal of Computational and Graphical Statistics 5*(2), 156–175.

Amit, Y. and D. Geman (1997). Shape quantization and recognition with randomized trees. *Neural computation 9*(7), 1545–1588.

Athey, S., J. Tibshirani, and S. Wager (2018). Generalized random forests. *Annals of Statistics*.

Back, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press.

Biau, G., L. Devroye, and G. Lugosi (2008). Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research 9*(Sep), 2015–2033.

Breiman, L. (1996). Bagging predictors. *Machine learning 24*(2), 123–140.

Breiman, L. (2000). Some infinity theory for predictor ensembles. Technical report, Technical Report 579, Statistics Dept. UCB.

Breiman, L. (2001). Random forests. *Machine learning 45*(1), 5–32.

Breiman, L. et al. (2001). Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science 16*(3), 199–231.

Breiman, L. and A. Cutler (2003). Setting up, using, and understanding random forests v4. 0. *University of California, Department of Statistics*.

Breiman, L., J. Friedman, C. J. Stone, and R. A. Olshen (1984). *Classification and regression trees*. CRC press.

Buja, A., T. Hastie, and R. Tibshirani (1989). Linear smoothers and additive models. *The Annals of Statistics*, 453–510.

Bureau, A., J. Dupuis, K. Falls, K. L. Lunetta, B. Hayward, T. P. Keith, and P. Van Eerdewegh (2005). Identifying snps predictive of phenotype using random forests. *Genetic epidemiology 28*(2), 171–182.

Chaudhuri, P., M.-C. Huang, W.-Y. Loh, and R. Yao (1994). Piecewise-polynomial regression trees. *Statistica Sinica*, 143–167.

Chipman, H. A., E. I. George, R. E. McCulloch, et al. (2010). Bart: Bayesian additive regression trees. *The Annals of Applied Statistics 4*(1), 266–298.

Cox, D. R. and N. Wermuth (1992). Response models for mixed binary and quantitative variables. *Biometrika 79*(3), 441–461.

Crichton, N. J., J. P. Hinde, and J. Marchini (1997). Models for diagnosing chest pain: is cart helpful? *Statistics in medicine 16*(7), 717–727.

Díaz-Uriarte, R. and S. A. De Andres (2006). Gene selection and classification of microarray data using random forest. *BMC bioinformatics 7*(1), 3.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning 40*(2), 139–157.

Doove, L. L., E. Dusseldorp, K. Van Deun, and I. Van Mechelen (2014). A comparison of five recursive partitioning methods to find person subgroups involved in meaningful treatment–subgroup interactions. *Advances in Data Analysis and Classification 8*(4), 403–425.

Doshi-Velez, F. and B. Kim (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Dusseldorp, E., C. Conversano, and B. J. Van Os (2010). Combining an additive and tree-based regression model simultaneously: Stima. *Journal of Computational and Graphical Statistics 19*(3), 514–530.

Dusseldorp, E. and J. J. Meulman (2004). The regression trunk approach to discover treatment covariate interaction. *Psychometrika 69*(3), 355–374.

Fan, G. and J. B. Gray (2005). Regression tree analysis using target. *Journal of Computational and Graphical Statistics 14*(1), 206–218.

Fan, G.-Z., S. E. Ong, and H. C. Koh (2006). Determinants of house price: A decision tree approach. *Urban Studies 43*(12), 2301–2315.

Fisher, R. A. (1935). The design of experiments. *Oxford, England: Oliver & Boyd*.

Freund, Y. (2001). An adaptive version of the boost by majority algorithm. *Machine learning 43*(3), 293–318.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The annals of statistics*, 1–67.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis 38*(4), 367–378.

Gelman, A., J. B. Carlin, H. S. Stern, and D. B. Rubin (1995). *Bayesian data analysis*. Chapman and Hall/CRC.

Geman, S. and D. Geman (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence* (6), 721–741.

Geurts, P., D. Ernst, and L. Wehenkel (2006). Extremely randomized trees. *Machine learning 63*(1), 3–42.

Gray, J. B. and G. Fan (2008). Classification tree analysis using target. *Computational Statistics & Data Analysis 52*(3), 1362–1372.

Grubinger, T., A. Zeileis, and K.-P. Pfeiffer (2011). evtree: Evolutionary learning of globally optimal classification and regression trees in r. Technical report, Working Papers in Economics and Statistics.

Guidotti, R., A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi (2018). A survey of methods for explaining black box models. *ACM Computing Surveys (CSUR) 51*(5), 93.

Hannah, L. A. and D. B. Dunson (2013). Multivariate convex regression with adaptive partitioning. *The Journal of Machine Learning Research 14*(1), 3261–3294.

Härdle, W. K., M. Müller, S. Sperlich, and A. Werwatz (2012). *Nonparametric and semiparametric models*. Springer Science & Business Media.

Hastie, T., R. Tibshirani, et al. (2000). Bayesian backfitting (with comments and a rejoinder by the authors. *Statistical Science 15*(3), 196–223.

Hastie, T., R. Tibshirani, and J. Friedman (2009). The elements of statistical learning: data mining, inference, and prediction.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence 20*(8), 832–844.

Hothorn, T., K. Hornik, and A. Zeileis (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical statistics 15*(3), 651–674.

Ishwaran, H. and U. B. Kogalur (2010). Consistency of random survival forests. *Statistics & probability letters 80*(13-14), 1056–1064.

Kapelner, A. and J. Bleich (2016). bartmachine: Machine learning with bayesian additive regression trees. *Journal of Statistical Software 70*(4).

Karaolis, M. A., J. A. Moutiris, D. Hadjipanayi, and C. S. Pattichis (2010). Assessment of the risk factors of coronary heart events based on data mining with decision trees. *IEEE Transactions on information technology in biomedicine 14*(3), 559–566.

Kass, G. V. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied statistics*, 119–127.

Kim, H. and W.-Y. Loh (2003). Classification trees with bivariate linear discriminant node models. *Journal of Computational and Graphical Statistics 12*(3), 512–530.

Lemon, S. C., J. Roy, M. A. Clark, P. D. Friedmann, and W. Rakowski (2003). Classification and regression tree analysis in public health: methodological review and comparison with logistic regression. *Annals of behavioral medicine 26*(3), 172–181.

Liu, Y., J. L. Zayas-Castro, P. Fabri, and S. Huang (2014). Learning high-dimensional networks with nonlinear interactions by a novel tree-embedded graphical model. *Pattern Recognition Letters 49*, 207–213.

Loh, W.-Y. (2002). Regression tress with unbiased variable selection and interaction detection. *Statistica Sinica*, 361–386.

Loh, W.-Y. and Y.-S. Shih (1997). Split selection methods for classification trees. *Statistica sinica*, 815–840.

Loh, W.-Y. and N. Vanichsetakul (1988). Tree-structured classification via generalized discriminant analysis. *Journal of the American Statistical Association 83*(403), 715–725.

Lunetta, K. L., L. B. Hayward, J. Segal, and P. Van Eerdewegh (2004). Screening large-scale association study data: exploiting interactions using random forests. *BMC genetics 5*(1), 32.

Ma, Q., D. Wyszynski, J. Farrell, A. Kutlar, L. Farrer, C. Baldwin, and M. Steinberg (2007). Fetal hemoglobin in sickle cell anemia: genetic determinants of response to hydroxyurea. *The pharmacogenomics journal 7*(6), 386.

Mammen, E., O. Linton, J. Nielsen, et al. (1999). The existence and asymptotic properties of a backfitting projection algorithm under weak conditions. *The Annals of Statistics 27*(5), 1443–1490.

Menze, B. H., B. M. Kelm, D. N. Splitthoff, U. Koethe, and F. A. Hamprecht (2011). On oblique random forests. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 453–469. Springer.

Morgan, J. N. and R. C. Messenger (1973). Thaid, a sequential analysis program for the analysis of nominal scale dependent variables. *Ann Arbor: Survey Research Center, Institute for Social Research, University of Michigan.*

Morgan, J. N. and J. A. Sonquist (1963). Problems in the analysis of survey data, and a proposal. *Journal of the American statistical association 58*(302), 415–434.

Murthy, S. K., S. Kasif, and S. Salzberg (1994). A system for induction of oblique decision trees. *Journal of artificial intelligence research 2*, 1–32.

Opsomer, J. D. and D. Ruppert (1999). A root-n consistent backfitting estimator for semiparametric additive modeling. *Journal of Computational and Graphical Statistics 8*(4), 715–732.

Petersen, A., N. Simon, and D. Witten (2016). Convex regression with interpretable sharp partitions. *The Journal of Machine Learning Research 17*(1), 3240–3270.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning 1*(1), 81–106.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

Rashmi, K. and R. Gilad-Bachrach (2015). Dart: Dropouts meet multiple additive regression trees. In *International Conference on Artificial Intelligence and Statistics*, pp. 489–497.

Scornet, E. (2016). Random forests and kernel methods. *IEEE Transactions on Information Theory 62*(3), 1485–1500.

Scornet, E., G. Biau, J.-P. Vert, et al. (2015). Consistency of random forests. *The Annals of Statistics 43*(4), 1716–1741.

Shmueli, G. et al. (2010). To explain or to predict? *Statistical science 25*(3), 289–310.

Strasser, H. and C. Weber (1999). On the asymptotic theory of permutation statistics.

Strobl, C., A.-L. Boulesteix, A. Zeileis, and T. Hothorn (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC bioinformatics 8*(1), 25.

Sutton, R. S., A. G. Barto, et al. (1998). *Reinforcement learning: An introduction*. MIT press.

Therneau, T. (2018). User written splitting functions for rpart. *Mayo Clinic. Disponível*.

Tukey, J. W. (1947). Non-parametric estimation ii. statistically equivalent blocks and tolerance regions–the continuous case. *The Annals of Mathematical Statistics*, 529–539.

Valera, V. A., B. A. Walter, N. Yokoyama, Y. Koyama, T. Iiai, H. Okamoto, and K. Hatakeyama (2007). Prognostic groups in colorectal carcinoma patients based on tumor cell proliferation and classification and regression tree (cart) survival analysis. *Annals of surgical oncology 14*(1), 34–40.

Vellido, A., J. D. Martín-Guerrero, and P. J. Lisboa (2012). Making machine learning models interpretable. In *ESANN*, Volume 12, pp. 163–172. Citeseer.

Wager, S. and S. Athey (2018). Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association 113*(523), 1228–1242.

Wager, S., S. Wang, and P. S. Liang (2013). Dropout training as adaptive regularization. In *Advances in neural information processing systems*, pp. 351–359.

Wasserman, L. (2006). *All of Nonparametric Statistics (Springer Texts in Statistics)*. Berlin, Heidelberg: Springer-Verlag.

White, A. P. and W. Z. Liu (1994). Bias in information-based measures in decision tree induction. *Machine Learning 15*(3), 321–329.

Wickramarachchi, D., B. Robertson, M. Reale, C. Price, and J. Brown (2016). Hhcart: An oblique decision tree. *Computational Statistics & Data Analysis 96*, 12–23.

Wolfson, J. and A. Venkatasubramaniam (2018). Branching out: Use of decision trees in epidemiology. *Current Epidemiology Reports 5*(3), 221–229.

Zhu, R., D. Zeng, and M. R. Kosorok (2015). Reinforcement learning trees. *Journal of the American Statistical Association 110*(512), 1770–1784.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical association 101*(476), 1418–1429.