Journal of Machine Learning Research 23 (2022) 1-31 Submitted 8/21; Revised 2/22; Published 6/22

Tree-Based Models for Correlated Data

Assaf Rabinowicz ASSAFRAB@GMAIL.COM
Department of Statistics and Operations Research
Tel Aviv University
Tel Aviv, Israel

Saharon Rosset SAHARON@TAUEX.TAU.AC.IL
Department of Statistics and Operations Research
Tel Aviv University
Tel Aviv, Israel

Editor: Maya Gupta

Abstract

This paper presents a new approach for regression tree-based models, such as simple regression tree, random forest and gradient boosting, in settings involving correlated data. We show the problems that arise when implementing standard regression tree-based models, which ignore the correlation structure. Our new approach explicitly takes the correlation structure into account in the splitting criterion, stopping rules and fitted values in the leaves, which induces some major modifications of standard methodology. The superiority of our new approach over tree-based models that do not account for the correlation, and over previous work that integrated some aspects of our approach, is supported by simulation experiments and real data analyses.

Keywords: random forest, linear mixed models, Gaussian process regression, prediction error for correlated data, model selection

1. Introduction

Tree-based models are widely used for tabular data due to their high prediction accuracy and their inherent model selection functionality (Hastie et al., 2009). Commonly, tree-based models are fitted without assuming any distributional setting on the dependent variable. While the distribution of the dependent variable is mostly unknown and therefore it is tempting to avoid distributional assumptions, the correlation structure, which relates to the sampling mechanism (e.g., clustered data, time-series data, longitudinal data, spatial data), is frequently known and therefore it is not reasonable to ignore it. Unlike in tree-based models, the correlation structure is an essential component in many machine learning models, for example, kernel covariance functions are used in Gaussian processes regression (Rasmussen, 2003), which is frequently implemented for modeling data sets with spatial correlation structure, such as neuroscience data sets (Caywood et al., 2017) and climatological data sets (Goovaerts, 1999). Another example is linear mixed model, which is used for data sets involving clusters and longitudinal correlation structures, as is common in health (Coull et al., 2001) and trading (Westveld and Hoff, 2011) applications.

In this paper we develop a method which combines the concepts of *random effects* and *random fields*, which are convenient platforms for analyzing correlated data, and tree-based models such as: regression tree, random forest and gradient boosting. The desired outcome is that the tree-based part results a high prediction accuracy and model selection capabilities, and the random effects part enables to boost the model performance by using the correlation structure, as well as to provide statistical inference. The idea of integrating between random effects/random fields and tree-based methods has previously been explored (for example, see Hajjem et al. 2011; Sela and Simonoff 2012; Stephan et al. 2015), and previous and parallel work in this area is discussed in detail in Section 3.1. However, we propose a novel approach which takes advantage of recent developments in model evaluation and selection methodologies for correlated settings, and yields improved results as demonstrated below.

Section 2 gives relevant background for the proposed method. The background contains a brief description of tree-based methods, linear mixed model (which is based on random effects) and prediction error estimation for correlated data, which has a key role in our approach. Section 3 presents previous and parallel work, and positions our approach within this context. Section 4 introduces our new approuch, REgression Tree for COrrelated data (RETCO), in detail and illustrates its central properties. Section 5 presents simulations and real data analyses.

2. Theoretical Background

This section briefly presents regression tree-based models, linear mixed models and prediction error estimation for correlated data. Additional information can be found in Appendix A and in resources which are cited below.

2.1 Tree-based Models

Given a vector of covariates $\boldsymbol{x}^* \in \mathbb{R}^p$, a regression tree estimates the corresponding response, $y^* \in \mathbb{R}$, as follows:

$$f(\boldsymbol{x}^*) = \sum_{s=1}^{S} I_{(\boldsymbol{x}^* \in g_s)} \mu_s,$$

where $g_s \subseteq \mathbb{R}^p$ and $\mu_s \in \mathbb{R}$, for all $s \in \{1, ..., S\}$. The regions $\{g_s\}_{s=1}^S$ define a partition of the covariate space, that is $g_s \cap g_t = \emptyset$ for $s \neq t$, and $\cup_{s=1}^S g_s$ is the entire covariates space. The scalar μ_s is the predictor for observations whose covariate vectors are in g_s. The set $\mathcal{S} = \{\mu_s, g_s\}_{s=1}^S$, is selected using a recursive optimization process that can be diagrammed as a tree, where $\mathcal{S}$ are the terminal nodes (the leaves). The recursive optimization for selecting $\mathcal{S}$ is based on the training set, $\{y_i, \boldsymbol{x}_i\}_{i=1}^n = \{\boldsymbol{y}, X\}$, where $\boldsymbol{y} \in \mathbb{R}^n$ and $X \in \mathbb{R}^{n \times p}$ is the design matrix, and it is implicitly assumed that the prediction points, y^* and $\boldsymbol{x}^*$, are drawn from the same distribution as y_i and $\boldsymbol{x}_i$. Each step in the tree's recursive optimization process is a model selection of linear models, where a threshold of one of the available covariates is selected in order to minimize a loss function, $Loss(\cdot, \cdot) : \mathbb{R}^2 \to \mathbb{R}$. Also, stopping rules that follow predefined hyper-parameters, e.g., maximal depth of the tree or minimal number of training set observations in a node, are enforced on the recursive optimization. These limit the tree's complexity and affect its statistical properties, in particular limit overfitting.

The widely used Random forest (RF) and gradient boosting (GB) predictive modeling approaches are ensemble methods that combine trees to create powerful and accurate prediction models. More information about regression tree, RF and GB, as well a formalized regression tree algorithm, is available in Appendix A, and can also be found in Freund et al. (1999); Friedman (2001); Breiman (2001); Hastie et al. (2009).

2.2 Linear Mixed Model

In linear mixed models (LMM) there are two covariate vectors: fixed effects covariates, $\boldsymbol{x}^* \in \mathbb{R}^p$, and random effects covariates, $\boldsymbol{z}^* \in \mathbb{R}^q$. Commonly, y^* is assumed to be normally distributed and decomposed as follows:

$$y^* = \boldsymbol{\beta}^t \boldsymbol{x}^* + \boldsymbol{b}^t \boldsymbol{z}^* + \epsilon^*,$$

where $\boldsymbol{\beta}$ is the fixed effects vector of coefficients, $\boldsymbol{b} \sim N_q(0, G)$ is the random effects vector, and $\epsilon^* \sim N(0, \sigma^2)$ is the residual. In addition, $\boldsymbol{\beta}^t \boldsymbol{x}^*$ is the *marginal mean* of y^*, $\mathbb{E}(y^*|\boldsymbol{x}^*)$, and $\boldsymbol{\beta}^t \boldsymbol{x}^* + \boldsymbol{b}^t \boldsymbol{z}^*$ is the *conditional mean* of y^* given $\boldsymbol{b}$, which is commonly denoted as $\mathbb{E}(y^*|\boldsymbol{x}^*, \boldsymbol{z}^*, \boldsymbol{b})$.

The vectors $\boldsymbol{\beta}$ and $\boldsymbol{b}$ are estimated using the training sample, $\{y_i, \boldsymbol{x}_i, \boldsymbol{z}_i\}_{i=1}^n = \{\boldsymbol{y}, X, Z\}$, which follows the same model:

$$\boldsymbol{y} = X\boldsymbol{\beta} + Z\boldsymbol{b} + \boldsymbol{\epsilon},$$

where $\boldsymbol{\epsilon} \sim N_n(0, \sigma^2 I_n)$, and X, Z are the fixed effects and random effects covariate matrices, respectively. Since y^* and $\boldsymbol{y}$ share the same random effects, $\boldsymbol{b}$, they are correlated, and estimating $\boldsymbol{b}$ by the training sample can later be used for improving the prediction accuracy of y^*. Given G and $V \coloneqq \mathrm{Var}(\boldsymbol{y}) = ZGZ^t + \sigma^2 I_n$, $\boldsymbol{\beta}$ and $\boldsymbol{b}$ can be estimated as follows:

$$\widehat{\boldsymbol{\beta}} = (X^t V^{-1} X)^{-1} X^t V^{-1} \boldsymbol{y}$$
$$\widehat{\boldsymbol{b}} = GZV^{-1}(\boldsymbol{y} - X\widehat{\boldsymbol{\beta}}).$$

Harville (1976) showed that given the true covariance matrices, G and V, the estimated *conditional mean*,

$$\widehat{\mathbb{E}}(y^*|\boldsymbol{x}^*, \boldsymbol{z}^*, \widehat{\boldsymbol{b}}) = \widehat{\boldsymbol{\beta}}^t \boldsymbol{x}^* + \widehat{\boldsymbol{b}}^t \boldsymbol{z}^*, \tag{1}$$

is the best linear unbiased predictor (BLUP) of y^*. In practice, the covariance matrices are mostly unknown and therefore are estimated using maximum likelihood or restricted maximum likelihood (Verbeke, 1997). Note that given the covariance matrices, LMM is linear in $\boldsymbol{y}$, i.e., the LMM predictor satisfies:

$$\widehat{y}^* = \boldsymbol{h}^{*t} \boldsymbol{y},$$

where $\boldsymbol{h}^*$, the hat vector, does not contain the training response vector $\boldsymbol{y}$, and the element $h^*[i]$ is the *weight* of y_i in predicting $y^*, \forall i \in [1, ..., n]$.

2.2.1 $\boldsymbol{b}^* \neq \boldsymbol{b}$ Scenario

In many cases the random effects of y^* are not the same as the random effects of $\boldsymbol{y}$, i.e.,

$$y^* = \boldsymbol{\beta}^t \boldsymbol{x}^* + \boldsymbol{b}^{*t} \boldsymbol{z}^* + \epsilon^*, \;\; \boldsymbol{b}^* \neq \boldsymbol{b}.$$

This means that the correlation between the observations in $\boldsymbol{y}$ is not the same as the correlation between y^* and the observations in $\boldsymbol{y}$. In case $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$, which implies $\mathrm{Cov}(y^*, \boldsymbol{y}) = 0$, estimating $\boldsymbol{b}$ does not improve the prediction accuracy of y^*. Therefore, in this case, y^* is predicted by the *marginal mean* , $\widehat{\mathbb{E}}(y^*|\boldsymbol{x}^*) = \widehat{\boldsymbol{\beta}}^t \boldsymbol{x}^*$. This predictor, which is a special case of the BLUP where $\widehat{\boldsymbol{b}} = 0$, is also called generalized least squares predictor (GLS). A simple example for this scenario, is when the prediction set contains different clusters than in the training set. Several prediction tasks that follow this scenario setting are analyzed in Section 5.2. For example, using the FIFA data set from Kaggle website, a predictive model for football players' market-value was trained, where the prediction goal is to predict the market-value of players that belong to clubs that *do not appear* in the training set. This data set has a clustered correlation structure, where cluster is the player's club, i.e., market-values of players that belong to the same club are correlated, while market-values of players that belong to different clubs are uncorrelated. Given the prediction goal of predicting the market-value of players from new clubs, this setting follows exactly the $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$ setting.

Another scenario is when $\boldsymbol{b}^* \not\perp\!\!\!\perp \boldsymbol{b}$ (although $\boldsymbol{b}^* \neq \boldsymbol{b}$) and therefore $\mathrm{Cov}(y^*, \boldsymbol{y}) \neq 0$. This can happen for example when some of the random effects of $\boldsymbol{y}$ and y^* are the same and some are not. In this case, the elements in $\boldsymbol{b}$ that are in $\boldsymbol{b}^*$ are estimated and used for predicting y^*. This scenario of $\boldsymbol{b}^* \neq \boldsymbol{b}$, is common and should be taken into account when developing tree-based methods for correlated data. Our proposed tree-based model covers this scenario, which is demonstrated and analyzed in Section 5.2 with a real data set — California Housing data set.

2.3 Prediction Error Estimation and Model Selection for Correlated Data

Once a predictive model is fitted, it is often evaluated by its prediction error estimator. Moreover, when there is a set of alternative models (e.g., for LMM: models with different covariates, for tree-based models: models with different hyper-parameters) the 'best' model can be selected based on minimizing the prediction error estimator. It is important to note that common prediction error estimators, e.g., AIC (Akaike, 1974), Cp (Mallows, 1973), and even cross-validation (Stone, 1974, CV) are biased in some settings involving correlated data. Naturally, their corresponding model selection criteria are also suboptimal in those scenarios. This bias was studied in the recent years, mostly for linear models. Here we present Cp, AIC and CV versions for correlated data. For description of the original Cp and AIC versions, that do not address correlation structure, see Appendix A.

2.3.1 Cp

In Cp, the goal is to estimate the squared prediction error:

$$\mathbb{E}_{\boldsymbol{y},\boldsymbol{y}^*} \frac{1}{n} \|\boldsymbol{y}^* - H\boldsymbol{y}\|_2^2,$$

where H is the hat matrix, and $\boldsymbol{y}^* \in \mathbb{R}^n$ is a vector of new observations measured at the same covariate values as $\boldsymbol{y}$, $\{X, Z\}$, but with new independent noise and potentially different random effects realizations. This type of prediction error, when both $\boldsymbol{y}$ and $\boldsymbol{y}^*$ relate to the same covariate points, $\{X, Z\}$, is called *in-sample* prediction error. In this setting, it is natural to consider $\{X, Z\}$ as fixed matrices rather than random variables. Hodges and Sargent (2001) extended Cp to LMM with $\boldsymbol{b}^* = \boldsymbol{b}$, here we employ a more general formulation which reduces to Hodges and Sargent (2001) when $\boldsymbol{b}^* = \boldsymbol{b}$, but also covers the case that they are different:

$$Cp = \frac{1}{n}\|\boldsymbol{y} - \widehat{\boldsymbol{y}}\|_2^2 + \frac{2}{n}\mathrm{tr}\Big(H\big(\mathrm{Var}(\boldsymbol{y}) - \mathrm{Cov}(\boldsymbol{y}^*, \boldsymbol{y})\big)\Big).$$

2.3.2 AIC

Similarly to Cp, AIC is also an in-sample error, however its loss function is based on the likelihood function. Vaida and Blanchard (2005) presented the conditional AIC (cAIC) and marginal AIC (mAIC) which are suitable for the scenarios where $\boldsymbol{b}^* = \boldsymbol{b}$ and $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$, respectively. Here we will use the name AIC for our formulation which subsumes cAIC and mAIC, but also covers the $(\boldsymbol{b}^* \neq \boldsymbol{b}) \cap (\boldsymbol{b}^* \not\perp\!\!\!\perp \boldsymbol{b})$ scenario:

$$AIC = -\frac{2\ell\big(\boldsymbol{y}; \widehat{\mathbb{E}}(\boldsymbol{y}|X, Z, \widehat{\boldsymbol{b}}), V_c\big)}{n} + \frac{2\mathrm{tr}\Big(H\big(\mathrm{Var}(\boldsymbol{y}) - \mathrm{Cov}(\boldsymbol{y}, \boldsymbol{y}^*)\big)V_c^{-1}\Big)}{n},$$

where $\ell(\boldsymbol{y}; \widehat{\mathbb{E}}(\boldsymbol{y}|X, Z, \widehat{\boldsymbol{b}}), V_c)$ is the conditional log-likelihood of $\boldsymbol{y}$ given $\widehat{\boldsymbol{b}}$, and $V_c = \mathrm{Var}(\boldsymbol{y}^*|\boldsymbol{b})$.

2.3.3 Cross-Validation (CV)

Unlike Cp and AIC, CV estimates the *generalization error*:

$$\mathbb{E}_{X,X^*,Z,Z^*}\mathbb{E}_{\boldsymbol{y}^*,\boldsymbol{y}} \sum_{i=1}^{n} \frac{1}{n} Loss\big(y_i^*, \widehat{y}_i^*(\boldsymbol{x}_i^*, \boldsymbol{z}_i^*; \boldsymbol{y}, X, Z)\big),$$

where $\boldsymbol{y}^*$ is drawn from the same marginal distribution as $\boldsymbol{y}$, but relates to *new covariate values*, $\{\boldsymbol{x}_i^*, \boldsymbol{z}_i^*\}_{i=1}^n = \{X^*, Z^*\}$, which were drawn from the same distribution as $\{X, Z\}$. For simplicity, a special case of CV algorithm, leave-one-out (LOO), is presented:

1. For each $i \in [1, ..., n]$, fit a model using the whole sample except the i^{th} observation. Denote the sample without the i^{th} observation by $\{\boldsymbol{y}_{-i}, X_{-i}, Z_{-i}\}$.
2. Predict y_i from the fitted model and denote the predictor by $\widehat{y}_i^{-i} = \widehat{y}_i(\boldsymbol{x}_i, \boldsymbol{z}_i; \boldsymbol{y}_{-i}, X_{-i}, Z_{-i})$.

For a squared error loss function and linear predictor of $\boldsymbol{y}$, the CV error is

$$CV = \frac{1}{n}\|\boldsymbol{y} - H_{cv}\boldsymbol{y}\|_2^2,$$

where H_{cv}, the CV hat matrix, is

$$H_{cv} = \begin{bmatrix} 0 & h_{1,2} & ... & h_{1,n} \\ h_{2,1} & 0 & & h_{2,n} \\ ... & & & \\ h_{n,1} & h_{n,2} & ... & 0 \end{bmatrix},$$

Method	Prediction Error Type	Distributional Assumptions
Cp	in-sample error	–
AIC	in-sample error	normal likelihood
CV_c	generalization error	–

Table 1: Summary of prediction errors for correlated data

$h_{k,k'} \in \mathbb{R}\ \forall k, k' \in \{1, ..., n\}$. In this presentation, the vector $[h_{1,2}, ..., h_{1,n}]$ is the hat vector of $\widehat{y_1^{-1}}$. K-fold CV generalizes LOO by partitioning $\{\boldsymbol{y}, X, Z\}$ into K equal size subsets, $\{\boldsymbol{y}_k, X_k, Z_k\}_{k=1}^K$, where $K \leq n$ ($K = n$ in LOO).

Rabinowicz and Rosset (2022) presented a generalization of CV, CV_c, which is suitable for scenarios involving correlated data:

$$CV_c = CV + \frac{2}{n}\mathrm{tr}\Big(H_{cv}\big(\mathrm{Var}(\boldsymbol{y}) - \mathrm{Cov}(\boldsymbol{y}^*, \boldsymbol{y})\big)\Big).$$

Note that when $\boldsymbol{b}^* = \boldsymbol{b}$, then $CV_c = CV$. For more information see Rabinowicz and Rosset (2022).

Cp and CV_c do not assume a specific distributional setting and can be applied for any linear model, while AIC is suitable for LMM, however it can also be adjusted for other linear models that assume normality, such as Gaussian process regression (GPR). Table 1 summarizes the prediction error estimators that are described in this section.

3. Literature Review and RETCO Approach

Section 3.1 presents a general model that combines regression tree-based models and LMM, as well as a literature review of previous and parallel work that refers to this model. Section 3.2 presents the main principles that RETCO is based on, and relates them to the previous work. RETCO is presented in detail in Section 4.

3.1 Previous and Parallel Work

A simple approach for integrating between tree-based methods and random effects is replacing the marginal mean in LMM, $\boldsymbol{\beta}^t\boldsymbol{x}^*$, by a tree-based model, $f(\boldsymbol{x}^*)$:

$$f(\boldsymbol{x}^*) + \boldsymbol{b}^t\boldsymbol{z}^*, \tag{2}$$

where $f(\boldsymbol{x}^*)$ is created in a way that does account for the correlation structure (unlike in the standard regression tree algorithm). The power of the model in expression (2) can be perceived from different points of views. From the LMM point of view, the additive representation of marginal and conditional means is preserved, however the marginal mean is more expressive than in standard LMM. From the regression tree point of view, this approach differentiates between the two types of covariates—fixed effects, which are used for splitting the tree's nodes, and random effects that are added linearly to the fitted tree—enabling expressing and using the correlation structure. This also enables using inference tools that do not exist for standard regression trees but exists in LMM, for example, comparing between

the variance components, σ^2 and G. Note that expression (2) assumes that the effect of the random effects on $\boldsymbol{y}$ is linear, however it can be generalized.

To our knowledge, the integration between LMM and regression tree was first proposed by Hajjem et al. (2011). A similar algorithm, RE-EM, was proposed independently by Sela and Simonoff (2012). The main idea in RE-EM is fitting $f(\cdot)$ to $\boldsymbol{y} - Z\widehat{\boldsymbol{b}}$, then, given the fitted $f(\cdot)$, calculating $\widehat{\boldsymbol{b}}$ using the standard BLUP formula. This process is done iteratively until convergence of $\widehat{\boldsymbol{b}}$. The fitting of $f(\cdot)$ is a two stages process, where first $\{g_s\}_{s=1}^{\mathcal{S}}$ are generated using a standard regression tree, then, given $\{g_s\}_{s=1}^{S}$, $\{\mu_s\}_{s\in\mathcal{S}}$ are estimated together by GLS (using the whole sample). Therefore, RE-EM takes the correlation into account when estimating $\{\mu_s\}_{s\in\mathcal{S}}$, but does not account for the correlation directly when selecting $\{g_s\}_{s=1}^{S}$ (it does account for the correlation indirectly since the tree is fitted to $\boldsymbol{y} - Z\widehat{\boldsymbol{b}}$). The RE-EM algorithm, which is presented in more detail in Appendix C, was extended several times, for example Fu and Simonoff (2015) suggested using conditional inference trees, Hajjem et al. (2014) and Capitaine et al. (2021) proposed RF algorithms that are based on the same logic as RE-EM algorithm.

Stephan et al. (2015) proposed the Mixed Random Forest (MRF). The goal in MRF is fitting a tree-based model that estimates accurately the variance components, rather than optimizing prediction accuracy. Also, MRF assumes a specific data type and is based on relatively strong distributional assumptions ($\boldsymbol{y}$ is normally distributed, and $G = \sigma_b^2 I_q$). Still, the approach in MRF is very relevant for this paper: selecting $\{g_s, \mu_s\}_{s=1}^{S}$ by maximizing the marginal likelihood of $\boldsymbol{y}$ at each split. In this way, MRF takes into account the correlation in selecting both $\{g_s\}_{s=1}^{S}$ and $\{\mu_s\}_{s\in\mathcal{S}}$. The MRF algorithm is presented in more detail in Appendix C. Recently, Saha et al. (2021) proposed a similar RF algorithm for spatial data, as well as presented various asymptotic properties of their method.

Extensions of these papers, where the response is binary or count data, as in generalized linear mixed model (Wolfinger and O'connell, 1993), were proposed by Fokkema et al. (2018); Hengl et al. (2018); Ngufor et al. (2019); Speiser et al. (2019); Pellagatti et al. (2021). Next, our proposed algorithm is presented and analyzed. Detailed methodological and numerical comparisons between our proposed algorithms and relevant previous algorithms are presented in Section 4.2 and Section 5.

3.2 RETCO Approach

RETCO, our new algorithm, proposes a different approach for fitting $f(\boldsymbol{x}^*)$ than the previous algorithms. This approach is based on the following points:

1. Distinguishing between the $\boldsymbol{b}^* = \boldsymbol{b}$ and $\boldsymbol{b}^* \neq \boldsymbol{b}$ scenarios in the splitting criterion. As will be presented in Section 4.2, this distinction, which is fundamental in mixed models, is crucial here.

2. Using *prediction error for correlated data* loss function in the splitting criterion, instead of training error loss function as in other algorithms. Using prediction error for correlated data loss function is essential in order to properly address point 1, and has many implications on the structure of the RETCO algorithm.

3. Adopting a global whole-sample point of view for the split selection, rather than the node-local point of view (as implemented in the standard tree-based models). Unlike in the i.i.d. setting, where it is reasonable to split each node independently of the other nodes, fitting trees to correlated data requires taking into account the whole sample when splitting a node due to the correlation between observations from different nodes. As presented in Section 3.1, the previous algorithms also include some global components, however RETCO goes a few steps further.

As presented in the next section, these principles derive the structure of RETCO.

4. RETCO Algorithm

This section introduces and analyzes our new proposed algorithm, RETCO— REgression Tree for COrrelated data.

4.1 Main Algorithm

RETCO formulation is general and is not based on a specific prediction error type or distributional setting. Also, the basic algorithm refers to a case when $f(\cdot)$ is a single regression tree, and the extensions to RF and GB will be discussed in Section 4.1.1.

Algorithm 1, which presents RETCO, uses the following notations:

- $f(\boldsymbol{x}_i|\mathcal{S})$ — the intermediate predictor of y_i during the tree fitting (for a "current" tree with the nodes $\mathcal{S}$).
$$f(\boldsymbol{x}_i|\mathcal{S}) = \sum_{s\in\mathcal{S}} I_{(\boldsymbol{x}_i\in g_s)}\eta(\mu_s),$$
where
  - μ_s is the GLS predictor of $\{y_i|\boldsymbol{x}_i \in g_s\},\ \forall s \in \mathcal{S}$.
  - $\eta(\mu_s)$ is the BLUP function for leaf s. Note, when $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$, then $\eta(\mu_s) = \mu_s$.

  After fitting the final tree $f(\boldsymbol{x}_i) = f(\boldsymbol{x}_i|\mathcal{S})$.

- $f\left(\boldsymbol{x}_i|j,c,\mathcal{S}/s\right)$, the predictor of y_i when splitting node s using covariate j at the threshold c for current tree with the nodes $\mathcal{S}$:
$$f\left(\boldsymbol{x}_i|j,c,\mathcal{S}/s\right) = I_{(\boldsymbol{x}_i\in g_s\ \cap\ x_{i,j}\le c)}\eta\big(\mu_s^l(c)\big)+I_{(\boldsymbol{x}_i\in g_s\ \cap\ x_{i,j}>c)}\eta\big(\mu_s^r(c)\big)+\sum_{m\in\mathcal{S}/s} I_{(\boldsymbol{x}_i\in g_m)}\eta(\mu_m),$$
where $\mu_s^l(c)$ and $\mu_s^r(c)$ are the GLS predictors of $\{y_i|\boldsymbol{x}_i \in g_s\ \cap\ x_{i,j} \le c\}$ and $\{y_i|\boldsymbol{x}_i \in g_s\ \cap\ x_{i,j} > c\}$, respectively.

Algorithm 1 REgression Tree for COrrelated Data (RETCO)

Input: $\boldsymbol{y}$, X, Z.
Output: $f(\cdot)$.
General setting:

- select $Loss(\cdot,\cdot)$: a prediction error estimator loss function—Cp, AIC or CV_c—from Table 1,
- define the relation between $\boldsymbol{b}^*$ and $\boldsymbol{b}$ ($\boldsymbol{b}^* = \boldsymbol{b}$ versus $\boldsymbol{b}^* \neq \boldsymbol{b}$),
- define the stopping rules.

Initialization: $\mathcal{S} = \{g_1, \mu_1\}$, where $g_1 = \mathbb{R}^p$ and μ_1 is its GLS predictor.
repeat
Given the predefined stopping rules, find the best node for splitting ($\tilde{s}$), the best covariate ($j_{\tilde{s}}$) and the best threshold ($c_{\tilde{s}}$) as follows:

$$\tilde{s}, j_{\tilde{s}}, c_{\tilde{s}} = \underset{s\in\mathcal{S}, j\in J_s, c\in\mathbb{R}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} Loss\big(y_i, f\left(\boldsymbol{x}_i|j,c,\mathcal{S}/s\right)\big)$$
$$s.t.: \ \frac{1}{n}\sum_{i=1}^{n} Loss\big(y_i, f\left(\boldsymbol{x}_i|j,c,\mathcal{S}/s\right)\big) < \frac{1}{n}\sum_{i=1}^{n} Loss\big(y_i, f\left(\boldsymbol{x}_i|\mathcal{S}\right)\big), \tag{3}$$

where J_s is the set of available covariates for splitting node s.
if $\{\tilde{s}, j_{\tilde{s}}, c_{\tilde{s}}\}$ exist, **then**
Update $\mathcal{S}$: replace $(g_{\tilde{s}}, \mu_{\tilde{s}})$ by the new two nodes, $\big(g_{\tilde{s}} \cap x_{j_{\tilde{s}}} \leq c_{\tilde{s}}, \mu^l_{\tilde{s}}(c_{\tilde{s}})\big)$, $\big(g_{\tilde{s}} \cap x_{j_{\tilde{s}}} > c_{\tilde{s}}, \mu^r_{\tilde{s}}(c_{\tilde{s}})\big)$, where $x_{j_{\tilde{s}}}$ is the covariate $j_{\tilde{s}}$.
end if
until $\{\tilde{s}, j_{\tilde{s}}, c_{\tilde{s}}\}$ do not exist or stopping rules are satisfied $\forall s \in \mathcal{S}$.

As can be seen in Algorithm 1, RETCO covers various settings including different prediction error measures and correlation structures. Before analyzing its properties, here are some technical details for RETCO algorithm:

- Predefined stopping rules: stopping rules, such as maximal tree's depth and minimal number of observations in a node, are commonly applied when fitting regression trees (for more details, see Section 2.1).
- Variance estimation: the variance components—needed for calculating $f\left(\boldsymbol{x}_i|j,c,\mathcal{S}/s\right)$, $f\left(\boldsymbol{x}_i|\mathcal{S}\right)$, and the loss function—can be estimated in different ways, for example using maximum likelihood or restricted maximum likelihood. For clustered data, simple closed-form equations that estimate the variance components are available and presented in Appendix B.
- Main optimization part:

  - the GLS predictors— $\mu_s^l(c)$, $\mu_s^r(c)$ and $\{\mu_m\}_{m\in\mathcal{S}/s}$—are the estimated coefficients of the current leaves: $\{i|\boldsymbol{x}_i \in g_s \cap x_{i,j} \leq c\}$, $\{i|\boldsymbol{x}_i \in g_s \cap x_{i,j} > c\}$ and $\{i|\boldsymbol{x}_i \in g_m\}_{m\in\mathcal{S}/s}$. This uses dummy variables setting, where each covariate is an indicator of a different leaf.
  - for every potential split, the variance components are estimated using the whole sample (rather than only $\{y_i|\boldsymbol{x}_i \in g_s\}$),
  - the condition of inequality (3) is required since the loss function is not a training error (as in the standard regression tree algorithm) and therefore splitting a node may increase the loss.

- Algorithm's Output: since the tree estimates the marginal mean, its predictors are $\{\mu_l\}_{l\in\mathcal{S}}$ rather than $\{\eta(\mu_l)\}_{l\in\mathcal{S}}$.

As was mentioned above, using prediction error estimator in the splitting criterion enables us to address different prediction goals, $\boldsymbol{b}^* = \boldsymbol{b}$ versus $\boldsymbol{b}^* \neq \boldsymbol{b}$. Although the regression tree model is a non-linear function of $\boldsymbol{y}$, each split is a model selection problem of linear models and therefore Cp, AIC and CV_c can be implemented. The importance of using prediction error instead of training error is illustrated in Section 4.2 and Section 5. The other key principle, of taking into account the whole sample when splitting the nodes in order to account for the correlation between observations from different paths, is addressed in several points in RETCO. First, the input at each split is the whole sample, and observations from different paths interact through various components in the loss function — GLS predictors, the condition of inequality (3), and the bias correction and likelihood function (depending on the correlation setting and the selected loss function). Second, an *iterative* approach is proposed here instead of the recursive approach that is used in the standard regression tree model. The iterative approach is expressed by selecting the optimal node, $\tilde{s}$, for splitting, rather than splitting each node independently. The reason behind this iterative approach is that splitting one node may affect the splitting of others thorough the loss. This is in contrast to the i.i.d. setting with training error loss function, where a recursive approach can be used since observations in different paths are independent and therefore splitting one node does not affect the splitting of others.

For LMM-like setting, when $\boldsymbol{b}^* \not\perp\!\!\!\perp \boldsymbol{b}$, once $f(\cdot)$ is fitted, the random effects can be estimated in the same way as in the BLUP, formula (1):

$$\widehat{\boldsymbol{b}} = \widehat{\mathrm{Cov}}(\boldsymbol{y}^*, \boldsymbol{y})\widehat{V}^{-1}\big(\boldsymbol{y} - f(X)\big), \tag{4}$$

where $f(X) = [f(\boldsymbol{x}_1), ..., f(\boldsymbol{x}_n)]$.

RETCO is formalized in the context of LMM: $\{\mu_s\}_{s=1}^S$ are GLS predictors and the variance is decomposed as in LMM, $V = ZGZ^t + \sigma^2 \times I_n$. However, these properties are not fundamental in the algorithm and can easily be generalized. For example, the covariance matrices, $\mathrm{Cov}(\boldsymbol{y}^*, \boldsymbol{y})$ and $\mathrm{Var}(\boldsymbol{y})$, can be expressed using a kernel covariance function as is common in Gaussian process regression (as will be demonstrated numerically in Section 5). Also, other linear models instead of GLS can be used for estimating the marginal means, $\{\mu_s\}_{s=1}^S$. Moreover, the random effects, $\boldsymbol{b}$, can be estimated in different ways than in equation (4).

Due to the iterative approach the complexity is higher than the complexity of a standard regression tree. Assuming $x_{i,j} \neq x_{i',j} \, \forall i \neq i' \in [1, ..., n]$ and $j \in [1, ..., p]$, the loss function is evaluated less than $n \times p \times 2^d$ times, where d is the tree's depth. Practically, due to hyper-parameters that restrict the number of potential splitting values (such as minimal number of observations at each node) and due to the constraint in inequality (3), which frequently shortens the depth of paths, the number of evaluations is smaller. The complexity of each loss function evaluation depends on the loss function, the predictor type, and the correlation structure. For example, in Cp loss function, GLS predictor, and clustered data, each evaluation complexity is $O(n_c^3)$, where n_c is the cluster size, and therefore the overall computational complexity is $O\big(n \times p \times 2^d \times n_c^3\big)$. The amount of memory required is quadratic in n due to storing of $\mathrm{Var}(\boldsymbol{y})$.

4.1.1 RETCO for RF and GB

Extending RETCO to RF and GB is done by building ensembles of RETCO trees, while taking into account the special adjustments that RF and GB require. The random effects are estimated in the same way. Several aspects should be noted when implementing RF and GB:

- *Training set sampling*: sampling with replacement cannot be implemented naively when the loss function involves calculation of $\mathrm{Var}(\boldsymbol{y})^{-1}$ (as when GLS predictor or marginal likelihood loss are used) since in that case $\mathrm{Var}(\boldsymbol{y})$ might be a singular matrix due to duplication. In our case half-sampling worked well, but other sub-sampling schemes might be relevant. For more information about sub-sampling in RF see Duroux and Scornet (2018).

- *Number of trees*: since the trees in RF and GB are correlated, in order to reduce the variance the number of trees should be large, especially when the trees are deep. When correlated data are involved, the trees are even more correlated due to the correlation between the observations. Therefore, the number of trees should be even larger than in the i.i.d. sample setting. RF based on RETCO is demonstrated in Section 5.

- *Response in GB*: in common regression GB the trees are fitted consecutively to the residual of the previous trees. Therefore, the input of the algorithm in all the trees except the first one is not $\boldsymbol{y}$. Correspondingly, the estimated variance matrices relate to the residual of the previous trees, rather than to $\boldsymbol{y}$.

4.1.2 Using CV Loss in Regression Tree

In typical predictive modeling settings, generalization error is the primary objective of learning, hence CV loss is the natural choice. Surprisingly, there is not much previous work on using CV loss in trees, even without correlation. Notable exceptions are the ALOOF algorithm (Painsky and Rosset, 2016) and approximations used in CatBoost (Prokhorenkova et al., 2018). The main drawback in using CV-based loss function is increasing the computational cost compared to Cp loss function. Moreover, since RETCO is iterative rather than recursive, the number of evaluations of the loss function can remain large for all the splits along the tree.

4.2 The Bias Correction Effect

As explained in the previous sections, we suggest to add a bias correction term to the training error such that the loss function estimates the prediction error. This section illustrates the effect of the bias correction on split selection. Extensive numerical analysis is presented in Section 5.

4.2.1 $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$ Scenario

Observations with positive correlation are more likely to be similar than uncorrelated observations. Therefore, loss functions that do not take into account the correlation, tend to split the nodes based on the correlation structure of the training set observations rather than their mean. Splitting based on the correlation of the training data is not useful for predicting uncorrelated observations. Therefore, when $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$ it is important to fit the regression tree based on the marginal mean only. The corrections in Cp and CV_c, which take into account the correlation structure, balance this tendency. Examples 4.1 and 4.2 demonstrate this mechanism. The code for the examples, as well as for the numerical part in Section 5, is written in Python and is available in `https://github.com/AssafRab/RETCO`.

Example 4.1 *Consider the setting of $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$ and a training data containing four observations from two clusters with the covariance matrix* $\mathrm{Var}(\boldsymbol{y}) = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix}$*, i.e., observations 1 and 2 belong to the first cluster and observations 3 and 4 belong to the second cluster. The CV_c correction is reduced in this setting to* $2\mathrm{tr}\big(H_{cv}\mathrm{Var}(\boldsymbol{y})\big)/n$*. Two models with the GLS predictor are tested:*

- *Model A, which splits the training set into the two clusters. Given the covariance matrix:*
$$\frac{2}{n}\mathrm{tr}\left[H_{cv}\mathrm{Var}(\boldsymbol{y})\right] = \frac{2}{n}\mathrm{tr}\left[\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix}\right] = 2.$$
- *Model B, that mixes between the clusters and selects observations 1 and 3 for one subset and 2 and 4 for the other subset. Given the covariance matrix:*
$$\frac{2}{n}\mathrm{tr}\left[H_{cv}\mathrm{Var}(\boldsymbol{y})\right] = \frac{2}{n}\mathrm{tr}\left[\begin{pmatrix} 0 & \frac{1}{4} & 1 & -\frac{1}{4} \\ \frac{1}{4} & 0 & -\frac{1}{4} & 1 \\ 1 & -\frac{1}{4} & 0 & \frac{1}{4} \\ -\frac{1}{4} & 1 & \frac{1}{4} & 0 \end{pmatrix}\begin{pmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix}\right] = 0.5.$$

As can be seen in Example 4.1, decomposing the penalty, $2\mathrm{tr}\big(H_{cv}\mathrm{Var}(\boldsymbol{y})\big)/n$, shows that the weights that relate to observations from the same cluster are multiplied by their positive covariance values and therefore contribute to the penalty, while weights that relate to observations from different clusters are multiplied by zero and therefore do not contribute to the penalty. As a result, the penalty of model A, which gives the full weight to observations from the same cluster, is larger than for model B. Therefore, in this scenario where $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$, while CV selects model A when $CV(A) < CV(B)$, CV_c selects model A when

$CV(A) < (CV(B) - 1.5)$. In that way CV_c, as well as Cp, balance the tendency to split based on the correlation structure of the training set. Obviously, when the observations in the training set are highly correlated, the penalty effect is stronger, and the superiority of RETCO over the standard algorithm is more prominent, compared to settings with lower correlation where the effect may be minor (see also Section 5.1.1).

Example 4.2 *Consider the setting of* $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$ *and* $\boldsymbol{y} \sim N_{100}(0.1 \times \boldsymbol{x}_1, V)$, *where*

$$V[i,j] = \begin{cases} 2, & \text{when } i = j, \\ 1, & \text{when } i \neq j \text{ and } (i,j \leq 50 \text{ or } i,j > 50), \\ 0, & \text{o.w,} \end{cases}$$

and

$$\boldsymbol{x}_1[i] = \begin{cases} 0.5 + \epsilon_x, & \text{when } i \text{ is odd}, \quad \epsilon_x \sim N(0, 0.1), \\ -0.5 + \epsilon_x, & \text{when } i \text{ is even}, \end{cases}$$

i.e., $\boldsymbol{y}$ *contains two clusters of* 50 *observations each, and its mean is not correlated with the clusters.*

Two models are tested, model A which uses the threshold $\boldsymbol{x}_1 = 0$, *and model B which uses* $\boldsymbol{x}_2 = 0$, *where* $\boldsymbol{x}_2$ *is highly correlated with the clusters:*

$$\boldsymbol{x}_2[i] = \begin{cases} 0.5 + \epsilon_x, & \text{when } i \leq 50, \\ -0.5 + \epsilon_x, & \text{when } i > 50. \end{cases}$$

A simulation of this setting is visualized in Figure 1. In this simulation $CV(A) = 1.47$ *and* $CV(B) = 1.04$, *while* $CV_c(A) = 2.47$ *and* $CV_c(B) = 3.04$. *Therefore, in this simulation under the* $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$ *prediction goal setting, while CV selects model B,* CV_c *selects model A.* [1]

Unlike Cp and CV_c, whose penalties depend on $\mathrm{Var}(\boldsymbol{y})$, the penalty of AIC in this setting ($\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$) is fixed regardless of the training covariance structure at $2p/n$. Therefore, we can conclude that in AIC the log-likelihood, $\ell(\boldsymbol{y}; \widehat{\mathbb{E}}(\boldsymbol{y}|X), V)$, is responsible for mixing uncorrelated training set observations in the different paths, while the penalty only affects the stopping rule.

4.2.2 $\boldsymbol{b}^* = \boldsymbol{b}$

When $\boldsymbol{b}^* = \boldsymbol{b}$, the correlation between $\boldsymbol{y}^*$ and $\boldsymbol{y}$ is the same as the correlation between observations in $\boldsymbol{y}$. Therefore, unlike in the $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$ setting, here there is no clear motivation to restrict the tendency to split the nodes based on the correlation structure of the training set (as is the case in standard regression trees). Correspondingly, the bias corrections in this setting are also different than in the $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$ setting. For example, CV is not biased in this setting, i.e., $CV_c = CV$ (for more details see Section 2.3). The penalty in Cp, $2\sigma^2 \mathrm{tr}(H)/n$, depends on $\mathrm{Var}(\boldsymbol{y})$ through H for some models (e.g., for LMM), however for other models it does not depend implicitly on $\mathrm{Var}(\boldsymbol{y})$. In any case, the effect of $\mathrm{Var}(\boldsymbol{y})$ is much less prominent than in the $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$ setting, where the bias is $2\mathrm{tr}\big(H\mathrm{Var}(\boldsymbol{y})\big)/n$. Similarly with the

1. For the $\boldsymbol{b}^* = \boldsymbol{b}$ setting, which will be discussed next, $CV_c = CV$ and both select model B.

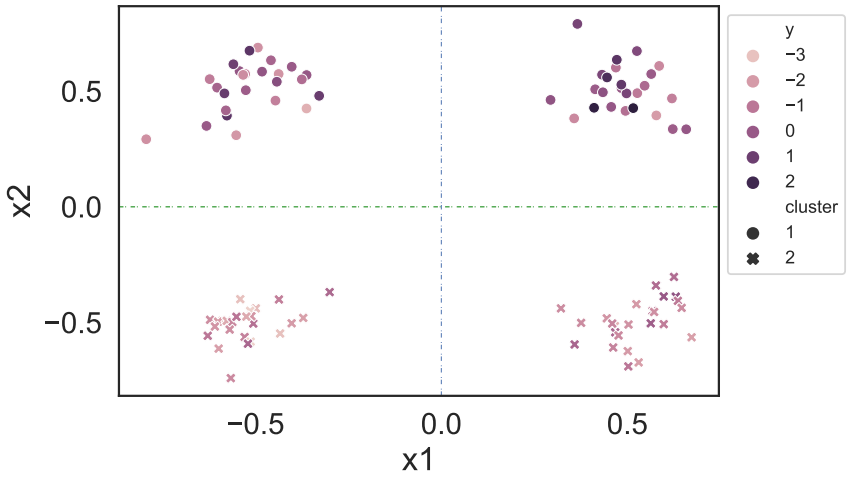

Figure 1: Illustration of Example 4.2 (see text for details). The dashed blue and green lines are the thresholds for model A and model B, respectively.

penalty in AIC, $2\text{tr}(H)/n$. Therefore, in this setting, both Cp and AIC penalties mainly affect the stopping rule rather than mix between uncorrelated observations. Given additional stopping rules (e.g., tree depth, minimal number of training set observations in each node), we can conclude that the effect of using prediction error estimator instead of training error is limited. This scenario is numerically demonstrated in Section 5.1.2. Still, it is important to emphasize that the proposed model—$f(\boldsymbol{x}^*) + \boldsymbol{b}^{*^t}\boldsymbol{z}^*$—is recommended also because of the inference and the use of a solid statistical perspective that the random effects framework enables.

4.2.3 $\boldsymbol{b}^* \not\perp\!\!\!\perp \boldsymbol{b}$ but $\boldsymbol{b}^* \neq \boldsymbol{b}$ Scenario

From a qualitative perspective, this scenario is the same as the $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$ scenario. In both scenarios, the correlation structure of $\boldsymbol{y}$ is not preserved in the prediction problem. As a result, the bias correction has a key role in balancing the tendency of standard regression trees to split based on the correlation structure of $\boldsymbol{y}$. The main difference between the scenarios is quantitative, and is explicitly expressed in the bias correction formulas. For example, the bias correction in Cp is $2\text{tr}\Big(H\big(\text{Var}(\boldsymbol{y}) - \text{Cov}(\boldsymbol{y}, \boldsymbol{y}^*)\big)\Big)/n$. This setting of $\boldsymbol{b}^* \not\perp\!\!\!\perp \boldsymbol{b}$ but $\boldsymbol{b}^* \neq \boldsymbol{b}$ is demonstrated in Section 5.2.

5. Numerical Results

This section compares the performance of RETCO to the standard regression tree algorithm and relevant modifications of it that will be described. The analysis is performed using both simulated data and real data sets for different correlation settings. The simulation part is based on random effects framework and presents results for $\boldsymbol{b}^* = \boldsymbol{b}$ as well as for $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$ correlation settings. The $\boldsymbol{b}^* \neq \boldsymbol{b}$ but $\boldsymbol{b}^* \not\perp\!\!\!\perp \boldsymbol{b}$ correlation setting is demonstrated in the random fields context using a real data set with spatial correlation. Also, different prediction

error estimator types (Cp, CV_c and AIC) and different tree-based models (regression tree and RF) are analyzed. The code is available in `https://github.com/AssafRab/RETCO`.

5.1 Simulation

The training set was generated from the following model:

$$\boldsymbol{y} = I_{(\boldsymbol{x}_1>0)} + I_{(\boldsymbol{x}_2>0)} + I_{(\boldsymbol{x}_3>0)} + I_{(\boldsymbol{x}_1>0)}I_{(\boldsymbol{x}_2>0)}I_{(\boldsymbol{x}_3>0)} + Z\boldsymbol{b} + \boldsymbol{\epsilon}, \tag{5}$$

where

- $I_{(\boldsymbol{x}_j>0)} \in \{0,1\}^n,\ \forall j \in [1,2,3]$ is the indicator vector for $x_{i,j} > 0,\ \forall i \in [1,...,n]$.
- The sample contains C clusters, each one of size $n_c = n/C$. $Z \in \mathbb{R}^{n\times C}$ indicates the clusters, i.e., for the first column the first n_c elements are 1, and the rest are zero, for the last column the last n_c elements are 1 and the rest are zero.
- $\boldsymbol{b} \in \mathbb{R}^C$ is the random effects vector, distributed $N_C(0, \sigma_b^2 I_C)$, and $\boldsymbol{\epsilon} \sim N_n(0, I_n)$.
- $\boldsymbol{x}_1,\ \boldsymbol{x}_2$ and $\boldsymbol{x}_3$ are generated as $Z\boldsymbol{\gamma}+\boldsymbol{\eta}$, where $\boldsymbol{\gamma} \sim N_C(0, \sigma_b^2 I_C)$ and η_i are uncorrelated and distributed uniformly, $U(-1,1), \forall i \in [1,...,n]$.

5.1.1 New Random Effects ($\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$)

As was mentioned in Section 2.2, when $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$ (i.e., $\text{Cov}(\boldsymbol{y}^*, \boldsymbol{y}) = 0$), then $\widehat{\boldsymbol{b}}^* = 0$ and the BLUP is reduced to GLS. Here, Cp prediction error estimator is used and therefore $\boldsymbol{y}^*$ is sampled from the same covariate values as in the training set, $\Phi = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, Z\}$. In order to reduce the variance of the prediction error estimate, the test sample contains 300 replicates of Φ. RETCO is compared to a standard regression tree, but with GLS predictor and with squared error loss function, which is the same loss as in Cp but without the bias correction term. In both algorithms, the stopping rules are depth of tree smaller than four, and number of observations in the terminal node greater than two. The relative difference between the RETCO test error, error(RETCO), and its alternative:

$$\text{error difference}[\%] = \frac{\text{error(RETCO)} - \text{error(standard tree)}}{\text{error(standard tree)}},$$

is calculated repeatedly for 100 simulation runs. The average simulation run time is 82.1 seconds, where RETCO takes on the order of 3-8 fold longer to run due to its iterative approach. For more details about RETCO's computational complexity, see Section 4.

Figure 2, left panel, presents boxplots of error difference[%] for different σ_b^2 values. As expected, when σ_b^2 is larger (i.e., the correlation is stronger), the improvement in using RETCO over the standard regression tree algorithm is bigger. However, on average, also for small σ_b^2 values, RETCO is slightly better than the standard regression tree algorithm. Appendix D presents additional comparison for different sample sizes (n), different cluster sizes (n_c), different true model setting than the setting in equation (5), as well as for generalization error setting using CV_c (instead of Cp).

As was mentioned in Section 4.2, RETCO balances the tendency of standard tree to split based on the correlation structure of the training sample. Therefore, we expect that

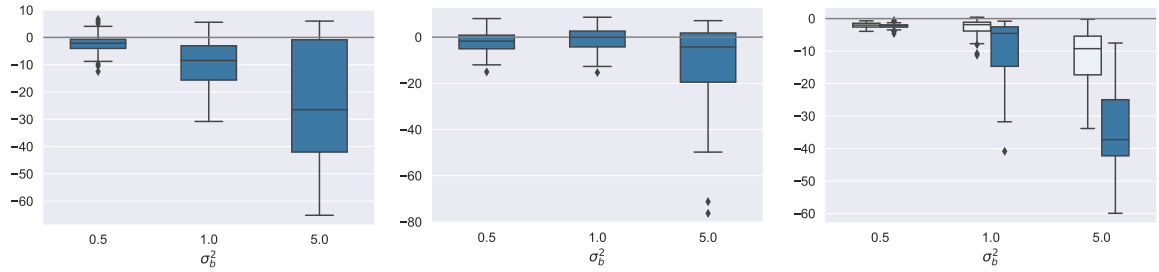

Figure 2: Boxplots of the error difference[%] for $n = 300$, $n_c = 50$ and different σ_b^2 (0.5, 1, 5) for different settings. Left: $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$ setting (Section 5.1.1). The means are -2.5, -9.9 and -25.2. Middle: $\boldsymbol{b}^* = \boldsymbol{b}$ setting (Section 5.1.2). The means are -2.1, -1.1 and -10.4. Right: RF setting with $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$ (Section 5.1.3). Two versions of RETCO are analyzed, in light the version that does not enforce the stopping rule constraint, and in bold the version that enforces the constraint. The means are -2.1, -2.9 -11.7 and -2.2, -9.6 and -34.7, respectively.

RETCO mixes training set observations from different clusters in the leaves more than the standard regression tree. The following measure quantifies this mixing property:

$$\text{homogeneity} = \sum_{s \in \mathcal{S}} \sum_{c=1}^{C} |n(c,s) - n(s)/C|,$$

where $n(c,s)$ is the number of training set observations in leaf s that belong to cluster c and $n(s) = \sum_{c=1}^{C} n(c,s)$. Smaller homogeneity means bigger mixing. Figure 3 plots the training sample homogeneity difference[%],

$$\frac{\text{homogeneity(RETCO)} - \text{homogeneity(standard tree)}}{\text{homogeneity(standard tree)}},$$

versus the error difference[%] for different σ_b^2 values. As can be seen in Figure 3, error difference[%] has a positive correlation with homogeneity difference[%], i.e., the property of RETCO to balance the tendency of standard regression tree to split based on the correlation structure of the training sample is essential.

5.1.2 Same Random Effects ($\boldsymbol{b}^* = \boldsymbol{b}$)

For the $\boldsymbol{b}^* = \boldsymbol{b}$ scenario, the training set is the same as in Section 5.1.1, but the prediction set is different, such that the random effects realizations from the training set are also used for constructing $\boldsymbol{y}^*$.

For RETCO, AIC loss function with LMM predictor for $\boldsymbol{y}$ is used for splitting. As presented in Section 4, the predictor in the tree's leaves is GLS and the random effects term is added after fitting the tree. For the standard regression tree, negative normal log likelihood loss function is used with no distinction between random and fixed effects, i.e., all the covariates, including the cluster, can be selected for splitting. Correspondingly, the negative log likelihood of i.i.d. normal distribution (which is effectively the same as squared

Figure 3: Homogeneity difference[%] versus error difference[%] for $n = 300$, $n_c = 50$ and different σ_b^2 values.

error loss) is used as a loss function for the standard regression tree algorithm. The middle panel of Figure 2 compares between the algorithms for different σ_b^2 values (when $n = 300$ and $n_c = 50$). As we can seen in the figure, RETCO outperforms the standard tree algorithm. For $\sigma_b^2 = 0.5$ and $\sigma_b^2 = 1$, the average error difference[%] is close to zero. As was mentioned in Section 4.2.2, this phenomenon is expected.

5.1.3 RF - New Random Effects

RF is demonstrated for $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$ and Cp loss function setting. The training sample model is:

$$\boldsymbol{y} = I_{(\boldsymbol{x}_1>0)} + I_{(\boldsymbol{x}_2>0)} + I_{(\boldsymbol{x}_3>0)} + I_{(\boldsymbol{x}_1>0)}I_{(\boldsymbol{x}_2>0)}I_{(\boldsymbol{x}_3>0)} + I_{(\boldsymbol{x}_4>0)}I_{(\boldsymbol{x}_5>0)}I_{(\boldsymbol{x}_6>0)} + Z\boldsymbol{b} + \boldsymbol{\epsilon},$$

where $n = 500$, $\boldsymbol{x}_i \; \forall i \in \{1, ..., 6\}$, $Z\boldsymbol{b}$, and $\boldsymbol{\epsilon}$ have the same distribution as in Section 5.1.1. Additional parameters that are relevant for RF are:

- the maximal tree depth is 10,
- the number of regression trees is $T = 100$,
- a random half-sample method is used for sampling the training set for each tree (i.e., the training sample size for each tree is 250 without duplicates),
- three covariates are randomly selected at each split, following the rule of thumb of selecting randomly $round\big(\log_2(p)\big)$ potential covariates at each split.

Also, two versions of RETCO are tested: the first uses the stopping rule constraint as presented in RETCO in inequality (3), the second does not enforce this constraint, and therefore results in deeper trees. The prediction set contains new random effects realizations, such that $\text{Cov}(\boldsymbol{y}^*, \boldsymbol{y}) = 0$. The covariates of the prediction set are 200 replicates of the covariates of the training set. This simulation was repeated 50 times. The right panel in Figure 2 presents boxplots of the error difference[%] for different σ_b^2. As can be seen, both versions of RETCO outperform the standard algorithm. Also, enforcing the stopping rule gives better results.

5.1.4 Comparison With Previous Algorithms

The competitors in the left and the right graphs in Figure 2 preserve the main characteristics of the MRF algorithm (Stephan et al., 2015): taking into account the correlation structure by using GLS estimator and differentiation between random and fixed effects. The exact MRF algorithm, which was designed for a genetic application, is not implemented here since some of its technical details are specific for genetic applications, which are not our main use case. A comparison between RETCO and RE-EM algorithm (Sela and Simonoff, 2012) is presented in Appendix D. As expected, RETCO's performance is uniformly superior to both algorithms due to its use of prediction error estimates for splitting, and the careful consideration of correlation structures in splitting and prediction.

5.2 Real Data Analysis

This section presents real data analyses comparing the performance of the standard regression tree and RF algorithms to their RETCO versions for six different data sets with various correlation structures. The data sets and the prediction problems are described in Section 5.2.1, then the results are presented in Section 5.2.2.

5.2.1 Prediction Problems Description

The list below briefly describes the different data sets and their prediction problems, additional technical information can be found in Table 2 and Appendix D. We note that for all problems we made the choice to divide the data into a relatively small training set and larger (in some cases, much larger) test set. This serves two purposes: by making the training sets smaller, we are increasing the difficulty of the problem and the importance of properly dealing with random effects in obtaining good performance; and by having large test sets, we can obtain more reliable estimates of model performance.

FIFA – This data set contains football players' market-values. The data set has a clustered correlation structure, where the cluster variable is the player's club, such that market-values of players from the same club are correlated, but from different clubs are not correlated. The prediction goal is to predict the market-values of new players from new clubs. In order to satisfy this prediction goal, the training set contains the observations of players from 20 clubs that were randomly sampled (548 observations), and the test set contains the observations of the other clubs ($17,939$ observations). Since the covariate values of the prediction set are not the same as the covariate values of the training set, CV-type loss function is used in the algorithms' splitting criterion – CV_c for RETCO and CV for the standard regression tree/RF algorithms. The prediction error is estimated by the average squared error of the test set.

Communities and Crime – This data set presents the number of violent crimes per population size in US communities. The data set has a clustered correlation structure, where the clusters are the US states (each state contains many communities). The training set contains 15 clusters that were randomly sampled (790 observations), where the test set contains the other clusters ($1,204$ observations). For the same reason as in the FIFA data set, CV-type loss function is used in the algorithms' splitting criterion.

South Korea Temperature ('bias correction of numerical prediction model temperature forecast') – This data set contains daily maximal temperature measurements (collected in August between the years $2013-2017$) at several sites in South Korea. The prediction goal is to predict the maximal temperature of new days. Measurements of the first two years were selected (575 observations) in order to predict the maximal temperature of the same set of days in the next years ($2,325$ observations). Due to the spatial correlation structure, exponential kernel covariance function was used for modeling. Since all the records are measured at the same sites, in-sample error type is used in the algorithms' splitting criterion (Cp for RETCO and training error with a squared error loss function for the standard regression tree/RF).

California Housing Prices – This data set contains the median house value within a block for different blocks in California. Some of the blocks belong to the same clusters (same coordinate values), therefore the data set has a clustered-spatial correlation structure, which can be represented by the following kernel covariance function:

$$\mathrm{Cov}(y_i, y_j) = \mathcal{K}(|\boldsymbol{z}_i - \boldsymbol{z}_j|) + \sigma_b^2 I_{cluster(i)=cluster(j)} + \sigma_\epsilon^2 I_{i=j},$$

where $\mathcal{K}(\cdot)$ is the exponential kernel covariance function, $\boldsymbol{z}_i, \boldsymbol{z}_j$ are the coordinates and $cluster(i)$, $cluster(j)$ are the clusters of y_i, y_j. The prediction goal is to predict the median house value of new blocks from new clusters. Therefore, 100 clusters are randomly sampled (279 observations) for the training set and the other clusters are used as the test set ($12,124$ observations). Since the prediction goal is to predict median house values from new clusters, then $\boldsymbol{b}^* \neq \boldsymbol{b}$. However, due to the spatial correlation, the prediction set median house values are correlated with the training set median house values, and therefore this setting satisfies the $\boldsymbol{b}^* \neq \boldsymbol{b} \cap \boldsymbol{b}^* \not\perp \boldsymbol{b}$ scenario (see Section 2.2.1). CV-type loss function is used in the algorithms' splitting criterion since the covariate points of the training set and the prediction set are different.

Parkinson's Disease Telemonitoring – This longitudinal data set contains Parkinson's disease symptom scores of 42 individuals along six-months trial. The clustered-temporal correlation structure, where the clusters refer to the individuals, can be modeled by LMM with random intercept for the cluster and random slope for the time variable. The prediction goal is predicting the score of new individuals, therefore five individuals were randomly sampled (742 observations) for the training set and the others were designated as the test set ($5,179$ observations). The covariates in this data set are biomedical voice measurements and their values are approximately the same for all the individuals. Therefore, Cp and training error with a squared error loss functions are used in the splitting criterion for RETCO and standard regression tree/RF, respectively.

Wages – This longitudinal data set presents the average hourly wages by year of 888 employees. As in the Parkinson's Disease Telemonitoring data set, this data set can be modeled by LMM with random intercept and random slope, where the employee is the cluster variable. The prediction goal is to predict the average hourly wage of new employees. 50 individuals were randomly sampled (331 observations) for the training set and the other are used as the test set ($6,071$ observations). Since the covariate values of the training set and the prediction set are different, CV-type loss function is used in the algorithms' splitting criterion.

Data set name	Correlation variables	Covariance tools	Reference	Link
FIFA	Player's club	Random intercept LMM	-	Kaggle
Crimes	State	Random intercept LMM	Redmond and Baveja (2002)	UCI
Korea Temperature	Coordinates + Day	Blocked GPR exponential kernel	Cho et al. (2020)	UCI
California Housing	Coordinates + Blocks' clusters	GPR exponential kernel and random intercept	Pace and Barry (1997)	Kaggle
Parkinson's Disease	Patient + Time	Random intercept and slope LMM	Tsanas et al. (2009)	UCI
Wages	Employee + Time	Random intercept and slope LMM	-	Brolgar package

Table 2: Data sets description

5.2.2 Real Data Results

Table 3 summarizes the results. As can be seen, for all the six data sets the test error of the standard regression tree and RF algorithms are greater than the test error of their RETCO versions (the error difference [%]'s are negative). Moreover, in several cases the improvement of RETCO over the standard algorithm is very big. These results, which cover various types of correlation structures and prediction goals, stress the importance in using RETCO as a unified framework for tree-based model in settings involving correlated data.

Correlation Structure	Data set Name	Regression Tree	RF
Clusters	FIFA	-4.9%	-7.4%
	Crimes	-8.1%	-2.8%
Spatial	Korea Temperature	-14.5%	-13.6%
	California Housing	-14.2%	-3.3%
Longitudinal	Parkinson's Disease	-32.9%	-35.4%
	Wages	-20.0%	-7.9%

Table 3: Error difference [%] between RETCO and standard regression tree, and between RETCO and standard RF.

6. Conclusions

This paper presents a new algorithm, RETCO, for fitting regression tree-based models for correlated data. Analyzing various settings with different correlation structures leads to the

conclusion that RETCO substantially improves prediction performance in settings involving correlated data.

Standard regression tree-based models ignore the correlation structure of the data and are thus clearly ill-fitting for dealing with correlated data situations. Various previous and parallel efforts have been made to develop approaches that take correlations into account, as reviewed in Section 3.1. Our suggested algorithm RETCO is unique in its comprehensive and flexible approach. It accounts for the correlation structure in various ways, including using prediction error estimates for correlated data as the loss function in the splitting criterion. As discussed and demonstrated, using prediction error estimators for correlated data instead of training error neutralizes the tendency to fit a tree that divides the training set based on its correlation structure, as is likely to happen in standard regression tree-based models.

Extensive data analysis, including analysis of six different real data sets with different correlation structures and models, shows the superiority of RETCO over standard regression tree-based models, as well as its generality that enables to implement it under various settings.

Acknowledgments

This work was supported by the Israeli Science Foundation, grant 1804/16 and by the European Union Seventh Framework Programme grant agreement no. 785907 (Human Brain Project).

Appendix A. Theoretical Background

This appendix section extends the theoretical background that is given in Section 2.

A.1 Regression Tree, Random Forest and Gradient Boosting

Algorithm 2 presents a typical regression tree algorithm.

Algorithm 2 A tree-based algorithm

Input: $\boldsymbol{y}$, X.
Output: $f(\cdot)$.
General setting: select a training error loss function and define stopping rules.
Initialization: $\mathcal{S} = \{g_1, \mu_1\}$, where $g_1 = \mathbb{R}^p$, $\mu_1 = \sum_{i=1}^{n} y_i/n$.
repeat

1. Given the predefined stopping rules, for each node $s \in \mathcal{S}$ solve the following optimization problem:

$$c_s, j_s = \underset{c\in\mathbb{R}, j\in J_s}{\operatorname{argmin}} \sum_{i\in I_s} Loss\big(y_i, I_{(x_{i,j}\le c)}\mu_s^l(c) + I_{(x_{i,j}>c)}\mu_s^r(c)\big),$$

   where J_s is the set of available covariates for splitting node s, $I_s = \{i|\boldsymbol{x}_i \in g_s\}$, $\mu_s^l(c)$ and $\mu_s^r(c)$ are the mean estimators of $\{y_i|\boldsymbol{x}_i \in g_s \cap x_{i,j} \le c\}$ and $\{y_i|\boldsymbol{x}_i \in g_s \cap x_{i,j} > c\}$, respectively.

2. Update $\mathcal{S}$ by replacing (g_s, μ_s) by the new two nodes: $\big(g_s \cap x_{j_s} \le c_s, \mu_s^l(c_s)\big)$, $\big(g_s \cap x_{j_s} > c_s, \mu_s^r(c_s)\big)$, where x_{j_s} is the covariate j_s and $\mu_s^l(c_s), \mu_s^r(c_s)$ are the related mean predictors.

until Stopping rules are satisfied $\forall s \in \mathcal{S}$

Random forest (RF) and gradient boosting (GB) predictor are based on averaging an ensemble of trees:

$$f(\boldsymbol{x}^*) = \sum_{t=1}^{T} \lambda_t \sum_{s=1}^{S_t} I_{(\boldsymbol{x}^*\in g_{t,s})}\mu_{t,s},$$

where T is the number of trees and $\lambda_t \in (0, 1]$ is the learning rate (for RF $\lambda_t = 1/T$, $\forall t$).

The regression trees in RF and GB are fitted in different ways than in a standard regression tree. In RF, the training set of each tree is sampled from the original sample (e.g., sampling with replacement of size n, half-sample), and the set of the potential covariates of each split is a random sample of J_s. In GB the trees are dependent and created consecutively, where the dependent variable of each tree is the residual of the previous tree. Also, there are many techniques for reducing over-fitting and model variance which are relevant for RF and GB, but not relevant for standard regression tree model. For more information about RF and GB see Freund et al. (1999); Friedman (2001); Breiman (2001); Hastie et al. (2009).

Note, unlike in regression tree model, which tends to over-fit and therefore suffers from high variance, RF has relatively low variance due to the averaging over the T trees. This property affects the optimal structure of the trees in RF. While the tree depth in regression tree model should be restricted in order to avoid over-fitting, the trees in RF can be large

whenever T is respectively large (Criminisi et al., 2011). Since the trees in RF are correlated, the RF variance decreases in a smaller rate than T. Commonly the trees' depth in RF is also restricted for various reasons, such as the computational cost of RF with deep trees (and consequently large T).

A.2 Prediction Error Estimation and Model Selection for Correlated Data

A.2.1 Cp

The original Cp, when $\text{Cov}(\boldsymbol{y}, \boldsymbol{y}) = \sigma^2 \times I_n$ and $\text{Cov}(\boldsymbol{y}^*, \boldsymbol{y}) = 0$, was introduced by Mallows (1973):

$$Cp = \frac{1}{n}\|\boldsymbol{y} - \widehat{\boldsymbol{y}}\|_2^2 + \frac{2\sigma^2}{n}p.$$

A.2.2 AIC

The standard AIC under normality and i.i.d. assumptions was introduced by Akaike (1974):

$$AIC = -\frac{2\ell\big(\boldsymbol{y}; \widehat{\mathbb{E}}(\boldsymbol{y}|X), \sigma^2\big)}{n} + \frac{2p}{n},$$

where $\ell\big(\boldsymbol{y}; \widehat{\mathbb{E}}(\boldsymbol{y}|X), \sigma^2\big)$ is the log-likelihood of $\boldsymbol{y}$.

Appendix B. Estimating Variance Components for Clustered Data

When $\boldsymbol{y}$ has a clustered correlation structure, i.e., its covariance matrix follows:

$$\text{Cov}[i,j] = \begin{cases} \sigma_\epsilon^2 + \sigma_b^2, & \text{when } i = j, \\ \sigma_b^2, & \text{when } i \neq j \text{ but } c(i) = c(j)\ , \\ 0, & \text{o.w,} \end{cases}$$

where $c(i)$ is the cluster that observation i belongs to and $\sigma_\epsilon,\ \sigma_b$ are in R^+, then σ_ϵ^2 and σ_b^2 can be estimated in a closed-form way. In order to simplify the equations let us assume that $\mathbb{E}\boldsymbol{y} = 0$. In this case:

$$\widehat{\sigma}_\epsilon^2 = \frac{\sum_{i=1}^n \big(y_i - \bar{y}(i)\big)^2}{(n - C)},$$

where $\bar{y}(i)$ is the average of the cluster that y_i belongs to, and C is the number of clusters,

$$\widehat{\sigma}_b^2 = \Big(\frac{\sum_{i=1}^n (\bar{y}\big(i) - \bar{y}\big)^2}{C-1} - \widehat{\sigma}_\epsilon^2\Big) \times \frac{C-1}{n - \sum_{j=1}^C n_j^2/n},$$

where $\bar{y}$ is the average of $\boldsymbol{y}$ and n_j is the number of observations in cluster j. When $\mathbb{E}(\boldsymbol{y}) \neq 0$, the variance parameters should be calculated for the residual, $\boldsymbol{y} - \widehat{E}\boldsymbol{y}$.

Algorithm 3 RE-EM Algorithm

Input: $\boldsymbol{y}$, X, Z.
Output: $f(\cdot)$, $\widehat{\boldsymbol{b}}$.
Initialization: $\widehat{\boldsymbol{b}}^{(0)} = 0$.
repeat

1. Using the fixed effects covariates, fit CART algorithm (Breiman et al., 1984) for $(y - Z\widehat{\boldsymbol{b}}^{(k-1)})$ and extract the selected regions, $\{g_s^{(k)}\}_{s=1}^{S^{(k)}}$, from the fitted tree.
2. Estimate $\{\mu_s\}_{s=1}^{S}$, G, and σ^2 using the following LMM model:

$$y_i = \sum_{s=1}^{S^{(k)}} I_{(\boldsymbol{x}_i \in g_s^{(k)})} \mu_s^{(k)} + \boldsymbol{z}_i \boldsymbol{b} + \epsilon_i,$$

   where $\boldsymbol{b} \sim N_q(0, G)$, $\epsilon_i \sim N(0, \sigma^2)$.

3. Given $\{\mu_s^{(k)}\}_{s=1}^{S^{(k)}}$, $\widehat{G}^{(k)}$ and $\widehat{V}^{(k)} = Z\widehat{G}^{(k)}Z^t + \widehat{\sigma}^{2,(k)} I_n$: estimate $\widehat{\boldsymbol{b}}^{(k)}$ using the BLUP formula.

until Convergence of $\widehat{\boldsymbol{b}}^{(k)}$.

Appendix C. Comparison with Other Methods

Algorithm 3 presents the RE-EM algorithm (Sela and Simonoff, 2012). Algorithm 4 presents the MRF algorithm (Stephan et al., 2015) for a single tree.[2] For MRF, in order to simplify notations, $\{\boldsymbol{y}, X, Z\}$ denotes the bootstrap sample, and the features sampling at each split is ignored.

Appendix D. Numerical Results

This appendix section presents additional results that are related to Section 5.1, as well as detailed information relating to the settings in Section 5.2.

D.1 Regression Tree - New Random Effects ($\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$)

D.1.1 In-Sample Error Setting

Given the simulation setting that is described in Section 5.1.1, Figure 4, left panel, presents the effect of the cluster size (n_c), on the performance of RETCO. As can be seen in the graph, the average error difference[%] is smaller for the larger block size ($n_c = 150$).

Figure 4, middle panel, presents the effect of the sample size on the performance of RETCO. As can be seen, RETCO performs better for all the settings. Also, as expected, the error difference[%] variance is small for the large sample size. The variance depends on the maximal three levels has potentially eight predictors, which is a large amount of predictors

2. Stephan et al. (2015) do not supply an organized algorithm, Algorithm 4 tries to formalize their approach as given in their supplementary material.

Algorithm 4 Mixed Random Forest (algorithm for a single tree)

Input: $\boldsymbol{y}$, X, Z.
Output: $f(\cdot)$, $\widehat{\sigma}_b^2, \widehat{\sigma}^2$.
Initialization: $g_1 = \{0,1\}^J$. Also, estimate $\delta = \sigma^2/\sigma_b^2$.
repeat

1. Given the predefined stopping rules, for each node $s \in \mathcal{S}$, find the following parameters:

$$\tilde{j_s}, \tilde{\sigma}^2 = \underset{j_s \in J_s, \sigma^2 \in \mathbb{R}^+}{\operatorname{argmax}} \ell(\boldsymbol{y}; \sigma^2, \mu_s^l, \mu_s^r, \{\mu_m\}_{m \in \mathcal{S}/s}, g_s \cap (x_{j_s} = 0), g_s \cap (x_{j_s} = 1), \{g_m\}_{m \in \mathcal{S}/s}),$$

where μ_s^l, μ_s^r and $\{\mu_m\}_{m \in \mathcal{S}/s}$ are the GLS estimators for $\{y_i | \boldsymbol{x}_i \in g_s \cap (x_{i,j_s} = 0)\}$, $\{y_i | \boldsymbol{x}_i \in g_s \cap (x_{i,j_s} = 1)\}$, and $\{y_i | \boldsymbol{x}_i \in g_m\}_{m \in \mathcal{S}/s}$ subsets, respectively.

2. Update $f(\cdot)$: replace each node $s \in \mathcal{S}$ by $\{g_s \cap (x_{\tilde{j}_s} = 0), \mu_s^l\}$ and $\{g_s \cap (x_{\tilde{j}_s} = 1), \mu_s^r\}$.

until Stopping rules are satisfied $\forall s \in \mathcal{S}$.

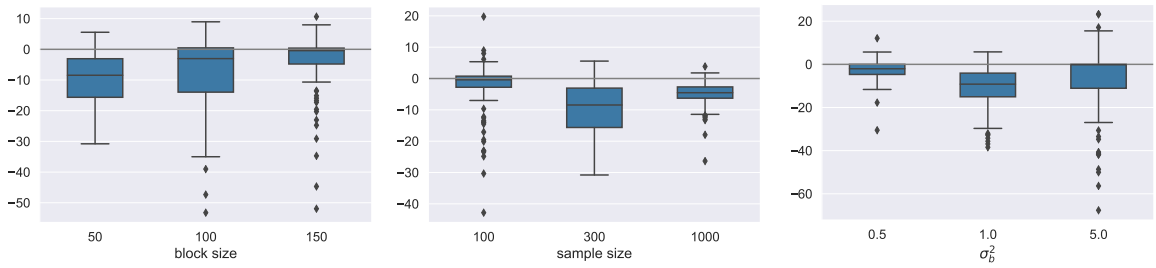

Figure 4: Boxplots of the error difference[%]. Left: In-sample error setting, $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$, $n = 300$, $\sigma_b^2 = 1$ and different n_c. The means are -9.9, -8.0 and -4.3. Middle: In-sample error setting, $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$, $\sigma_b^2 = 1$ $n_c = 50$ and different n. The means are -2.9, -9.9 and -5.0. Right: Generalization error setting, $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$, $n = 300$, $n_c = 50$ and different σ_b^2. The means are -2.8, -11.0 and -7.3.

when $n = 100$, but small when $n = 1000$. Therefore, when $n = 100$ the tree predictions are noisy for both algorithms, and their relative difference is noisy as well.

D.1.2 Generalization Prediction Error Setting

In order to analyze RETCO performance in generalization prediction error problem, the test set setting was changed such that the prediction set covariates are nonidentical to the training sample covariates (but are sampled from the same distribution). As was described in Section 2.3, CV_c loss function estimates the generalization error unbiasedly by correcting the standard CV error. Therefore, CV_c loss function is used in RETCO and CV loss function is used for the standard regression tree algorithm. Figure 4, right panel, presents the error difference[%] for different σ_b^2.

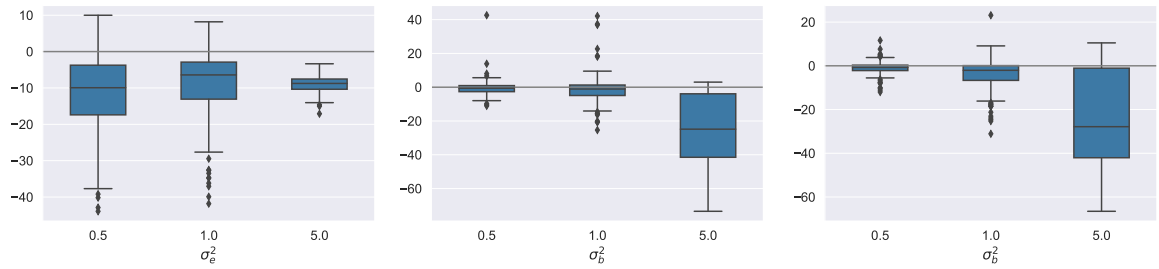

Figure 5: Boxplots of the error difference[%]. Left: In-sample error setting, $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$, $n = 300$, $\sigma_b^2 = 1$ and different σ^2. The means are -11.6, -9.1 and -8.9. Middle: In-sample error setting, $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$, for new true model with continues covariates functions. The means are -0.7, -1.2 and -25.4. Right: In-sample error setting, $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$, for new true model with continues covariates functions, noise covariates and different distributions of $\boldsymbol{x}_j$. The means are -0.1, -3.8 and -25.2.

D.1.3 Additional Settings

Figure 5 presents the performance of RETCO for various additional settings.

- Left panel: the same setting as in Section 5.1.1, but for different σ_ϵ^2 (and $\sigma_b^2 = 1$).
- Middle panel: the same setting as in Section 5.1.1, but with a different true model, with continues covariates functions:

$$\boldsymbol{y} = \sin(\boldsymbol{x}_1) + \sin(\boldsymbol{x}_2) + \sin(\boldsymbol{x}_3) + \sin(\boldsymbol{x}_1) \times \sin(\boldsymbol{x}_2) \times \sin(\boldsymbol{x}_3) + Z\boldsymbol{b} + \boldsymbol{\epsilon}, \qquad (6)$$

  where sin is the Sine function.

- Right panel: the same setting as in equation (6), but with new available covariates for splitting, $\boldsymbol{x}_4$, $\boldsymbol{x}_5$, $\boldsymbol{x}_6$, that do not relate to $\boldsymbol{y}$. Also, the distribution of the covariates are different:
  - $\boldsymbol{x}_j = Z\boldsymbol{\gamma} + \boldsymbol{\eta}$, $\forall\, j \in \{1, 3, 5\}$, where $\boldsymbol{\gamma} \sim N_C(0, \sigma_b^2 I_C)$ and η_i are uncorrelated and distributed uniformly, $U(-1, 1)$,
  - $\boldsymbol{x}_j = \boldsymbol{\eta}$ $\forall\, j \in \{2, 4, 6\}$.

D.2 Comparison with RE-EM

Figure 6, left figure, presents the error difference between RETCO and RE-EM for the scenario when $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$. All the other settings are the same as in Section 5.1.1. Similarly, Figure 6, right figure, presents the error difference between RETCO and RE-EM for the scenario when $\boldsymbol{b}^* = \boldsymbol{b}$. All the other settings are the same as in Section 5.1.2. As can be seen in Figure 6, RETCO performs better than RE-EM.

Figure 6: Boxplot of Error difference[%] between RETCO and RE-EM. Left figure: $\boldsymbol{b}^* \perp\!\!\!\perp \boldsymbol{b}$ setting. The means are -0.5, -5.2 and -25.5. Right Figure: $\boldsymbol{b}^* = \boldsymbol{b}$ setting. The means are -10.8, -26.4 and -28.86.

D.3 Real Data Analysis

Detailed information about the settings of the real data analyses in Section 5.2 is given below.

In all implementations, the minimum number of observations in a node is 3. For RF implementations, the number of covariates that were sampled at each split is $round\big(log_2(p)\big)$, the number of trees is 80, and the maximal tree depth is ten. For regression tree implementations, the maximal regression tree depth is five.

Additional technical information for each data set:

FIFA-

- Dependent variable: Players' market values
- Number of covariates: 11
- Comments: The covariates that are used in this analysis are: 'Age', 'Overall', 'Potential', 'Wage', 'Special', 'Preferred Foot', 'International Reputation', 'Weak Foot', 'Skill Moves', 'Height', 'Weight'. Other variables have many missing values or are irrelevant.

Crimes (Communities and Crime in US)-

- Dependent variable: Violent crimes in US communities per population size
- Number of covariates: 100 (all the available covariates were used)

Korea Temperature ('bias correction of numerical prediction model temperature forecast')-

- Dependent variable: Daily maximum temperature at several sites in South Korea
- Number of covariates: 4
- Comments:
  - The covariates that are used in this analysis are: 'Present_Tmin','DEM', 'Slope', 'Solar radiation'. Other variables in this data set are models scores of the data set supplier, which are based on previous dependent variable measurements (and therefore cannot be used for LMM).

- Exponential kernel covariance function was used. Maximal temperature in different days are assumed to be uncorrelated.
- The original data set contains records from July and August. Due to many missing values in July along the years, only records from August are analyzed. Also, sites with missing values along the years were omitted.

California Housing-

- Dependent variable: Values of houses in California
- Number of covariates: 6 (all the available covariates were used)

Parkinson's Disease Telemonitoring-

- Dependent variable: Total UPDRS score, which is a score of Parkinson's disease progression
- Number of covariates: 18
- Comments: All the supplied covariates were used except the motor_UPDRS (which its relation with the dependent variable is not fully clear to us).

Wages-

- Dependent variable: Average hourly wages
- Number of covariates: 6 (all the available covariates were used)

References

Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

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

Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

Louis Capitaine, Robin Genuer, and Rodolphe Thiébaut. Random forests for high-dimensional longitudinal data. *Statistical Methods in Medical Research*, 30(1):166–184, 2021.

Matthew S Caywood, Daniel M Roberts, Jeffrey B Colombe, Hal S Greenwald, and Monica Z Weiland. Gaussian process regression for predictive but interpretable machine learning models: An example of predicting mental workload across tasks. *Frontiers in human neuroscience*, 10:647, 2017.

Dongjin Cho, Cheolhee Yoo, Jungho Im, and Dong-Hyun Cha. Comparative assessment of various machine learning-based bias correction methods for numerical weather prediction model forecasts of extreme air temperatures in urban areas. *Earth and Space Science*, 7 (4), 2020.

Brent A Coull, Joel Schwartz, and MP Wand. Respiratory health and air pollution: additive mixed model analyses. *Biostatistics*, 2(3):337–349, 2001.

Antonio Criminisi, Jamie Shotton, and Ender Konukoglu. Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning. *Microsoft Research Cambridge, Tech. Rep. MSRTR-2011-114*, 5(6):12, 2011.

Roxane Duroux and Erwan Scornet. Impact of subsampling and tree depth on random forests. *ESAIM: Probability and Statistics*, 22:96–128, 2018.

Marjolein Fokkema, Niels Smits, Achim Zeileis, Torsten Hothorn, and Henk Kelderman. Detecting treatment-subgroup interactions in clustered data with generalized linear mixed-effects model trees. *Behavior research methods*, 50(5):2016–2034, 2018.

Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

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

Wei Fu and Jeffrey S Simonoff. Unbiased regression trees for longitudinal and clustered data. *Computational Statistics & Data Analysis*, 88:53–74, 2015.

Pierre Goovaerts. Geostatistics in soil science: state-of-the-art and perspectives. *Geoderma*, 89(1-2):1–45, 1999.

Ahlem Hajjem, François Bellavance, and Denis Larocque. Mixed effects regression trees for clustered data. *Statistics & probability letters*, 81(4):451–459, 2011.

Ahlem Hajjem, François Bellavance, and Denis Larocque. Mixed-effects random forest for clustered data. *Journal of Statistical Computation and Simulation*, 84(6):1313–1328, 2014.

David Harville. Extension of the Gauss-Markov theorem to include the estimation of random effects. *The Annals of Statistics*, 4(2):384–395, 1976.

Trevor Hastie, Robert Tibshirani, and JH Friedman. *The elements of statistical learning: data mining, inference, and prediction*. New York, NY: Springer, 2009.

Tomislav Hengl, Madlene Nussbaum, Marvin N Wright, Gerard BM Heuvelink, and Benedikt Gräler. Random forest as a generic framework for predictive modeling of spatial and spatio-temporal variables. *PeerJ*, 6, 2018.

James S Hodges and Daniel J Sargent. Counting degrees of freedom in hierarchical and other richly-parameterised models. *Biometrika*, 88(2):367–379, 2001.

Colin L Mallows. Some comments on c_p. *Technometrics*, 15(4):661–675, 1973.

Che Ngufor, Holly Van Houten, Brian S Caffo, Nilay D Shah, and Rozalina G McCoy. Mixed effect machine learning: A framework for predicting longitudinal change in hemoglobin a1c. *Journal of biomedical informatics*, 89:56–67, 2019.

R Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

Amichai Painsky and Saharon Rosset. Cross-validated variable selection in tree-based methods improves predictive performance. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2142–2153, 2016.

Massimo Pellagatti, Chiara Masci, Francesca Ieva, and Anna M Paganoni. Generalized mixed-effects random forest: A flexible approach to predict university student dropout. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 14(3):241–257, 2021.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Assaf Rabinowicz and Saharon Rosset. Cross-validation for correlated data. *Journal of the American Statistical Association*, 117(538):718–731, 2022.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

Michael Redmond and Alok Baveja. A data-driven software tool for enabling cooperative information sharing among police departments. *European Journal of Operational Research*, 141(3):660–678, 2002.

Arkajyoti Saha, Sumanta Basu, and Abhirup Datta. Random forests for spatially dependent data. *Journal of the American Statistical Association*, pages 1–19, 2021.

Rebecca J Sela and Jeffrey S Simonoff. RE-EM trees: a data mining approach for longitudinal and clustered data. *Machine learning*, 86(2):169–207, 2012.

Jaime Lynn Speiser, Bethany J Wolf, Dongjun Chung, Constantine J Karvellas, David G Koch, and Valerie L Durkalski. BiMM forest: A random forest method for modeling clustered and longitudinal binary outcomes. *Chemometrics and Intelligent Laboratory Systems*, 185:122–134, 2019.

Johannes Stephan, Oliver Stegle, and Andreas Beyer. A random forest approach to capture genetic effects in the presence of population structure. *Nature communications*, 6:7432, 2015.

Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133, 1974.

Athanasios Tsanas, Max Little, Patrick McSharry, and Lorraine Ramig. Accurate telemonitoring of parkinson's disease progression by non-invasive speech tests. *Nature Precedings*, pages 1–1, 2009.

Florin Vaida and Suzette Blanchard. Conditional akaike information for mixed-effects models. *Biometrika*, 92(2):351–370, 2005.

Geert Verbeke. Linear mixed models for longitudinal data. In *Linear mixed models in practice*, pages 63–153. Springer, 1997.

Anton H Westveld and Peter D Hoff. A mixed effects model for longitudinal relational and network data, with applications to international trade and conflict. *The Annals of Applied Statistics*, 5(2A):843–872, 2011.

Russ Wolfinger and Michael O'connell. Generalized linear mixed models a pseudo-likelihood approach. *Journal of statistical Computation and Simulation*, 48(3-4):233–243, 1993.