Statistical Analyses of Tree-Based Ensembles

Dissertation zur Erlangung des Doktorgrades Dr. rer. nat. der Fakultät Statistik der Technischen Universität Dortmund

Vorgelegt von

Lena Schmid

geboren in Schwäbisch Gmünd

Dortmund, Dezember 2023

Amtierender Dekan:
Prof. Dr. Philipp Doebler

Gutachter:
Prof. Dr. Markus Pauly (Technische Universität Dortmund)
Prof. Dr. Andreas Groll (Technische Universität Dortmund)

Tag der Prüfung:
26.02.2024

Abstract

This thesis focuses on the study of tree-based ensemble learners, with particular attention to their behavior as a prediction tool for multivariate or time-dependent outcomes and their implementation for efficient execution. In particular, well-known examples such as Random Forest and Extra Trees are often used for the prediction of univariate outcomes. However, for multivariate outcomes, the question arises whether it is better to fit univariate models separately or to follow a multivariate approach directly. Our results show that the advantages of the multivariate approach can be observed in scenarios where there is a high degree of dependency between the components of the results. In particular, significant differences in the performance of the different Random Forest approaches are observed. In terms of predictive performance for time series, we are interested in whether the use of tree-based methods can offer advantages over traditional time series methods such as ARIMA, particularly in the area of data-driven logistics, where the abundance of complex and noisy data - from supply chain transactions to customer interactions - requires accurate and timely insights. Our results indicate the effectiveness of machine learning methods, especially in scenarios where data generation processes are layered with a certain degree of further complexity. Motivated by the trend towards increasingly autonomous and decentralized processes on resource-constrained devices in logistics, we explore strategies to optimize the execution time of machine learning algorithms for inference, focusing on Random Forests and decision trees. In addition to the simple approach of enforcing shorter paths through decision trees, we also investigate hardware-oriented implementations. One optimization is to adapt the memory layout to prefer paths with higher probability, which is particularly beneficial in cases with uneven splits within tree nodes. We present a regularization method that reduces path lengths by rewarding uneven probability distributions during decision tree training. This method proves to be particularly valuable for a memory architecture-aware implementation, resulting in a substantial reduction in execution time with minimal degradation in accuracy, especially for large datasets or datasets concerning binary classification tasks. Simulation studies and real-life data examples from different fields support our findings in this thesis.

Acknowledgments

First of all, I would like to thank my supervisor Professor Markus Pauly for his guidance, support and patience during my dissertation. I also thank Professor Andreas Groll for agreeing to be the second supervisor of my dissertation and the BMEL for their funding.

Further, I would also like to thank all my co-authors for their helpful comments, suggestions and discussions, which substantially contributed to this thesis.

My time as a PhD candidate was made truly special by my colleagues in both Ulm and Dortmund. Despite the challenges of switching universities and locations, their support and friendship made everything so much easier. Whether we were having discussions, attending conferences together, or simply enjoying coffee breaks, cakes, and 'Institutsstammtische', my colleagues made every moment a delight. I would like to thank them all for joining me on this journey.

Finally, I would like to thank my parents and my sisters for their encouragement, unconditional love, and support, which made of all this possible. Thank you for always being there for me!

"The only true wisdom is in knowing you know nothing."

– Socrates

Contents

List of the Main Publications

This cumulative thesis is mainly based on the following three manuscripts:

Article 1: Schmid, L., Gerharz, A., Groll, A. & Pauly, M. (2023). Tree-based ensembles for multi-output regression: Comparing multivariate approaches with separate univariate ones. *Computational Statistics & Data Analysis*, **179**, 107628, https://doi.org/10.1016/j.csda.2022.107628.
Copyright (2023), with permission from Elsevier.

Contribution of the author:
The author of this thesis had a leading role in the preparation and structuring of the manuscript. In addition, she mainly implemented the simulation studies as well as the analysis of the data examples.

Article 2: Schmid, L., Roidl, M. & Pauly, M. (2023). Comparing statistical and machine learning methods for time series forecasting in data-driven logistics - A simulation study. *arXiv preprint*, https://doi.org/10.48550/arXiv.2303.07139.

Contribution of the author:
The author of this thesis implemented extensive simulation studies and analyzed the data example under Prof. Pauly's guidance. She had a leading role in drafting and writing the paper, with helpful comments from all co-authors.

Article 3: Schmid, L., Biebert, D., Hakert, C., Chen, K.-H., Lang, M., Pauly, M. & Chen, J.-J. (2024). TREE: Tree Regularization for Efficient Execution. *arXiv preprint*, https://doi.org/10.48550/arXiv.2406.12531.

Contribution of the author:
The author of this thesis had the leading role in implementing the simulation study. She also prepared and structured the manuscript with the input from all co-authors.

Abbreviations

AI	Artificial Intelligence
ARIMA	Autoregressive Integrated Moving Average.
BR	Binary Relevance.
CART	Classification and Regression Trees.
CSPS	Corn Silage Processing Score.
Extra Trees	Extremely Randomized Trees.
HAR	Human Activity Recognition.
LGCM	Latent Growth Curve Models.
pRSL	Probabilistic Rule Stacking Learne.r
resKIL	Resource-Efficient Artificial Intelligence for Embedded Systems in Agricultural Technology.
XGBoost	Extreme Gradient Boosting.
$\boldsymbol{M}$	Matrix are denoted by bold symbols.
$\boldsymbol{M}^\top$	Transpose of a matrix or vector $\boldsymbol{M}$.
$\mathbb{N}$	Natural numbers.
$\mathbb{R}$	Real numbers.
$\mathbb{R}^+$	Set of all positive real numbers.
$\boldsymbol{x}$	Vectors are denoted by bold symbols.
$\mathbb{Z}$	Set of all integers.
$\|\|\mathbf{x}\|\|_2$	Euclidean norm of the vector $\mathbf{x}$.
$\mathbb{I}_{x\in A}$	Indicator function.
$\lfloor x \rfloor$	Floor function, i.e., the largest integer smaller than or equal to x.
$\mathcal{C}_A$	The collection of all possible cuts in the cell A.
$\mathcal{D}_n$	Training data set.
$\mathcal{T}_\tau$	The set of all nodes of a tree τ.

Part I

Introduction

1 Introduction

The rapid rise of digitization and increasing computing power have recently led to vast amounts of data being generated worldwide daily. For example, social media platforms such as Facebook and Instagram produce a huge amount of data through user interactions, posts, comments and shared content. The growing popularity of connected devices such as smart fitness trackers and home security systems also creates a wealth of data about user behavior and interactions with the devices.

As these data sources become increasingly complex, extracting relevant and accurate patterns for predictive models becomes a significant challenge. This complexity can result in mathematical problems where a traditional modeling strategy can be either too time-consuming or even too complex to provide satisfactory results. This has led in part to a change in approach to statistical modeling. Instead of modeling the complex data structure and searching for suitable analysis tools, the focus is on developing algorithms that are tailored to solve the problem. The development of such techniques has mainly emerged from the field of machine learning.

Formally, machine learning is defined as the study of systems that can learn from data without explicit programming. Following Mitchell (1997), it is assumed that a computer program is considered to learn data within a particular class of tasks and a defined performance measure if its capabilities in these tasks improve with additional data. Machine learning algorithms are in particular used to process classification or regression tasks. For both types of tasks, a learning algorithm requires two datasets: one that is used to train the algorithm and another for evaluating performance measures, which consists of previously unseen observations. If the outcomes of interest in the training set are known, the learning problem is categorized as supervised learning. In contrast, in scenarios where the outcome of interest is unknown during the training phase, the learning problem falls under unsupervised learning, specifically categorized as clustering. Both types of problems can encounter challenges in the modeling phase if the information in the features is of mixed type, including nominal, ordinal or metric scaled data. This variety of learning scenarios places high demands on the algorithms used in machine learning processes.

Ensemble learning is a possible solution to these challenges, especially in the context of supervised learning. In this machine learning technique, the predictions of multiple individual models are combined to make a more robust and accurate prediction. In particular, ensembles of decision trees are preferred over trees as a learning method due to the instability of decision trees and their computational efficiency in training and testing (Hastie et al., 2009). These tree-based ensemble learning methods often achieve top performance on many supervised learning problems (Dietterich, 2000; Fernández-Delgado et al., 2014; Zhang and Ma, 2012; Sagi and Rokach, 2018).

Two popular ensemble strategies, boosting (Freund et al., 1999) and bootstrap aggregation (bagging) (Breiman, 1996), have been extensively studied. Gradient tree boosting (Friedman, 2001) has gained great popularity due to its iterative learning process in which errors are corrected with each subsequent tree. Random Forests, a well-known tree-based ensemble method, uses bagging (bootstrap aggregation) to create an ensemble of decision trees. As shown in Figure 1.1[1] the number of publications citing the original Random Forests publication (Breiman, 2001) is increasing, highlighting the growing research interest in this approach.

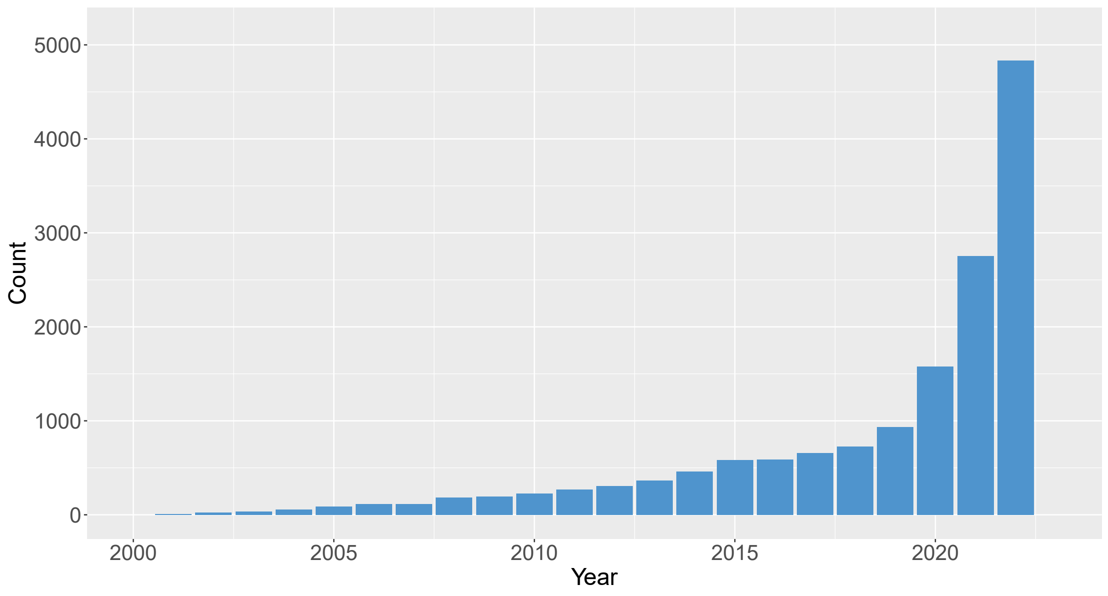


Figure 1.1: Number of publications with citation of the initial Random Forest paper of Breiman (2001) between 2000 and 2023.
© Copyright Clarivate 2023. All rights reserved.

[1]This figure was created based on data from the Web of Science in November 2023.

The aim of this thesis is to analyze tree-based ensembles from both performance and efficiency perspectives, with a focus on supervised learning. While these methods have proven successful in predicting univariate outcomes, e.g., (Wang et al., 2018; Liu and Wu, 2017; Everingham et al., 2016), extending them to multivariate or time-dependent outcomes presents a number of complex challenges and opportunities.

In multivariate analysis, the question arises: Should univariate models be fitted separately, or is a direct multivariate approach more appropriate? Univariate models treat each component of the outcome independently and capture individual relationships within the data. In contrast, a direct multivariate approach considers the joint distribution of outcomes, allowing for the capture of potentially more complex interactions or dependencies (De'Ath, 2002; Segal and Xiao, 2011). A special case of multivariate data is repeated measures (Crowder and Hand, 2017) that we treat within another research project, where the aim is to investigate the potential of multivariate tree-based models to predict learning trajectories. Note that repeated measures are essentially a subset of multivariate data with a temporal dimension that connects the domains of multivariate data and time series (Littell et al., 1998).

For time-dependent outcomes, tree-based ensemble methods have proven to be practical tools (Kane et al., 2014), offering the advantage of capturing nonlinear relationships and accommodating different data types. Despite their frequent use, the comparison of the performance of tree-based ensembles with traditional time series methods, which are generally faster to compute and easier to interpret, has not yet been widely discussed, especially not in the context of logistics, where time series forecasts are important at all levels of the supply chain (Syntetos et al., 2016). In addition to the importance of time series forecasts for logistics, there is also a trend towards increasingly autonomous and decentralized processes, such as autonomous robots (Krüger et al., 2020; Shamout et al., 2022). These often go hand in hand with the switch from a centralized control system to a decentralized one with less (computational) resources (Delfmann et al., 2018; Venkatapathy et al., 2017). In such scenarios with resource-constrained devices, optimizing the execution time of machine learning models is essential. In this thesis, the concept of regularizing the training process is presented as a way to achieve this goal.

This thesis is organized as follows: Chapter 2 describes the underlying statistical models and methods, such as the tree-based ensembles considered in our work. Chapter 3 provides a summary of the three main research articles underlying this dissertation, followed by Chapter 4, which offers a short overview of the results from additional projects undertaken during my doctoral studies. Chapter 5 contains a discussion of the results, as well as an outlook for future research. Finally, Part II contains the three articles summarized in Chapter 3.

2 Statistical Methods

2.1 Supervised Learning Problems

Supervised learning is a branch of machine learning that deals with extracting knowledge from datasets to make predictions when the outcome variable of interest is known (Hastie et al., 2009).

In supervised learning, an algorithm is trained with training data $\mathcal{D}_n$ consisting of pairs of input features $X \in \mathcal{X}$ and their corresponding known outcomes $Y \in \mathcal{Y}$, where both $\mathcal{X}$ and $\mathcal{Y}$ are metric spaces. More precisely, $\mathcal{D}_n$ is given by $\mathcal{D}_n = \{(X_i, Y_i) \in \mathcal{X} \times \mathcal{Y}, i = 1, \ldots, n\}$, where $n \in \mathbb{N}$ and (X_i, Y_i), $i = 1, \ldots, n$, are often assumed to be independent and identically distributed random variables with the same distribution as (X, Y).

A central object of supervised learning is modeling a functional mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$ between the features X and the corresponding outcomes Y. Since f is usually unknown, a method for determining an approximation $\hat{f}$ is used. The quality of this estimate is evaluated using loss functions $L(Y, \hat{f}(X))$ that measure the discrepancy between the predicted results and the actual values. The main goal is to determine $\hat{f}$ such that it minimizes the expected loss over all possible data points. In practice, however, the multivariate distribution of (X, Y) is usually unknown. Therefore, $\hat{f}$ is determined by optimizing the estimated generalization error instead. Since we are often interested in predicting unseen data, the original dataset $\mathcal{D}_n$ is usually split (several times) into a training and a test dataset, where the training dataset is then used to fit the model, while the test dataset is used to evaluate the model's performance. Various resampling techniques such as cross-validation and bootstrapping can be used to create training and test datasets from the original dataset (Hastie et al., 2009).

Classification One application of supervised learning is classification. In classification, the goal is to assign class labels to the input features. In other words, in this case, the support of $\mathcal{Y}$ is a countable set with finite cardinality. This type of learning finds extensive applications in various fields, e.g., Zhang et al. (2018); Angelini et al.

(2008); Conneau et al. (2017). In medical diagnosis, for example, the aim is to determine whether a disease is present or not based on patient data and symptoms (Gupta et al., 2019; Kourou et al., 2015). Another example is given by human activity recognition in warehouses, which I also studied during my thesis (Kirchhof et al., 2021b,a; Nair et al., 2023). Here, the aim is to classify typical activities of workers based upon video and sensor data.

Classification problems are often categorized based on the number of class labels. In binary classification, there are two classes, while in multi-class classification, there are more than two.

Regression In contrast to classification, regression problems deal with outcomes whose support is not finitely countable. Regression finds applications, for example, in predicting housing prices based on size and location (Park and Bae, 2015) and forecasting the annual growth of a country's gross domestic product based on historical data and economic indicators (Yoon, 2021). Further examples are concerned with the prediction of school performances based upon certain interventions and personal information (see Section 4.3) or forecasting of quantitative time series in logistics (Schmid et al., 2023b). If $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{Y} = \mathbb{R}^d$, $p, d \in \mathbb{N}$, a regression task is often described as follows

$$\mathbf{Y} = \mathbf{f}(\mathbf{X}) + \varepsilon,$$

where ε denotes the error variable. It is often assumed that the error variable has an expected value of zero and an existing covariance matrix. If $d > 1$, the task is referred to as a multi-output regression problem.

In this thesis, we explore both classification and regression problems. The first two articles focus on regression, while the third addresses classification. Moreover, my additional research projects conducted during my studies (Section 4) also deal with both tasks, particularly covering application fields such as logistics (Section 4.1), agriculture (Section 4.2) and social sciences (Section 4.3) and additional methods such as capsule networks (Section 4.4).

2.2 Tree-Based Methods

Tree-based methods divide the feature space into a series of regions and then fit simple models to each partition. They are conceptually relatively simple but often powerful (Hastie et al., 2009). We first describe the well-known CART method, which serves as the basis for more advanced models, including Random Forests, Extreme Randomized Trees and Extreme Gradient Boosting. These advanced models are ensemble learners combining multiple decision trees to improve prediction performance. In the following, we assume that $\mathbf{X} = (X^{(1)}, \dots X^{(p)})^\top \in \mathcal{X} = \mathbb{R}^p$ and $\mathcal{Y} \subset \mathbb{R}$.

2.2.1 CART

Classification and regression trees (CART) (Breiman, 2017) are a class of nonparametric algorithms that can handle classification and regression learning problems. As single decision trees, they fall into the category of weak learners (Zhou, 2012). Consequently, the predictions for $\mathbf{X}$ can vary if the training set $\mathcal{D}_n$ is changed. The general principle of CART is to partition the support of $\mathbf{X}$ into disjoint regions $R_1, \dots, R_J$, where $J \in \mathbb{N}$. These regions are often referred to as leaves. Each region $R_j, j = 1, \dots, J$, is assigned a constant value $c_j \in \mathcal{Y}$, resulting in the final CART model f^{CART} which is given by

$$f^{CART} : \mathbb{R}^p \to \mathcal{Y}, \quad \mathbf{x} \to \sum_{j=1}^{J} c_j \mathbb{1}_{\mathbf{x} \in R_j}.$$

CART grows the tree greedily in a top-down fashion using binary splits. It starts with the entire feature space of $\mathbf{X}$ at the root node. To define a split at a node, let $A \subset \mathcal{X} = \mathbb{R}^p$ be the corresponding region and let $N_n(A)$ denote the number of observations $(\mathbf{X}_i, \mathbf{Y}_i)$ in the training dataset $\mathcal{D}_n$ with $\mathbf{X}_i \in A$. A split is then given by $(i, z_i)^\top$, where $i \in \{1, \dots, p\}$ is the split dimension and z_i is the split value from the support of the i-th coordinate within A. The choice of a split is determined by optimizing a criterion dependent on the type of problem under consideration.

In the case of a regression problem ($\mathcal{Y} = \mathbb{R}$), the split is determined by maximizing the decrease in empirical variance, also known as L2 loss (Loh, 2011). Let $\mathcal{C}_A$ be the set of all possible splits for a given node and its corresponding region A. Then, the best split (i_n^*, z_n^*) based on the L2 loss is given by

$$\begin{aligned}(i_n^*, z_n^*) = \arg\max_{(i,z) \in \mathcal{C}_A} \Bigg\{ \frac{1}{N_n(A)} \Bigg(& \sum_{k=1}^{n} (Y_k - \overline{Y}_A)^2 \mathbb{1}_{\mathbf{X}_k \in A} \\ & - \sum_{k=1}^{n} \Big(Y_k - \overline{Y}_{A_L} \mathbb{1}_{X_k^{(i)} < z} - \overline{Y}_{A_R} \mathbb{1}_{X_k^{(i)} \geq z} \Big)^2 \mathbb{1}_{\mathbf{X}_k \in A} \Bigg) \Bigg\},\end{aligned}$$

where $A_L = \{\mathbf{x} \in A : x^{(i)} < z\}$, $A_R = \{\mathbf{x} \in A : x^{(i)} \geq z\}$ and $\overline{Y}_A$, $\overline{Y}_{A_L}$ and

$\overline{Y}_{A_R}$ denote the corresponding arithmetic means of the outcomes belonging to A, A_L and A_R, respectively. To save computing resources, the algorithm selects the middle of two consecutive observations as possible split values in the optimization step (Loh and Shih, 1997). Note that the definition of the best split is only valid if the i-th component of $\mathbf{X}$ is continuous. If it is categorical, the criterion slightly changes. The indicators are now given by $\mathbb{I}_{X_k^{(i)}=\chi_i}$ for all categories χ_i of the i-th component and $A_L = \{\mathbf{x} \in A : x^{(i)} \neq \chi_i\}$, $A_R = \{\mathbf{x} \in R : x^{(i)} = \chi_i\}$, see Breiman (2017).

The L2 loss is not suitable for classification problems. Here, impurity measures are used to evaluate splits. These measures assess the purity of a region, indicating the extent to which it contains observations of a single class label. The most commonly used impurity measures Q_K for $K \in \mathbb{N}$ classes and a region A are (Hastie et al., 2009):

(i) Misclassification error: $Q_K(\hat{\mathbf{p}}(A)) = 1 - \max_{k=1,\dots,K} \hat{p}_k(A)$,

(ii) Gini index: $Q_K(\hat{\mathbf{p}}(A)) = \sum_{k=1}^{K} \hat{p}_k(A)\,(1 - \hat{p}_k(A))$,

(iii) Cross entropy: $Q_K(\hat{\mathbf{p}}(A)) = -\sum_{k=1}^{K} \hat{p}_k(A) \log{(\hat{p}_k(A))}$,

where $\hat{\mathbf{p}}(A) = (\hat{p}_1(A), \dots, \hat{p}_K(A))^\top$ and $\hat{p}_k(A)$ denotes the proportion of observations that fall into A with the class label r_k, $1 \leq k \leq K$. The optimum split of such an impurity measure is then given by (Hastie et al., 2009)

$$\begin{aligned}(i_n^*, z_n^*) = \arg\max\nolimits_{(i,z)\in\mathcal{C}_A} \Bigg\{ & Q_K(\hat{\mathbf{p}}(A)) - \frac{N_n(A_L)}{N_n(A)} Q_K(\hat{\mathbf{p}}(A_L)) \\ & - \frac{N_n(A_R)}{N_n(A)} Q_K(\hat{\mathbf{p}}(A_R)) \Bigg\}.\end{aligned}$$

After applying these splits recursively until a stopping criterion is fulfilled, the goal is to appropriately estimate the constant values (regression) respectively class labels (classification) $\{c_1, \dots, c_J\}$. For regression problems, the region R_j assigned estimated constant $\hat{c}_j$ is given by $\hat{c}_j = \sum_{i:\mathbf{X}_i \in R_j} Y_i$, while in classification problems majority voting is performed in each region to obtain the class label, i.e., $\hat{c}_j = \text{mode}_{i:\mathbf{X}_i \in R_j}(Y_i)$.

Parameters that control the complexity of the tree are often used as stopping criteria. Examples include limiting the maximum tree depth, specifying a minimum number of samples in each leaf or using a minimum improvement threshold for the selected splitting criterion (Therneau and Atkinson, 1997). If a CART tree grows too deep, the number of regions J is very large and overfitting can occur, i.e., the tree fits the training data too well but has difficulties in correctly predicting new, unseen data. Ways to reduce the effect of overfitting are pruning, the subsequent reduction and optimization of the trees, or bagging, where a bootstrap sample of the training data is used when creating the trees (Hastie et al., 2009).

2.2.2 Random Forest

Random Forest is an ensemble learning method for regression and classification tasks based on the construction of multiple decision trees. Different randomness sources are introduced into the tree construction process to obtain various decision trees from a single dataset. This randomness can include randomizing the dataset itself (Breiman, 1996), the feature set (Ho, 1998) or a combination of both (Breiman, 2001; Cutler and Zhao, 2001).

The most popular Random Forest algorithm (Breiman, 2001) aims to eliminate the prediction deficiencies of CARTs (Breiman, 2017), such as overfitting and the strong dependence on the training dataset. By growing multiple trees in parallel, the biases and variance of the model are reduced simultaneously (Breiman, 2001). The main principles of Random Forests are bagging and feature sub-selection. This means a random selection of training data is used when creating an unpruned tree. During construction, only a small and fixed number of randomly selected m_{try} features are used as split candidates instead of all available features. The same split criteria are used as for CART, i.e., the L2 loss for the regression and the Gini index for the classification. However, many other splits, such as the absolute deviation from the median, can also be used. To make predictions for new data, Random Forest aggregates results from multiple trees by calculating the arithmetic mean for regression tasks and applying majority voting for classification.

The most essential hyperparameters for the Random Forests are:

(i) The number of decision trees $B \in \mathbb{N}$ in the ensemble.

(ii) The number of pre-selected features m_{try} for conducting splits.

(iii) The number of sampled data points in each tree $a_n \in \{1, \dots, n\}$.

(iv) The sampling strategy $\mathcal{S}$.

(iv) Various tree complexity parameters that can be used as stopping criteria, such as the minimum number of observations that each leaf node should contain $n_{min} \in \{1, \dots, a_n\}$ or the number of leaves in each tree $\max_{nodes} \in \{1, \dots, a_n\}$.

The choice of hyperparameters is task-dependent and can be determined through techniques like cross-validation or random search (Bartz et al., 2023). Several packages already implement automatic tuning procedures for Random Forests (Kuhn, 2008; Bischl et al., 2023). However, it is known that Random Forests often provide good results in the default settings (Probst et al., 2019a; Fernández-Delgado et al., 2014). The number of trees in a forest is a parameter that is usually not tuned as it is known that more trees are better (Díaz-Uriarte and Alvarez de Andrés, 2006; Scornet, 2017). The common default value is $B = 500$ or $B = 1,000$ (Probst et al., 2019b).

One of the central hyperparameters is the number of pre-selected features $m_{try} \in \{1, \dots, p\}$. Lower values of m_{try} lead to more different and less correlated trees, enhancing stability during aggregation. However, lower m_{try} values can also cause the trees to perform worse on average, as they may be based on a small suboptimal subset of randomly selected candidate features (Probst et al., 2019b). Breiman (2001) proposes to use $m_{try} = \lfloor \frac{p}{3} \rfloor$ for regression and $\lfloor \sqrt{p} \rfloor$ for classification as default values, which is still the standard choice in many software implementations such as the fast `ranger` implementation in `R` (Wright and Ziegler, 2017). There are often two options for the sampling strategy: Sampling with or without replacement. By default, the number of sampled data points in each tree a_n is set to n, which means resampling is performed with replacement (Hastie et al., 2009). However, in theoretical investigations sampling without replacement is often assumed as it is easier to handle (Scornet et al., 2015; Ramosaj, 2020). The default value for n_{min} is five for regression and one for classification (Díaz-Uriarte and Alvarez de Andrés, 2006). Typically, trees are fully grown, i.e., $\max_{nodes} = a_n$, or in the case of sampling without replacement, $\max_{nodes} = \lfloor 0.632 a_n \rfloor$ (Wright and Ziegler, 2017).

Several modifications and extensions of the original Random Forest algorithm have been developed, encompassing alterations in various aspects, including the random feature selection process, bagging procedures and the split criteria (Meinshausen, 2006; Clémençon et al., 2013; Ishwaran et al., 2008). In our research, we investigate the use of random forests for multi-output regression (Schmid et al., 2023a) and explore the use of regularization to improve performance in classification tasks (Schmid et al., 2024). Furthermore, we investigate its application in different domains such as social sciences (Section 4.3) and data-driven logistics (Schmid et al., 2023b).

2.2.3 Extremely Randomized Trees

Extremely Randomized Trees (Extra Trees) (Geurts et al., 2006) is an ensemble method that aims to obtain trees that are more decorrelated than trees from the Random Forest approach by introducing an additional source of randomness into the method. The tree construction process is very similar to that of Random Forests, with some differences. In the original Extra Trees algorithm, no bagging is performed, i.e., each tree is built using the entire available training dataset. Instead of computing each feature's locally optimal split value through a greedy search, Extra Trees randomly selects a set of split values (Geurts et al., 2006).

Most of the primary hyperparameters are the same as for Random Forests. An additional hyperparameter is the number of random split values at each node, denoted as n_{splits}. Geurts et al. (2006) investigate the influence of this parameter in a benchmark study. Typically, n_{splits} is set to one (Simm et al., 2014).

Although bagging is not included in the original Extra Trees method, some implementations, such as `ranger` (Wright and Ziegler, 2017), have integrated it. Moreover, Simm et al. (2014) has extended the method to multi-task learning. In Schmid et al. (2023a), we investigated the use of Extra Trees for multi-output regression and compared their performance with the extension of Simm et al. (2014).

2.2.4 Boosting

Boosting is a powerful class of learning algorithms that combine multiple sequentially constructed weak learners. In general, a boosting model f^{boost} has the form

$$f^{boost}(\mathbf{x}) = \sum_{m=1}^{M} f_m(\mathbf{x}),$$

where $f_1, \dots f_M, M \in \mathbb{N}$ are some weak learners, often referred to as base functions in the following. Friedman et al. (2000) showed that boosting can be viewed as an additive expansion in a set of elementary functions. Therefore, f^{boost} can be rewritten as

$$f^{boost}(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m h(\mathbf{x}; \mathbf{a}_m),$$

where $\alpha_1, \dots \alpha_M \in \mathbb{R}$ are expansion coefficients and $h : \mathbb{R}^d \to \mathbb{R}$ with $\mathbf{x} \to h(\mathbf{x}; \mathbf{a})$ are simple base functions characterized by a set of parameters $\mathbf{a} \in \mathbb{R}^D$. For example, if CART learners are chosen as base functions h, the parameters $\mathbf{a}_1, \dots, \mathbf{a}_M$ refer to the parameterization of the decision trees.

To fit such a model, the base functions are selected and then their parameters $\{\alpha_m, \mathbf{a}_m\}_{m=1}^{M}$

are determined by minimizing a loss function L averaged over the training dataset $\mathcal{D}_n$:

$$\{\alpha_m, \mathbf{a}_m\}_{m=1}^M = \arg\min_{\{\tilde{\alpha}_m, \tilde{\mathbf{a}}_m\}_{m=1}^M} \frac{1}{n} \sum_{i=1}^n L\left(Y_i, \sum_{m=1}^M \tilde{\alpha}_m h(\mathbf{X}_i, \tilde{\mathbf{a}}_m)\right).$$

As this process can be computationally intensive, a forward stagewise additive modeling approach is often used (Friedman et al., 2000). Here, the optimization is performed by sequentially adding new basis functions to the additive expansion without adjusting the parameters and coefficients of those that have already been added. At each iteration m, $m = 1, \ldots, M$, we determine

$$(\alpha_m, \mathbf{a}_m) = \arg\min_{\alpha, \mathbf{a}} \sum_{i=1}^n L(Y_i, g_{m-1}(\mathbf{X}_i) + \alpha h(\mathbf{X}_i, \mathbf{a})), \tag{2.1}$$

with $g_0(\mathbf{x}) = 0$ and $g_{m-1}(\mathbf{x}) = \sum_{i=1}^{m-1} \alpha_i h(\mathbf{x}; \mathbf{a}_i)$, which results in

$$g_m(\mathbf{x}) = g_{m-1}(\mathbf{x}) + \alpha_m h(\mathbf{x}; \mathbf{a}_m),$$

see Hastie et al. (2009) for details. The understanding of boosting algorithms as numerical optimization methods in the function space was developed by Breiman (1998, 1999). Based on this, more general boosting algorithms were developed to optimize any differentiable loss function (Friedman, 2001; Mason et al., 1999). They employed functional gradient descent to solve Equation (2.1). The negative gradient for a differentiable loss function L based on the training dataset is given by

$$-\gamma_m(\mathbf{x}_i) = -\left[\frac{\partial L(Y_i, g(\mathbf{X}_i))}{\partial g(\mathbf{X}_i)}\right]_{g(\mathbf{x}) = g_{m-1}(\mathbf{x})}.$$

Note that the negative gradient is only available at the data points given in $\mathcal{D}_n$. Thus, to generalize to other points, we approximate the negative gradient $-\gamma_m(\mathbf{x})$ by $h(\mathbf{x}; \mathbf{a}_m)$, where

$$\{\alpha_m, \mathbf{a}_m\} = \arg\min_{\alpha, \mathbf{a}} \sum_{i=1}^n \left(-\gamma_m(\mathbf{X}_i) - \alpha h(\mathbf{X}_i; \mathbf{a})\right)^2.$$

The step length ρ_m to take in this step direction is then determined using a line search of the form

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^n L(Y_i, g_{m-1}(\mathbf{X}_i) + \rho h(\mathbf{X}_i; a_m)).$$

Friedman (2001) additionally introduces a regularization parameter $0 < \nu \leq 1$, which is multiplied to the step length at each iteration. The factor is also referred to as the learning rate. The step taken at each iteration m is given by

$$\nu \rho_m h(\mathbf{x}; \mathbf{a}_m).$$

The gradient boosting method for arbitrary supervised learning problems is outlined in Algorithm 1. Instead of initializing $g_0(\mathbf{x}) = 0$ as in Equation (2.1), the optimal constant

Algorithm 1 Gradient Boosting

```
1: **Input:** Training dataset $\mathcal{D}_n$, number of iteration $M$, loss function $L$, learning rate $\nu$.
2: **Initialize:** $f_0(x) = \arg\min_\rho \sum_{i=1}^n L(Y_i, \rho)$;
3: **for** $m = 1, \dots, M$ **do**
4:     Compute for each $i = 1, \dots, n : -\gamma_m(\mathbf{x}_i) = -\left[\frac{\partial L(Y_i, g(\mathbf{X}_i))}{\partial g(\mathbf{X}_i)}\right]_{g(\mathbf{x})=g_{m-1}(\mathbf{x})}$;
5:     Set $\{\alpha_m, \mathbf{a}_m\} = \arg\min_{\alpha,\mathbf{a}} \sum_{i=1}^n [-\gamma_m(\mathbf{X}_i) - \alpha \cdot h(\mathbf{X}_i; \mathbf{a})]^2$;
6:     Set $\rho_m = \arg\min_\rho \sum_{i=1}^n L(Y_i, g_{m-1}(\mathbf{X}_i) + \rho h(\mathbf{X}_i; \mathbf{a}_m))$;
7:     Update: $g_m(\mathbf{x}) = g_{m-1}(\mathbf{x}) + \nu \rho_m h(\mathbf{x}; \mathbf{a}_m)$
8: **end for**
9: **Output** $g_M(\mathbf{x})$
```

model is initialized (line 2) (Hastie et al., 2009).

A well-known implementation of gradient boosting is Extreme Gradient Boosting (XG-Boost) proposed by Chen and Guestrin (2016). It uses the Newton step instead of the gradient step, which, according to Friedman et al. (2000), can reduce susceptibility to overfitting and lead to improved performance. In this approach, in addition to the negative gradients, the second-order derivatives are also taken into account, which are given by

$$\delta_m(\mathbf{x}_i) = \left[\frac{\partial^2 L(Y_i, g(\mathbf{X}_i))}{\partial g(\mathbf{X}_i)^2}\right]_{g(\mathbf{x})=g_{m-1}(\mathbf{x})}.$$

The Newton step is found by solving:

$$\mathbf{a}_m = \arg\min_{\mathbf{a}} \sum_{i=1}^n \left(\gamma_m(\mathbf{X}_i) h(\mathbf{X}_i, \mathbf{a}) + \tfrac{1}{2}\delta_m(\mathbf{X}_i) h(\mathbf{X}_i, \mathbf{a})^2\right),$$

which amounts to a weighted least-squares regression problem. The step length is then given by $\nu h(\mathbf{x}; \mathbf{a}_m)$.

Furthermore, XGBoost incorporates randomization and regularization techniques to reduce overfitting while increasing training speed. In particular, when using CART as the base learning method, XGBoost applies several strategies to speed up training. In particular, it addresses the computational complexity of identifying the optimal split, a particularly time-consuming aspect of decision tree construction algorithms. Instead of evaluating all possible candidate splits, XGBoost employs a method based on percentiles of the data, where only a subset of candidate splits is tested and their gain is computed using aggregated statistics (Chen and Guestrin, 2016).

In general, the hyperparameter for XGBoost can be divided into the following categories:

(i) General boosting parameters, including the number of iterations M and the learning rate ν.

(ii) Base learner dependent parameters.

Friedman (2001) shows that smaller values of ν tend to improve the generalization performance. Note, however, that a reduction of ν usually requires more iterations. Consequently, the reduction of ν is associated with an increased computational effort. In fact, Bühlmann and Hothorn (2007) mention that too large a choice of iterations M can lead to a slow overfitting problem. When trees are used as base learners, the additional hyperparameters are very similar to those discussed for the previously presented methods and control the complexity of the individual trees.

Since XGBoost has demonstrated state-of-the-art performance across diverse problem domains (Chen and Guestrin, 2016), we have investigated its effectiveness in time series prediction (Schmid et al., 2023b). Some mathematical background for the latter is provided, assuming a basic knowledge of time series analysis. For a more detailed description, the reader is referred to Box et al. (2015) and Brockwell and Davis (2002).

2.3 Time Series Forecasting

A time series process is given by a sequence $\{y_t, t \in T\}$ of realization of real-valued variables Y_t, where t belongs to an at most countable index set T, typically $T = \mathbb{N}$ or $T = \mathbb{Z}$, with equidistant elements (Hamilton, 2020). The goal of time series prediction is to approximate a future variable y_{t+H} by a function

$$\hat{y}_{t+H} = v(y_t, y_{t-1}, \ldots)$$

of the observed values up to the present time t, where $H > 0$ is the so-called prediction horizon. Typically, this function v is chosen from a class of functions such that the prediction error is minimal.

Commonly used for modeling time series, Autoregressive Integrated Moving Average (ARIMA) processes (Box et al., 2015) combine autoregressions (AR) and moving averages (MA) with differencing to make the time series stationary. The ARIMA(q_1, q_2, q_3) model is defined as

$$\varphi(\Delta)(1 - \Delta^{q_3})y_t = \theta(\Delta)\varepsilon_t,$$

where $\{\varepsilon_t\}$ is a white noise process, Δ the backshift operator, $\varphi(z)$ and $\theta(z)$ are polynoms of order q_1 and q_2, respectively.

To use this approach for prediction, appropriate values for the parameters have to be estimated. This typically involves a two-step procedure: determining the model order parameters q_1, q_2, q_3 and then estimating the coefficients of the polynomials $\varphi(z), \theta(z)$ based on these parameters (Hyndman and Athanasopoulos, 2018). The estimation methods presented in the following are based on the approach of Hyndman and Khandakar (2008) which is implemented in the R-package `forecast`. The differentiation parameter q_3 is determined through successive unit root tests (Kwiatkowski et al., 1992). The procedure consists of testing the data for a unit root. If the result is significant, the differenced data is tested for a unit root, continuing until a non-significant result is obtained. The values of q_1 and q_2 are selected by minimizing a model information criterion (Konishi and Kitagawa, 2008) such as AIC, AICc or BIC after differentiating the data q_3 times. Instead of considering every possible combination of q_1 and q_2, the algorithm uses a stepwise search to explore the model space. Once the model order parameters, i.e., the values of q_1, q_2 and q_3, are identified, the coefficients of the polynomials $\varphi(z)$ and $\theta(z)$ are estimated using maximum likelihood estimation.

Seasonality in a time series refers to recurring patterns or fluctuations that exhibit regular and predictable behavior over fixed time intervals m. A seasonal model contains coefficients of order q_1 and q_2 and includes observations with lags of multiples of m. The order of these terms is denoted by q_1^s for the seasonal AR part and q_2^s for the seasonal MA part. The polynomials $\Phi(x)$ and $\Theta(x)$ are constructed similarly to their non-seasonal counterparts, resulting in a seasonal ARIMA$(q_1, q_2, q_3)(q_1^s, q_2^s, q_3^s)_m$ model (Shumway and Stoffer, 2017), which is given by

$$\Phi(\Delta^m)\phi(\Delta)(1 - \Delta^m)^{q_3^s}(1 - \Delta^{q_3})y_t = \Theta(\Delta^m)\theta(\Delta)\varepsilon_t.$$

Exponential smoothing offers an alternative approach for time series forecasting. By iteratively updating weighted averages of historical observations, exponential smoothing adapts to changing trends and captures seasonality. The TBATS model (De Livera et al., 2011), which is based on exponential smoothing, is designed to model time series with multiple trigonometric seasonalities. It uses a combination of Fourier terms with an exponentially smoothing state space model and a Box-Cox transformation (Box and Cox, 1964).

To be more precise, a TBATS model at time t with S seasonal pattern is given by

$$\begin{aligned}
y_t^{(\zeta)} &= \begin{cases} \frac{y_t^{\zeta}-1}{\zeta}, & \text{if } \zeta \neq 0 \\ \log y_t & \text{if } \zeta = 0, \end{cases} \\
y_t^{(\zeta)} &= \ell_{t-1} + \psi b_{t-1} + \sum_{i=1}^{S} s_t^{(i)} + w_t, \\
\ell_t &= \ell_{t-1} + \psi b_{t-1} + \eta w_t, \\
b_t &= (1-\psi) b + \psi b_{t-1} + \beta w_t, \\
w_t &= \sum_{i=1}^{q_1'} \varphi_i' w_{t-i} + \sum_{i=1}^{q_2'} \theta_i' \varepsilon_{t-i} + \varepsilon_t,
\end{aligned}$$

where ζ is the Box-Cox transformation parameter, ℓ_t is the local level at time t, b is the long-run trend, b_t the short-run trend at time t, w_t denotes an ARIMA$(q_1', q_2', 0)$ process and ε_t is Gaussian white noise process with zero mean and constant variance σ^2. The smoothing parameters are given by η and β. The local level is incorporated to capture short-term variations or fluctuations in the data (Brockwell and Davis, 2002). In addition, a damped trend (Gardner Jr. and McKenzie, 1985) with a damping parameter ψ ensures predictions of future values of the short-run trend b_t, which refers to fluctuations or patterns associated with periodic or cyclical movements that repeat over a short time horizon, converge to the long-run trend b (Gardner Jr. and McKenzie, 1985). This long-run trend represents the general tendency of the time series. For the seasonal component's process $s_t^{(i)}$, $i = 1, \ldots, S$, a trigonometric representation based on Fourier series is used (West and Harrison, 2006; Harvey, 1990):

$$\begin{aligned}
s_t^{(i)} &= \sum_{j=1}^{\kappa_i} s_{j,t}^{(i)}, \\
s_{j,t}^{(i)} &= s_{j,t-1}^{(i)} \cos(\omega_j^{(i)}) + s_{j,t-1}^{*(i)} \sin(\omega_j^{(i)}) + \xi_1^{(i)} w_t, \\
s_{j,t}^{*(i)} &= -s_{j,t-1}^{(i)} \sin(\omega_j^{(i)}) + s_{j,t-1}^{*(i)} \cos(\omega_j^{(i)}) + \xi_2^{(i)} w_t,
\end{aligned}$$

where $\omega_j^{(i)} = 2\pi \frac{j}{m_i}$, $\xi_1^{(i)}$ and $\xi_2^{(i)}$ are smoothing parameters and κ_i denotes the number of summands in the Fourier series for the i-th seasonal component.

Selecting the final model in the TBATS algorithm involves a systematic evaluation of various model specifications. TBATS explores a spectrum of alternatives by fitting multiple models with distinct characteristics, including the consideration of:

(i) The inclusion or exclusion of the Box-Cox transformation.

(ii) The presence or absence of trend components.

(iii) The incorporation or omission of trend damping effects.

(iv) The utilization or non-utilization of ARIMA$(q_1', q_2', 0)$ processes for modeling residuals.

(v) The exploration of non-seasonal models.

(vi) The assessment of different quantities of harmonics to effectively model seasonal effects.

The final model is chosen based on the model information criterion AIC and is implemented in the R-package `tbats` (De Livera et al., 2011).

In addition to traditional time series approaches such as ARIMA and TBATS, machine learning methods are increasingly being used for time series forecasting. These methods are able to capture complex patterns and non-linear relationships and have proven their effectiveness in various areas (Kane et al., 2014; Khaidem et al., 2016; Dudek, 2015; Wang et al., 2018). In finance, for example, XGBoost approaches are used to predict crude oil prices (Gumus and Kiran, 2017) and electricity loads on the Australian energy market (Abbasi et al., 2019). In retail, XGBoost approaches are being explored for predicting store sales (Zhang et al., 2021), while in transportation they are used to predict NYC cab travel time (Huang et al., 2020).

Random Forest models have been applied to generate hour-ahead wind power forecasts (Lahouar and Slama, 2017). In the domain of online retailing, Random Forest approaches are employed to model real-time delivery time forecasts (Salari et al., 2022) and for predicting product demand for grocery items (Vairagade et al., 2019). Notably, modifications of Random Forest tailored for time series, particularly in resampling strategies, have been explored (Härdle et al., 2003). Goehry et al. (2021) have investigated the use of different block bootstrap strategies, preserving dependency structures by grouping data into blocks and incorporated their findings into the `rangerts` package (Goehry et al., 2017). Fokam (2022) has examined the use of an AR-sieve bootstrap (Bühlmann, 1997) that applies residual resampling by fitting an AR process to the data. In a simulation study, Fokam (2022) compared the block bootstrap strategy with the standard one. The results indicate that the block bootstrap method may not offer a statistically significant advantage over the classic bootstrap method.

In Schmid et al. (2023b), we conduct a comparative analysis in which we evaluate the performance of Random Forest and XGBoost compared to traditional time series methods in the context of data-driven logistics, where robust forecasting methods are essential for optimizing inventory management, ensuring on-time deliveries and optimizing supply chains (Syntetos et al., 2016).

3 Summary of the Main Articles

3.1 Article 1: Tree-based ensembles for multi-output regression: Comparing multivariate approaches with separate univariate ones

We consider the case of multi-output regression described in Section 2.1, where our goal is to establish a functional relationship between a multivariate outcome $\mathbf{Y} \in \mathbb{R}^d, d \geq 2$ and some features $\mathbf{X} \in \mathbb{R}^p$. The motivation behind such multivariate analyses arises from the possibility of capturing dependencies between outcomes, which may lead to improved prediction performance compared to separate univariate analyses. While the need to develop valid multivariate methods has already been recognized in inferential statistics, there is still a gap in comprehensive studies that fully exploit the potential of multivariate machine learning regression methods for prediction. Our focus is on tree-based ensemble methods and we aim to fill this gap by comparing the predictive accuracy of separate univariate analyses with a simultaneous multivariate analysis using exhaustive simulation.

To be more precise, we consider Random Forests and Extra Trees. The multivariate extension of these methods aligns with established work by De'Ath (2002); Segal and Xiao (2011). We determine the impurity of a node τ using the multivariate L2 loss

$$L_2(k) = \sum_{j:\mathbf{Y}_j \in \tau} (\mathbf{Y}_j - \overline{\mathbf{Y}}_t)^\top (\mathbf{Y}_j - \overline{\mathbf{Y}}_t),$$

where $\mathbf{Y}_t$ is the arithmetic mean of the outcome vector at node τ. In addition, we also consider the multivariate L1 loss, which is defined by

$$L_1(k) = \sum_{j:\mathbf{Y}_j \in \tau} \sum_{i=1}^{d} \left| Y_j^i - \tilde{Y}_t^i \right|,$$

where $\tilde{\mathbf{Y}}_t = (\tilde{Y}_t^1, \ldots \tilde{Y}_t^d)^\top$ is the vector of marginal medians of the outcome at node τ.

For comparison, we use the extension of Extra Trees for multi-task learning (Simm et al., 2014). To ensure a fair evaluation with respect to computational time, we implemented our tree construction algorithm, except for the extra trees for multi-task, for which we used the R-package `extraTrees` (Simm et al., 2014).

Our simulation study employs runtime and mean squared error as evaluation metrics to assess performance. We focus on a three-dimensional outcome with ten real-valued features, each specified with different distributions, dependencies and underlying models. Specifically, we explore three distinct dependency structures among the features: complete independence, weak dependence and strong dependence. Following Loh (2002), we define six relationships between outcomes and some of the features. For each relationship type, we consider models where all outcomes are generated by the same data generating process, resulting in high correlation. Additionally, we examine models where the data generating process remains constant, but the features used in the process to compute outcomes differ. Moreover, we introduce models where the data generating process varies for each component of the outcome. We also cover a spectrum of dependency structures in the outcome, ranging from complete independence to high dependence. To ensure comprehensive analysis, we consider three different sample sizes and conduct 1,000 simulation runs.

The results indicate that when comparing the predictive accuracy of multivariate approaches with their univariate counterparts, the latter have performance advantages in scenarios where the data generating processes for the outcome components are different. However, in all other simulation configurations, the multivariate approaches have at least similar or even better performance than their univariate counterparts. In particular, when comparing Random Forests, the multivariate approaches are strongly superior to the univariate approaches in certain scenarios. Furthermore, all methods are sensitive to different dependency structures within the features, with similar effects observed for univariate and multivariate approaches. As for the two algorithmic approaches, either the Random Forest or the Extra Tree approaches have outperformed the others. With regard to runtime, multivariate approaches demonstrate a clear runtime advantage over univariate approaches with comparable implementation. There are only minor differences in runtime and performance results concerning the multivariate loss function used.

We also compare the methods on five datasets from the UCI repository (Dua and Graff, 2017). These data analyses show that the multivariate approaches can improve performance when considering multivariate outcomes, especially for the Extra Trees. For the Random Forests, the multivariate counterpart only improved performance in a few of the considered datasets.

To sum up, we propose the use of multivariate extensions of tree-based ensemble methods. In an extensive simulation study, we compare these with their univariate counterpart and a multi-task extension of Extra Trees. Finally, all methods are analyzed on five real-life multi-output regression datasets of varying complexity, considering performance and runtime.

3.2 Article 2: Comparing statistical and machine learning methods for time series forecasting in data-driven logistics - A simulation study.

Accurate forecasts derived from historical data play a central role in data-driven logistics. They enable informed decisions, ensure timely delivery or minimize disruption in the dynamic logistics domain. Therefore, forecasting methods are essential for strategic planning in various logistics areas, including warehousing, transportation and supply chain management (Syntetos et al., 2016). While time series models are widely used for forecasting, the increasing prevalence of large and complex data sets in logistics has led to a growing interest in machine learning methods for demand forecasting. To address the ambiguity often faced when choosing the most appropriate forecasting method, we conduct a comprehensive comparison of the forecasting performance of time series models and machine learning methods. More precisely, different forecasting methods are evaluated in terms of their out-of-the-box performance on a large number of simulated time series that are important for logistics.

Our study focuses on (seasonal) ARIMA and TBATS as time series models. The former model is one of the best known in time series forecasting and is often used as a benchmark model (Zhang et al., 2001; Al-Saba and El-Amin, 1999). In addition, TBATS models contain various approaches commonly used in forecasting, such as the Box-Cox transformation or exponential smoothing (De Livera et al., 2011). Since the use of machine learning approaches is already widely explored (Zhang and Qi, 2005; Siami-Namini et al., 2018), we decided to focus on tree-based ensemble learners. In particular, we consider XGBoost, which is known for its predictive power (Chen and Guestrin, 2016) and Random Forests, one of the most popular tree-based ensembles that is quite robust to hyperparameter tuning (Probst et al., 2019a). To provide a baseline, a naive approach utilizing the last observation of the time series as prediction is also considered.

We investigate twelve distinct data generating processes, each presenting different challenges. These processes range from minimally nonlinear, such as AR, to strongly nonlinear relationships between past and current values, such as seasonal AR. To increase the complexity of the time series, we add a jump process and a random walk to each data generating process. The jump process leads to sudden regime changes, while the random walk introduces noise into the data. Our study encompasses four scenarios: (1) the data generating process with no added complexity, (2) the process overlaid with the jump process, (3) the process superposed with random noise and (4) the process subject to both the jump process and the random noise.

In addition, we apply the methods to a real-life dataset from logistics and to datasets generated from two queuing models, M/M/1 and M/M/2. These models are commonly used to analyze the dynamics of queues with one (M/M/1) or two (M/M/2) servers, taking into account factors such as the expected waiting time, the expected number of customers in the queue and the expected server utilization.

For each setting, we generate a time series of length n from the respective data generation processes with $n \in \{100, 500, 1,000\}$. In our simulation study, we use the mean absolute percentage error and the mean squared error as evaluation measures for the predictive power since these metrics are widely used in forecasting time series in logistics (Kuhlmann and Pauly, 2023).

In our simulation study, we employ a sliding window (Dietterich, 2002) approach for forecasting using machine learning algorithms. This approach involves moving a fixed-sized window over the time series data, using the data within the window as input for prediction at each step. Furthermore, we explore the impact of time series differentiation on the prediction performance of machine learning algorithms. This is motivated by the aim of making the time series more stationary through differentiation.

When analyzing the simulation results, we find that in all simulation settings the Random Forests outperform the XGBoost approaches. In general, either Random Forests or time series methods show superior performance compared to the other approaches. In particular, Random Forests perform better in scenarios with queueing models and in cases where a Poisson process overlaps with the data generating processes. In all other simulation settings, however, Random Forests perform comparably or slightly worse than time series methods. Regarding the effect of data differentiation on the performance of the two machine learning methods, we observe similar patterns. Differentiation improved performance, especially in queuing scenarios and in situations where additional complexity is introduced into the data generation process. Introducing a jump process into the data generation process leads to a significant drop in performance for all methods and settings while overlaying the data with additional noise can lead to differences in performance depending on the setting.

To sum up, we investigate the forecasting performance of several tree-based ensembles and time series models in an extensive simulation study. Based on these simulations, we were able to derive recommendations for practitioners. Furthermore, we analyzed a real-life dataset on demand forecasting from logistics.

3.3 Article 3: TREE: Tree Regularization for Efficient Execution.

The optimization of machine learning model execution time on resource-constrained edge devices has been widely studied (Murshed et al., 2021; David et al., 2021). A central aspect in this context is the limited availability of energy and time for executing inference due to resource constraints. A well-known strategy to speed up inference is to reduce the model size, e.g., by limiting the maximum tree depth. While this leads to a significant reduction in execution time as fewer computations are required, it often comes at the expense of accuracy. Another effective approach is using CPU cache behavior, especially in the area of tree-based ensembles (Chen et al., 2022; Tabanelli et al., 2022). Chen et al. (2022) have shown that a cache-friendly reordering of the nodes improves execution time. In their approach, the split probabilities of the nodes are used to determine the new order in memory, with unequal splits being beneficial as they lead to nodes that are accessed more frequently.

We propose an alternative method to reduce the overall model size effectively. Our approach involves optimizing the decision tree construction by introducing a regularization term into the split criterion. More precisely, the regularization term Ξ_τ, defined by

$$\Xi_\tau = 1 - \frac{|N_n Mu(A_L) - N_n(A_R)|}{N_n(A)},$$

for a given node τ and the corresponding cell A serves as a control parameter to establish a trade-off between tree size and prediction accuracy. This modification aims to increase the occurrence of uneven splits, favoring shorter paths to leaf nodes.

To control the amount of penalization, we introduce the weighting factor $\lambda \in \mathbb{R}^+$ into the regularization term. This factor has to be chosen effectively to obtain a balanced compromise between accuracy and asymmetric splits. Note that there is a limit to which each split can be usefully regularized, as at some point, all samples would go to one child node. Therefore, the impact of the regularization factor reaches a limit the larger the factor becomes. We quantify the expected performance improvement by the expected depth d_o^{exp} of a single tree o, which is given by

$$d_o^{exp} = \sum_{\tau \in \mathcal{T}_o} p_\tau \cdot d_{o,\tau},$$

where $\mathcal{T}_o$ is the set of all leaves τ of a tree o. The probability of a leaf τ is denoted by p_τ and $d_{o,\tau}$ defines the depth of leaf τ in the tree o.

Reducing the expected depth indicates increased performance as fewer nodes need to be loaded during inference. As soon as the expected depth stabilizes, the influence of the regularization factor is less evident and a lower performance gain is to be expected. To estimate the optimal regularization factor, we iteratively increase λ until the difference in expected depth falls below a certain threshold.

To evaluate the impact of regularization in the training of Random Forests, we perform extensive experiments on eleven datasets from the UCI repository (Dua and Graff, 2017), focusing on classification tasks. Furthermore, we investigated different implementations in terms of cache awareness of Random Forests as proposed by Chen et al. (2022) using Gini impurity as the split criterion. The experiments are evaluated with respect to the balanced accuracy and relative mean execution time of the inference that compares the regularized implementation with the comparable non-regularized counterpart over 50 repetitions. For Random Forest hyperparameters, we varied the maximal depth and the size of the inner bootstrap sample m_{try} across a set of recommended default values.

The results show that certain datasets, especially those with binary classification or a large sample size, can benefit significantly from a high degree of regularization. This improvement is seen in an improved runtime without significant loss of accuracy. While regularization for shallow decision trees generally cannot offer a large spectrum for the trade-off and quickly degrades to extreme cases, a broader spectrum for the trade-off is generally offered for deeper tree models. Moreover, our observations show that both cache-optimized and non-cache-optimized implementations yield similar results, which supports the design principle of orthogonal optimization.

In order to better understand the effects of tree regularization on binary classification datasets, we conducted a simulation study. We focused on a binary classification problem with ten features and modeled three different dependency structures ranging from independence to strong dependence while systematically modifying the class balances. Various relations between the outcome and the features are considered. For each setting, we generated samples of size $n \in \{100, 200, 500\}$. The regularization strength λ and the other tree hyperparameters are varied as in the data analyses.

The results show that class balance and sample size are the driving factors that influence the effects of regularization. Analyzing the effects of sample size revealed trends consistent with previous results from analyzing real-world datasets. A significant improvement in execution time was observed with increasing sample size, accompanied by a slight decrease in accuracy. Furthermore, our results showed that highly imbalanced classes in the results allowed for a significant improvement in speed with only a slight degradation in accuracy.

To sum up, we introduce a regularization term for the provoking uneven splits in decision tree training, including an implementation in scikit-learn (Pedregosa et al., 2011). We examine its influence on several real-life datasets from UCI and on simulated data. We show that, especially for binary classification data sets and data sets with many samples, this form of regularization can lead to a reduction of up to $\approx 4\times$ in the execution time with only a slight accuracy degradation.

4 Further Research

During my doctoral studies, I was involved in several other research projects. These cover a wide range of topics, such as human activity recognition, interpretable transfer learning, multi-label stacking, image-based analysis, predictive modeling and theoretical foundations of machine learning algorithms. The following sections summarize the main results of these works.

4.1 Human Activity Recognition in Logistics

All the results presented in this section were developed in collaboration with colleagues from the Department of Statistics and the Chair of Materials Handling and Warehousing at TU Dortmund University dealing with different aspects of human activity recognition (HAR), see our joint research papers Kirchhof et al. (2021b,a); Nair et al. (2023) for details.

The aim of HAR is to automatically identify and classify human activities based on sensor or multi-channel time series data. It has applications in various fields, including healthcare (Wang et al., 2019; Osmani et al., 2008), sports (Tian et al., 2013; Hsu et al., 2018), smart homes (Yang et al., 2011; Van Kasteren et al., 2010) and logistics (Reining et al., 2019; Rueda et al., 2018).

4.1.1 Interpretable Multi-Label Stacking

Recent developments in HAR focus on the introduction of multi-label representations of human motion. These developments show that exploiting the structure between these classes is essential to achieve better performance than with isolated classifiers (Pakrashi et al., 2016).

For example, consider a warehouse worker's human activity recognition task shown in Figure 4.1 (Kirchhof et al., 2021b). Three embedded sensors provide probabilistic

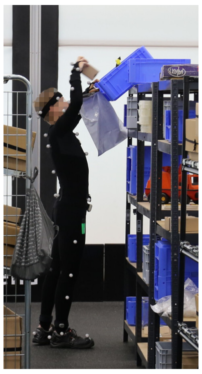


Figure 4.1: The three embedded machine learning sensors tell us: $P(\text{overhead work}) = 0.5$, $P(\text{standing}) = 0.95$, $P(\text{hands high}) = 0.7$. Given the rules: *if standing and hands high then often overhead work* and *if normal work then almost always hands centered or low*, what is the new probability $P(\text{overhead work})$?

information about the worker's posture and hand position, as well as an initial guess about whether the worker is performing overhead tasks. Intuitively, we would combine these beliefs to understand the worker's situation. In addition, we can use rules to decide whether certain assumptions contradict or support each other, as shown in Figure 4.1. The challenge is mathematically applying these non-deterministic rules to uncertain input beliefs to compute updated beliefs.

Most models that address such problems are often stacked on top of ground learners and refine the probabilistic estimates of the ground learners by modeling the structure between labels in various ways: attribute class methods, knowledge graphs, Bayesian networks and probabilistic rule-based approaches.

Attribute class methods introduce a layer of semantically interpretable attributes between input data and output labels (Lampert et al., 2013; Atzmon and Chechik, 2018; Liu et al., 2020), facilitating zero-shot learning (Xian et al., 2019). In contrast, other approaches do not partition labels into attributes and classes but use knowledge graphs (Wu et al., 2018; Lee et al., 2018), which allow for more comprehensive relationships between labels and more straightforward interpretation but often require ground truth graphs. Bayesian networks can extend these relationships to the probabilistic framework (Chen et al., 2020; Shen et al., 2018), while probabilistic rule learners account for uncertainty in labeling (Ding et al., 2015; Rapp et al., 2021).

In Kirchhof et al. (2021b), we introduce the probabilistic rule stacking learner (pRSL), which combines elements from these methods by using Bayesian networks with a bipartite connection of labels through rules, allowing complex and probabilistic relationships between labels while maintaining acyclicity. These rules can either be based on prior knowledge or learned.

In a benchmark study, we compare pRSL to three methods: a neural network with two hidden layers, MLWSE (Xia et al., 2021) and BOOMER (Rapp et al., 2021), which has similarities to pRSL. The benchmark study includes six datasets from Mulan (Tsoumakas et al., 2011). Random Forests (Malley et al., 2012) serve as binary relevance (BR) learners and for evaluation, we examine joint accuracy, joint log-likelihood and label-wise Hamming loss.

The results show that pRSL can compete with other interpretable and black-box algorithms in modeling inter-label dependencies in various metrics. The multi-label algorithms studied generally outperform the BR baseline in all datasets and metrics. Each of the four multi-label algorithms show strengths on different datasets and metrics. However, it should be noted that there are datasets where pRSL does not lead to improvements and may even worsen the results of the underlying classifiers. The reasons for these differences are not yet fully understood, although calibration is considered a possible factor. Further research is required to determine the causes of these differences in performance.

4.1.2 Transfer Learning for Human Activity Recognition in Warehousing

Since manual processes in warehouses account for more than half of total operating costs (De Koster et al., 2007; Grosse et al., 2015), human activities need to be quantifiable to enable their evaluation and improvement in terms of economy and ergonomics (Calzavara et al., 2017). Note that different labels are of interest in each warehouse scenario. Ideally, a classifier would be trained with scenario-specific data. However, collecting and labeling such data is a labor-intensive endeavor.

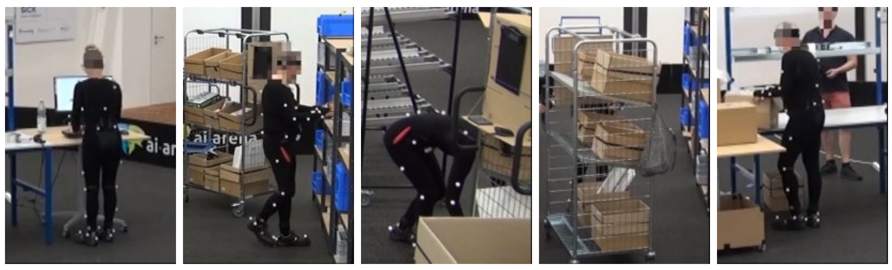
Motivated by the large number of emerging HAR datasets that have activities and labels that resemble warehouse activities (Niemann et al., 2020), we investigate the extent to which existing datasets can be used for HAR in different warehouse scenarios using transfer learning (Kirchhof et al., 2021a). The goal here is to connect the labels of an existing dataset with the possibly different labels of a related target dataset. Transfer learning can be performed using both interpretable and non-interpretable means. In the former case, other HAR classes are associated with rules or structures that are semantically interpretable.

Non-interpretable transfer learning usually involves fine-tuning neural networks (Zhang et al., 2023). Here, a temporal convolution neural network (Rueda and Fink, 2021) is initially trained on a HAR dataset to recognize certain classes. During this process, the network must learn how to summarize the input data in a more compact form, its internal latent representation space, which is usually a black box. This internal representation space is then used as a starting point for learning the classes of a new data set, typically requiring only a few labeled examples of the new dataset, a scenario known as few-shot learning (Cheng et al., 2013).

We focus on localizing the border between zero-shot interpretable and few-shot non-interpretable transfer learning in HAR. Therefore, we apply pRSL and fine-tuning to both a transfer from ergonomics to logistics and a transfer from sports to logistics.

The results of the experiments show that interpretable feature-based transfer learning enables competitive zero-shot recognition but relies on semantically related features between the reference and target datasets. On the other hand, non-interpretable fine-tuning promotes transfer learning even in the face of significant domain shifts, e.g., switching from sports to logistics. Fine-tuned learners outperform fast-learned nulls with only a few annotated examples from the target dataset. These results suggest that transfer learning can significantly reduce the amount of labeled target data required, an essential cost when using deep learning techniques in novel warehousing scenarios.

In analyzing the semantic meaning of the self-learned rules by pRSL, the example of Figure 4.2 (Kirchhof et al., 2021a) shows that label-based transfer learning can produce interpretable rules in the real world. Still, we emphasize that the interpretation strategy needs to be fully developed.



(Stand) $\wedge$ (Bulky Unit / Computer / No Item) $\rightarrow$ Right Hand $\vee$ Left Hand $\vee$ (Step / Standing Still)

Figure 4.2: A self-learned rule from the domain-related transfer learning task together with examples where the rule has a high impact on the classification decision (as measured by the L1 distance to prediction without the rule).

4.1.3 Dataset Bias

Ensuring the accuracy, fairness and generalizability of HAR models is essential for their practical use and compliance with ethical and legal principles. It requires the development of a robust classifier with respect to several criteria, such as personal features. The efficient creation of high-quality datasets is a critical aspect of training such a classifier (Cruz-Sandoval et al., 2019; Reining et al., 2020; Avsar et al., 2021). The issue of classifier robustness is complex and often involves biases introduced by the dataset. Recent research has explored dynamic inductive biases to address challenges such as varying sensor placements (Chang et al., 2020), domain shifts (Khan et al., 2018), inconsistent labels (Reining et al., 2020) and class imbalances (Niemann et al., 2020).

The latest attention to bias in classification, especially within deep network models, has highlighted the influence of human characteristics. However, the significance of subject-specific characteristics in multi-channel time series HAR datasets still needs to be explored. Each individual exhibits unique motion patterns when performing activities, which can be recognized through short-term signal patterns (Retsinas et al., 2020). Consequently, when training and testing data involve subjects with varying physical characteristics, a drop in performance is to be expected.

In our study (Nair et al., 2023), we investigate the extent to which biases in human physical features in the training and testing datasets affect classifier performance. The experiments are performed on different HAR datasets from different domains, considering the heterogeneity introduced by different features in the training dataset. As a classifier, we use the established method proposed by Rueda et al. (2018) and evaluate the performance based on the accuracy and weighted F1 score.

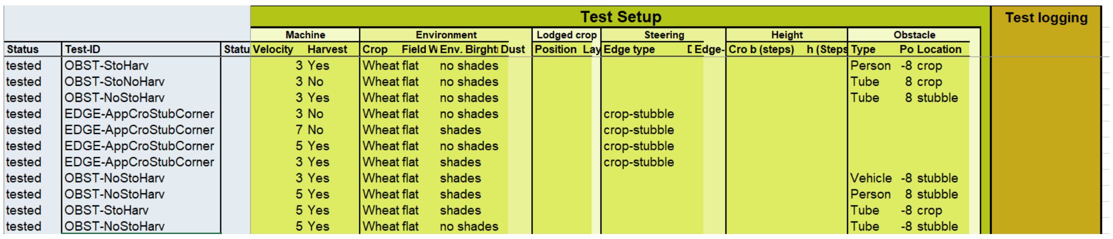
We have found that training data containing a wide range of physical features leads to better accuracy for unseen test data containing objects with different physical features. In addition, systematically increasing the size of the training sample to include subjects with different features further improves accuracy. We therefore recommend that dataset creators ensure that they have subjects with varying characteristics in their datasets. Thus, the number of subjects and the diversity of their physical features can be increased to improve the classifier's performance if the effort to collect more data is reasonable.

4.2 resKIL: Resource Efficient AI for Embedded Systems in Agricultural Machinery

The three-year project ”resource-efficient artificial intelligence for embedded systems in agricultural technology” (resKIL), funded by the German Federal Ministry of Food and Agriculture (BMEL), is being carried out in collaboration with industrial partners CLAAS and Zauberzeug as well as academic institutions such as the University of Osnabrück and the German Research Center for Artificial Intelligence (Bundesministerium für Ernährung und Landwirtschaft, 2022).

The goal of the project is to develop a scalable and adaptable solution to significantly expand the use of machine learning in agriculture in the near future. The project focuses on specific areas, such as quality assessment of harvested crops and feature recognition in the machinery environment, with an agile development strategy to ensure continuous quality improvement. The project planning revolves around three harvest periods, determining the workflow and iterations of individual work packages. The transfer of developments across multiple harvests is a crucial aspect of demonstrating the generalizability, robustness and improved quality of the approaches.

In this project, my main tasks can be divided into three components. First, I am involved in developing a statistical framework for data collection and processing, which includes the creation of detailed experimental designs for the subsequent training of intelligent methods, i.e., to cover the artificial intelligence (AI) training solutions space comprehensively. Based on CLAAS' experience, detailed test plans are designed by means of surrogate models, statistically evaluated and expanded to achieve optimal coverage of the solution space. In addition to field or harvest trials, trials are conducted on test parcels to minimize uncertainties in harvesting, such as data collection problems or unsuitable trial conditions. Moreover, sequential experimental designs are used because they can be adapted quickly. A small extraction of the experimental plans for the harvest in 2022 is presented in Figure 4.3.



			Test Setup																Test logging
			Machine		Environment				Lodged crop		Steering			Height		Obstacle			
Status	Test-ID	Statu	Velocity	Harvest	Crop	Field W	Env. Birght	Dust	Position	Lay	Edge type	D	Edge-	Cro b (steps)	h (Steps	Type	Po	Location	
tested	OBST-StoHarv		3	Yes	Wheat	flat	no shades									Person	-8	crop	
tested	OBST-StoNoHarv		3	No	Wheat	flat	no shades									Tube	8	crop	
tested	OBST-NoStoHarv		3	Yes	Wheat	flat	no shades									Tube	8	stubble	
tested	EDGE-AppCroStubCorner		3	No	Wheat	flat	no shades				crop-stubble								
tested	EDGE-AppCroStubCorner		7	No	Wheat	flat	shades				crop-stubble								
tested	EDGE-AppCroStubCorner		5	Yes	Wheat	flat	no shades				crop-stubble								
tested	EDGE-AppCroStubCorner		3	Yes	Wheat	flat	shades				crop-stubble								
tested	OBST-NoStoHarv		3	Yes	Wheat	flat	shades									Vehicle	-8	stubble	
tested	OBST-NoStoHarv		5	Yes	Wheat	flat	shades									Person	8	stubble	
tested	OBST-StoHarv		5	Yes	Wheat	flat	shades									Tube	-8	crop	
tested	OBST-NoStoHarv		5	Yes	Wheat	flat	no shades									Tube	-8	stubble	

Figure 4.3: Extraction of the experimental design for the harvest 2022.

Another aspect of my involvement is providing statistical support to the collaboration partners. This includes advising on the selection and evaluation of models in the development of a scalable AI platform product line for embedded systems or the evaluation of the various annotation tools developed.

The final aspect is the evaluation of crop quality, with a focus on corn harvesting. The primary metric used for this purpose is the Corn Silage Processing Score (CSPS), which evaluates the efficiency of corn grain processing during harvest and ensiling (Ferreira and Mertens, 2005). In general, it indicates how much of the corn kernels were processed appropriately. To calculate the CSPS, a harvest sample is sent to a laboratory where it is dried and sieved through a 4.74 mm sieve. Whole, unprocessed kernels are usually left on the sieve. Starch content is determined in the whole sample and in the fraction smaller than 4.75 mm. CSPS is then given by the percentage of starch content in the fine fraction relative to the total starch content. Percentages above 70 % are considered very good, while percentages between 70 % and 50 % are considered good and those below 50 % are considered unsatisfactory (Drewry et al., 2019).

In order to optimize this labor-intensive and costly process and to overcome the disadvantage that the results are only available after harvesting when adjustments to the machine settings are no longer possible, an algorithm was developed with the following objectives: The algorithm is intended to closely approximate the CSPS determined in the laboratory while ensuring that the algorithm operates with high resource efficiency so that it can be applied in real-time during the harvesting process. The algorithm uses images captured during the harvesting process as input and then applies an image segmentation procedure, see Figure 4.4 from CLAAS. This allows the calculation of an optical CSPS based on the segmented data.

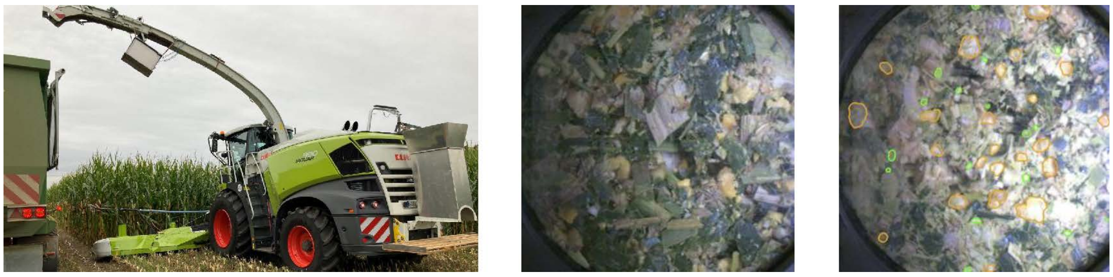


Figure 4.4: Harvest set up (left), example image captured during harvest (center) and its segmented version (right).

In our study, we perform a comprehensive analysis of the optical CSPS and make a direct comparison with the laboratory CSPS. We also compared the algorithm for calculating the optical CSPS with alternative statistical methods. These methods use all the information obtained from image segmentation as input. In this analysis, we evaluated the trade-off between accurate CSPS determination and associated resource allocation to gain a better understanding of the practicality and effectiveness of these resource-efficient approaches. We plan to publish the results of this study together with our partners from CLAAS soon.

4.3 Predicting Effects of Math Training with Multivariate Random Forests

Predicting the success of interventions is essential in education because it is critical to realizing the full potential of individual learners, reducing educational disparities and allocating societal resources efficiently. These predictive efforts span multiple educational approaches, including adaptive interventions, in which learning experiences are adjusted based on real-time data (Collins et al., 2004; Almirall and Chronis-Tuscano, 2016); response to intervention, a system for identifying and supporting students at risk of learning disabilities (Grosche and Huber, 2012); and personalized education, which includes tailored strategies and technologies (Tetzlaff et al., 2021). Educational outcomes are often multidimensional, reflecting performance in different domains or showing trajectories across different points in time.

Standard methods for modeling learning trajectories include latent growth curve models (LGCMs) and multilevel regression models. LGCMs are used to capture individual-level changes over time. De Koning et al. (2014) used LGCMs to predict clickstream data trajectories. Another approach, multilevel regression models, considers the hierarchical structure of data and includes repeated measures within individuals or clustered data. Interestingly, multilevel regression models can often be expressed as LGCMs (McNeish and Matta, 2018). However, they can be limited in their applicability, especially for small sample sizes, high dimensionality, unidentified data, or noisy datasets. Alternative approaches, including machine learning, can provide solutions to overcome these limitations.

Based on this and the findings of our previous work (Schmid et al., 2023a), we started together with our colleagues Susanne Frick, Philip Doebler and Jörg-Tobias Kuhn from the interdisciplinary research group FAIR [1] to investigate the use of multivariate and univariate Random Forests to predict learning trajectories based on digital training data

[1] `https://fair.tu-dortmund.de/en/`

from the Meister CODY Talasia app from January 2018 to December 2019. The project is in the final stages and will be submitted soon.

The math training app CODY Talasia is designed to support children in second and third grades who have problems with math. Developed by researchers at the University of Münster, designed and marketed by Meister CODY GmbH, the app follows a structured training concept with a detailed training plan. The training plan consists of various components:

1. CODY-M 2-4 Test (Kuhn et al., 2013): This test is a validated mathematical assessment to determine a user's training profile and is categorized as A, B, or C. It is recommended to perform it at the beginning and to repeat it if necessary.
2. Training sessions (Kuhn and Holling, 2014): Users are required to attend training sessions five times per week. Each session consists of two tasks, each lasting around ten minutes. The level of difficulty is adaptive and aims for a success rate of 80%. The tasks follow a specific training plan determined by the training profiles. Figure 4.5 shows some example tasks from the training (Chromik, 2021).
3. Status test: This test, done after every five days of training (approximately once a week), evaluates the user's performance in three subdomains: addition, subtraction and number lines. It uses a gold coin scoring system that combines speed and accuracy and has been shown to perform better than separate assessments, as shown in Schwenk et al. (2017).

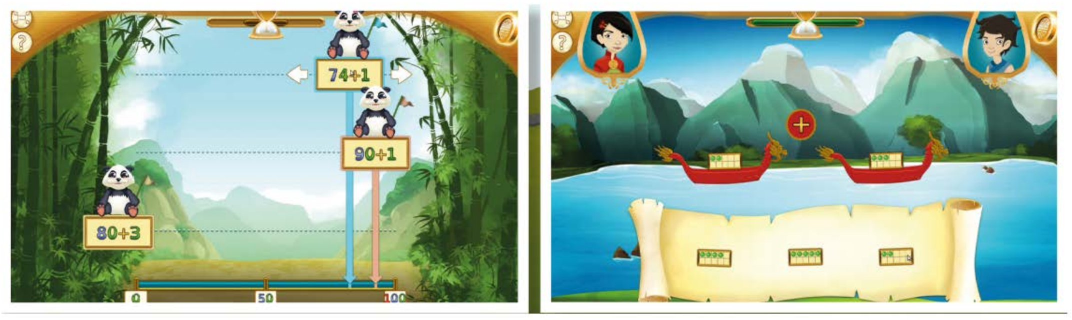


Figure 4.5: Excerpts from the app training tasks.

In our study, we use two different models to identify features that influence the predictions. The first model, referred to as the base model, includes key variables such as pretest score, grade and their interaction. In contrast, the second model extends the base model to include additional training-related features such as average response time and difficulty level. The evaluation of both models for each status test included univariate and multivariate approaches. In the univariate approach, we developed separate models for each subdomain. In the multivariate approach, however, we created a single model to predict performance in all three subdomains. For our analysis, we used Random Forests and regularized linear regression with lasso penalty as prediction methods.

The results indicate that feature selection has a significant impact on prediction performance. Models with more features show better performance, therefore information on the training data is a useful source for early predictions. Comparison between multivariate and univariate approaches shows minimal differences in performance, with multivariate models favoring the baseline setting and the opposite in the training setting. Furthermore, we find no significant differences in predictive performance between Random Forests and regularized linear regression.

The decision to adopt systems like Meister Cody based on these results involves both statistical and technical considerations. On the statistical side, it requires achieving low prediction errors and recognizing intervention effect heterogeneity. Simultaneously, from a technical standpoint, we need to assess the feasibility of implementing these statistical insights.

4.4 Capsule Network

Capsule networks are a recent advance in deep learning that aims to overcome some limitations of traditional convolutional neural networks. They were introduced by Sabour et al. (2017) to improve computer vision tasks, particularly in handling hierarchical and spatial relationships within images. Capsule networks use capsules as basic building blocks instead of traditional neurons. These capsules are designed to capture and represent different parts and attributes of an object, allowing for better generalization and improved recognition of complex patterns (Hinton et al., 2011).

A neuron computes the weighted sum of its scalar inputs and applies a nonlinear activation function to produce a scalar outcome. In contrast to that, the multivariate outcome from a capsule is determined as follows: Let $p, d \in \mathbb{N}$ be arbitrary and $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^p$ be the p-dimensional input vectors of a given capsule. Denote by $\mathbf{W}_i \in \mathbb{R}^{d \times p}$ the corresponding weight matrices. Then, the d-dimensional outcome $\mathbf{y} \in \mathbb{R}^d$ is given by

$$\mathbf{y} = \Psi\left(\sum_{i=1}^{n} u_i \mathbf{W}_i \mathbf{x}_i\right),$$

where $u_i \in \mathbb{R}$ is the so-called i-th coupling coefficient (calculated via the dynamic routing algorithm Sabour et al. (2017)) and $\Psi : \mathbb{R}^p \to \mathbb{R}^p$ is the activation function.

Hinton et al. (2011) also introduces a new nonlinear activation function, the $squash$ function, which limits the length of a vector to a maximum value of one while maintaining the original direction of the vector, i.e.,

$$squash : \mathbb{R}^p \to \mathbb{R}^p, \quad \mathbf{x} \to \frac{\|\mathbf{x}\|}{\|\mathbf{x}\| + 1}\mathbf{x},$$

where $\|\cdot\|$ denotes the euclidean norm. The universal approximation theorem in neural networks (Devroye et al., 2013) states that a single hidden layer in a feedforward neural network with a sufficient number of neurons and a suitable activation function can approximate any continuous function with arbitrary accuracy. We have extended this theorem for multivariate functions in the context of capsule networks for certain activation functions and network structures. This work is in the final stages of preparation for submission, contributing to the ongoing exploration and advancement of neural network capabilities in handling complex multivariate relationships.

5 Discussion and Outlook

In this work, we investigated the application of tree-based ensembles and evaluated their effectiveness in multi-output regression and their adaptability in time-dependent outcomes. Driven by the increase of autonomous and decentralized processes, especially on resource-constrained devices in logistics, we investigate the impact of regularization techniques to improve the execution efficiency of these algorithms.

Our investigation of multi-output regression has shown the potential advantages of simultaneously considering dependencies between outcomes. A comparative analysis of multivariate approaches, in particular Random Forests and Extra Trees, with their univariate counterparts revealed that the multivariate methods show comparable or better performance in scenarios where the data generation process does not differ significantly between all outcome components. Conversely, the univariate approaches showed a performance advantage in scenarios with different data generation processes for each component. In our implementation, the multivariate methods outperformed the univariate methods in terms of runtime efficiency. However, it is important to acknowledge that our implementation still has room for improvement compared to the effective implementation of Simm et al. (2014) and Wright and Ziegler (2017). While our simulations focused primarily on continuous features, analogous results are observed when applied to mixed features in real-world datasets. In a joint project with colleagues from FAIR (Section 4.3), we extended our analysis to a real-life dataset derived from a digital learning app to predict the effects of math training. Here, we focus on the comparative analysis of multivariate tree-based ensembles with other regression methods, such as lasso regression. Simultaneously, the variable importance measure and the effective tuning of hyperparameters in the context of multi-output regression is being explored. Another direction for future research could be exploring hybrid approaches for multi-output regression. These hybrid models, in which components are modeled in part using both univariate and multivariate approaches, promise more flexibility in the modeling process.

Further, we conducted an exhaustive comparison between traditional time series models such as (seasonal) ARIMA and TBATS and tree-based ensembles, including Random Forests and XGBoost, to forecast time series in the dynamic landscape of data-driven logistics. Our simulations uncovered nuanced performance variations, showcasing the superiority of machine learning methods in specific scenarios, particularly when addressing queuing situations and data overlaid with additional (nonlinear) complexity. The out-of-the-box performance of Random Forests, which outperformed XGBoost in all considered scenarios, underlines its robustness in terms of hyperparameter tuning. Training on differentiated time series can significantly improve the machine learning resilience. Our findings are relevant to the logistics domain as they represent a carefully considered, data-driven approach to improving forecasting accuracy, inventory management and operational efficiency. Based on these findings, we have developed recommendations for practical researchers, providing guidance for implementing robust forecasting strategies in real-world logistics scenarios. Extending our research to multistep forecasting, considering multivariate time series and dealing with uncertainty in forecasts are interesting potential topics.

Due to the increasing trend towards autonomous and decentralized processes, we introduced a regularization approach in the construction of tree-based methods to improve the efficient execution time of inference. Our proposed approach, leveraging regularization, aimed to balance model size reduction and maintaining prediction accuracy, effectively addressing challenges posed by limited energy and time resources. The experimental results highlighted significant improvements in runtime, particularly for datasets concerning binary classification problems and those with large sample sizes. An interesting application perspective arises in projects like resKIL (Section 4.2) as well as in autonomous robots in logistics, where it is important to use resource-efficient algorithms to enable real-time applications. Future considerations involve extending regularization to the entire structure of the tree-based ensemble instead of individual trees, offering a promising approach for further investigation. For instance, stratifying the dataset into subsets with strong dependencies for training different trees could potentially enhance the effectiveness of the regularization approach.

Bibliography

Abbasi, R. A., Javaid, N., Ghuman, M. N. J., Khan, Z. A., Ur Rehman, S., and Amanullah (2019). Short term load forecasting using xgboost. In *Web, Artificial Intelligence and Network Applications (WAINA 2019)*, volume 927, pages 1120–1131. Springer.

Al-Saba, T. and El-Amin, I. (1999). Artificial neural networks as applied to long-term demand forecasting. *Artificial Intelligence in Engineering*, 13(2):189–197.

Almirall, D. and Chronis-Tuscano, A. (2016). Adaptive interventions in child and adolescent mental health. *Journal of Clinical Child & Adolescent Psychology*, 45(4):383–395.

Angelini, E., di Tollo, G., and Roli, A. (2008). A neural network approach for credit risk evaluation. *The Quarterly Review of Economics and Finance*, 48(4):733–755.

Atzmon, Y. and Chechik, G. (2018). Probabilistic and-or attribute grouping for zero-shot learning. *arXiv preprint arXiv:1806.02664*.

Avsar, H., Altermann, E., Reining, C., Rueda, F. M., Fink, G. A., and ten Hompel, M. (2021). Benchmarking Annotation Procedures for Multi-channel Time Series HAR Dataset. In *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and Other Affiliated Events*, pages 453–458.

Bartz, E., Bartz-Beielstein, T., Zaefferer, M., and Mersmann, O. (2023). *Hyperparameter Tuning for Machine and Deep Learning with R: A Practical Guide*. Springer Nature.

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A.-L., Deng, D., and Lindauer, M. (2023). Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13(2):e1484.

Box, G. E. and Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society Series B (Methodology)*, 26(2):211–243.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

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

Breiman, L. (1998). Arcing classifier (with discussion and a rejoinder by the author). *The Annals of Statistics*, 26(3):801–849.

Breiman, L. (1999). Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493–1517.

Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

Breiman, L. (2017). *Classification and Regression Trees*. Routledge.

Brockwell, P. J. and Davis, R. A. (2002). *Introduction to time series and forecasting*. Springer.

Bühlmann, P. (1997). Sieve bootstrap for time series. *Bernoulli*, pages 123–148.

Bühlmann, P. and Hothorn, T. (2007). Boosting algorithms: Regularization, prediction and model fitting. *Statistical Science*, 22(4):477–505.

Bundesministerium für Ernährung und Landwirtschaft (2022). resKIL – damit maschinelles Lernen in der Landwirtschaft deutlich mehr zum Einsatz kommt. `https://www.bmel.de/SharedDocs/Praxisbericht/DE/kuenstliche-intelligenz/resKIL.html`, last accessed on 2023-11-01.

Calzavara, M., Glock, C. H., Grosse, E. H., Persona, A., and Sgarbossa, F. (2017). Analysis of economic and ergonomic performance measures of different rack layouts in an order picking warehouse. *Computers & Industrial Engineering*, 111:527–536.

Chang, Y., Mathur, A., Isopoussu, A., Song, J., and Kawsar, F. (2020). A Systematic Study of Unsupervised Domain Adaptation for Robust Human-Activity Recognition. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 4(1):1–30.

Chen, J., Dai, X., Yuan, Q., Lu, C., and Huang, H. (2020). Towards interpretable clinical diagnosis with bayesian network ensembles stacked on entity-aware cnns. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3143–3153.

Chen, K.-H., Su, C., Hakert, C., Buschjäger, S., Lee, C.-L., Lee, J.-K., Morik, K., and Chen, J.-J. (2022). Efficient realization of decision trees for real-time inference. *ACM Transactions on Embedded Computing Systems*, 21(6):1–26.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. Association for Computing Machinery.

Cheng, H.-T., Sun, F.-T., Griss, M., Davis, P., Li, J., and You, D. (2013). Nuactiv: Recognizing Unseen New Activities Using Semantic Attribute–based Learning. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, page 361–374.

Chromik, S. L. (2021). Evaluation mehrdimensionaler längsschnittlicher felddaten aus einer dyskalkulie - trainingsapp. Master's thesis, TU Dortmund University.

Clémençon, S., Depecker, M., and Vayatis, N. (2013). Ranking forests. *Journal of Machine Learning Research*, 14:39–73.

Collins, L. M., Murphy, S. A., and Bierman, K. L. (2004). A conceptual framework for adaptive preventive interventions. *Prevention Science*, 5:185–196.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Crowder, M. J. and Hand, D. J. (2017). *Analysis of repeated measures*. Routledge.

Cruz-Sandoval, D., Beltran-Marquez, J., Garcia-Constantino, M., Gonzalez-Jasso, L. A., Favela, J., Lopez-Nava, I. H., Cleland, I., Ennis, A., Hernandez-Cruz, N., Rafferty, J., Synnott, J., and Nugent, C. (2019). Semi-Automated Data Labeling for Activity Recognition in Pervasive Healthcare. *Sensors*, 19(14).

Cutler, A. and Zhao, G. (2001). Pert-perfect random tree ensembles. *Computing Science and Statistics*, 33(4).

David, R., Duke, J., Jain, A., Janapa Reddi, V., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Wang, T., Warden, P., and Rhodes, R. (2021). Tensorflow lite micro: Embedded machine learning for tinyml systems. In *Proceedings of Machine Learning and Systems*, volume 3, pages 800–811.

De Koning, B., Timmerman, M., van Geert, P., and van der Meulen, B. (2014). Identifying learning trajectories while playing a learning-to-learn computer game in different children and instruction types. *Journal of Cognitive Education and Psychology*, 13(1):53–75.

De Koster, R., Le-Duc, T., and Roodbergen, K. J. (2007). Design and control of warehouse order picking: A Literature Review. *European Journal of Operational Research*, 182(2):481–501.

De Livera, A. M., Hyndman, R. J., and Snyder, R. D. (2011). Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, 106(496):1513–1527.

De'Ath, G. (2002). Multivariate regression trees: a new technique for modeling species - environment relationships. *Ecology*, 83(4):1105–1117.

Delfmann, W., Ten Hompel, M., Kersten, W., Schmidt, T., and Stölzle, W. (2018). Logistics as a science: Central research questions in the era of the fourth industrial revolution. *Logistics Research*, 11(9):1–13.

Devroye, L., Györfi, L., and Lugosi, G. (2013). *A Probabilistic Theory of Pattern Recognition*, volume 31. Springer Science & Business Media.

Díaz-Uriarte, R. and Alvarez de Andrés, S. (2006). Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7(3).

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15. Springer.

Dietterich, T. G. (2002). Machine learning for sequential data: A review. In *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops SSPR 2002 and SPR 2002 Windsor, Ontario, Canada, August 6–9, 2002 Proceedings*, pages 15–30. Springer.

Ding, N., Deng, J., Murphy, K. P., and Neven, H. (2015). Probabilistic label relation graphs with ising models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Drewry, J. L., Luck, B. D., Willett, R. M., Rocha, E. M., and Harmon, J. D. (2019). Predicting kernel processing score of harvested and processed corn silage via image processing techniques. *Computers and Electronics in Agriculture*, 160:144–152.

Dua, D. and Graff, C. (2017). Uci machine learning repository.

Dudek, G. (2015). Short-term load forecasting using random forests. In *Intelligent Systems' 2014: Proceedings of the 7th IEEE International Conference Intelligent Systems IS'2014, September 24-26, 2014, Warsaw, Poland, Volume 2: Tools, Architectures, Systems, Applications*, pages 821–828. Springer.

Everingham, Y., Sexton, J., Skocaj, D., and Inman-Bamber, G. (2016). Accurate prediction of sugarcane yield using a random forest algorithm. *Agronomy for sustainable development*, 36:1–9.

Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15(1):3133–3181.

Ferreira, G. and Mertens, D. (2005). Chemical and physical characteristics of corn silages and their effects on in vitro disappearance. *Journal of Dairy Science*, 88(12):4414–4425.

Fokam, C. T. (2022). Ar-sieve bootstrap for the random forest and a simulation-based comparison with rangerts time series prediction. Master's thesis, TU Dortmund University.

Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. *Journal of Japanese Society For Artificial Intelligence*, 14:771–780.

Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The Annals of Statistics*, 28(2):337–407.

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

Gardner Jr., E. S. and McKenzie, E. (1985). Forecasting trends in time series. *Management Science*, 31(10):1237–1246.

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

Goehry, B., Yan, H., Goude, Y., Massart, P., and Poggi, J.-M. (2017). Rangerts: a ranger implementation for time series. `https://github.com/hyanworkspace/rangerts`. Last accessed on 10.12.2023.

Goehry, B., Yan, H., Goude, Y., Massart, P., and Poggi, J.-M. (2021). Random forests for time series. *hal-03129751*.

Grosche, M. and Huber, C. (2012). Das response-to-intervention-modell als grundlage für einen inklusiven paradigmenwechsel in der sonderpädagogik. *Zeitschrift für Heilpädagogik*, 8(12):312–322.

Grosse, E. H., Glock, C. H., and Neumann, W. P. (2015). Human Factors in Order Picking System Design: A Content Analysis. *IFAC-PapersOnLine*, 48(3):320–325.

Gumus, M. and Kiran, M. S. (2017). Crude oil price forecasting using xgboost. In *2017 International conference on computer science and engineering (UBMK)*, pages 1100–1103. IEEE.

Gupta, Y., Lama, R. K., Kwon, G.-R., and Initiative, A. D. N. (2019). Prediction and classification of alzheimer's disease based on combined features from apolipoprotein-e genotype, cerebrospinal fluid, mr, and fdg-pet imaging biomarkers. *Frontiers in Computational Neuroscience*, 13:72.

Hamilton, J. D. (2020). *Time Series Analysis*. Princeton university Press.

Härdle, W., Horowitz, J., and Kreiss, J.-P. (2003). Bootstrap methods for time series. *International Statistical Review*, 71(2):435–459.

Harvey, A. C. (1990). *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*, volume 2. Springer.

Hinton, G. E., Krizhevsky, A., and Wang, S. D. (2011). Transforming auto-encoders. In *Artificial Neural Networks and Machine Learning – ICANN 2011*, pages 44–51. Springer.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844.

Hsu, Y.-L., Yang, S.-C., Chang, H.-C., and Lai, H.-C. (2018). Human daily and sport activity recognition using a wearable inertial sensor network. *IEEE Access*, 6:31715–31728.

Huang, H., Pouls, M., Meyer, A., and Pauly, M. (2020). Travel time prediction using tree-based ensembles. In *Computational Logistics: 11th International Conference, ICCL 2020, Enschede, The Netherlands, September 28–30, 2020, Proceedings 11*, pages 412–427. Springer.

Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts.

Hyndman, R. J. and Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software*, 27(3):1–22.

Ishwaran, H., Kogalur, U. B., Blackstone, E. H., and Lauer, M. S. (2008). Random survival forests. *The Annals of Applied Statistics*, 2(3):841 – 860.

Kane, M. J., Price, N., Scotch, M., and Rabinowitz, P. (2014). Comparison of arima and random forest time series models for prediction of avian influenza h5n1 outbreaks. *BMC Bioinformatics*, 15(1):1–9.

Khaidem, L., Saha, S., and Dey, S. R. (2016). Predicting the direction of stock market prices using random forest. *arXiv preprint arXiv:1605.00003*.

Khan, M. A. A. H., Roy, N., and Misra, A. (2018). Scaling Human Activity Recognition via Deep Learning-based Domain Adaptation. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–9.

Kirchhof, M., Schmid, L., Reining, C., ten Hompel, M., and Pauly, M. (2021a). Chances of interpretable transfer learning for human activity recognition in warehousing. In *Computational Logistics*, pages 163–177. Springer.

Kirchhof, M., Schmid, L., Reining, C., ten Hompel, M., and Pauly, M. (2021b). prsl: Interpretable multi-label stacking by learning probabilistic rules. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161, pages 461–470.

Konishi, S. and Kitagawa, G. (2008). *Information criteria and statistical modeling*. Springer Science & Business Media.

Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V., and Fotiadis, D. I. (2015). Machine learning applications in cancer prognosis and prediction. *Computational and Structural Biotechnology Journal*, 13:8–17.

Krüger, A., Feldmann, F., Pauly, M., and Ten Hompel, M. (2020). Einsatzmöglichkeiten maschineller lernverfahren in einer dezentral organisierten lagerverwaltung auf basis intelligenter behälter. *Logistics Journal: Proceedings*, 2020(12).

Kuhlmann, L. and Pauly, M. (2023). A dynamic systems model for an economic evaluation of sales forecasting methods. *Tehnički glasnik*, 17(3):397–404.

Kuhn, J.-T. and Holling, H. (2014). Number sense or working memory? The effect of two computer-based trainings on mathematical skills in elementary school. *Advances in cognitive psychology*, 10(2):59.

Kuhn, J.-T., Raddatz, J., Holling, H., and Dobel, C. (2013). Dyskalkulie vs. rechenschwäche: Basisnumerische verarbeitung in der grundschule. *Lernen und Lernstörungen*.

Kuhn, M. (2008). Building predictive models in r using the caret package. *Journal of Statistical Software*, 28:1–26.

Kwiatkowski, D., Phillips, P. C., Schmidt, P., and Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, 54(1–3):159–178.

Lahouar, A. and Slama, J. B. H. (2017). Hour-ahead wind power forecast based on random forests. *Renewable energy*, 109:529–541.

Lampert, C. H., Nickisch, H., and Harmeling, S. (2013). Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465.

Lee, C.-W., Fang, W., Yeh, C.-K., and Wang, Y.-C. F. (2018). Multi-label zero-shot learning with structured knowledge graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1576–1585.

Littell, R. C., Henry, P., and Ammerman, C. B. (1998). Statistical analysis of repeated measures data using sas procedures. *Journal of animal science*, 76(4):1216–1231.

Liu, L., Zhou, T., Long, G., Jiang, J., and Zhang, C. (2020). Attribute propagation network for graph zero-shot learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):4868–4875.

Liu, Y. and Wu, H. (2017). Prediction of road traffic congestion based on random forest. In *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, volume 2, pages 361–364. IEEE.

Loh, W.-Y. (2002). Regression tress with unbiased variable selection and interaction detection. *Statistica Sinica*, 12(2):361–386.

Loh, W.-Y. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23.

Loh, W.-Y. and Shih, Y.-S. (1997). Split selection methods for classification trees. *Statistica Sinica*, 7(4):815–840.

Malley, J. D., Kruppa, J., Dasgupta, A., Malley, K. G., and Ziegler, A. (2012). Probability machines. *Methods of Information in Medicine*, 51(01):74–81.

Mason, L., Baxter, J., Bartlett, P., and Frean, M. (1999). Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

McNeish, D. and Matta, T. (2018). Differentiating between mixed-effects and latent-curve approaches to growth modeling. *Behavior Research Methods*, 50:1398–1414.

Meinshausen, N. (2006). Quantile regression forests. *Journal of Machine Learning Research*, 7:983–999.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill New York.

Murshed, M. S., Murphy, C., Hou, D., Khan, N., Ananthanarayanan, G., and Hussain, F. (2021). Machine learning at the network edge: A survey. *ACM Computing Surveys*, 54(8):1–37.

Nair, N. R., Schmid, L., Rueda, F. M., Pauly, M., Fink, G. A., and Reining, C. (2023). Dataset bias in human activity recognition. *arXiv preprint arXiv:2301.10161*.

Niemann, F., Reining, C., Rueda, F. M., Nair, N. R., Steffens, J. A., Fink, G. A., and Hompel, M. t. (2020). LARa: Creating a Dataset for Human Activity Recognition in Logistics Using Semantic Attributes. *Sensors*, 20(15).

Osmani, V., Balasubramaniam, S., and Botvich, D. (2008). Human activity recognition in pervasive health-care: Supporting efficient remote collaboration. *Journal of Network and Computer Applications*, 31(4):628–655.

Pakrashi, A., Greene, D., and MacNamee, B. (2016). *Benchmarking Multi-label Classification Algorithms*, pages 149–160. CEUR Workshop Proceedings.

Park, B. and Bae, J. K. (2015). Using machine learning algorithms for housing price prediction: The case of fairfax county, virginia housing data. *Expert Systems with Applications*, 42(6):2928–2934.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

Probst, P., Boulesteix, A.-L., and Bischl, B. (2019a). Tunability: Importance of hyperparameters of machine learning algorithms. *Journal of Machine Learning Research*, 20(1):1–32.

Probst, P., Wright, M. N., and Boulesteix, A.-L. (2019b). Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3):e1301.

Ramosaj, B. (2020). *Analyzing Consistency and Statistical Inference in Random Forest Models*. PhD thesis, Technical University of Dortmund.

Rapp, M., Mencía, E. L., Fürnkranz, J., Nguyen, V.-L., and Hüllermeier, E. (2021). Learning gradient boosted multi-label classification rules. In *Machine Learning and Knowledge Discovery in Databases*, pages 124–140. Springer.

Reining, C., Niemann, F., Moya Rueda, F., Fink, G. A., and ten Hompel, M. (2019). Human activity recognition for production and logistics – a systematic literature review. *Information*, 10(8):245.

Reining, C., Rueda, F. M., Niemann, F., Fink, G. A., and Hompel, M. t. (2020). Annotation performance for multi-channel time series har dataset in logistics. In *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 1–6.

Retsinas, G., Filntisis, P. P., Efthymiou, N., Theodosis, E., Zlatintsi, A., and Maragos, P. (2020). Person Identification Using Deep Convolutional Neural Networks on Short-Term Signals from Wearable Sensors. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3657–3661.

Rueda, F. M. and Fink, G. A. (2021). From human pose to on-body devices for human-activity recognition. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 10066–10073.

Rueda, F. M., Grzeszick, R., Fink, G. A., Feldhorst, S., and ten Hompel, M. (2018). Convolutional neural networks for human activity recognition using body-worn sensors. *Informatics*, 5(2):26.

Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. *Advances in Neural Information Processing Systems*, 30.

Sagi, O. and Rokach, L. (2018). Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249.

Salari, N., Liu, S., and Shen, Z.-J. M. (2022). Real-time delivery time forecasting and promising in online retailing: When will your package arrive? *Manufacturing & Service Operations Management*, 24(3):1421–1436.

Schmid, L., Biebert, D., Hakert, C., Chen, K.-H., Lang, M., Pauly, M., and Chen, J.-J. (2024). Tree: Tree regularization for efficient execution. *arXiv preprint*.

Schmid, L., Gerharz, A., Groll, A., and Pauly, M. (2023a). Tree-based ensembles for multi-output regression: Comparing multivariate approaches with separate univariate ones. *Computational Statistics & Data Analysis*, 179:107628.

Schmid, L., Roidl, M., and Pauly, M. (2023b). Comparing statistical and machine learning methods for time series forecasting in data-driven logistics - a simulation study. *arXiv preprint*.

Schwenk, C., Kuhn, J.-T., Doebler, P., and Holling, H. (2017). Auf goldmünzenjagd: Psychometrische kennwerte verschiedener scoringansätze bei computergestützter lernverlaufsdiagnostik im bereich mathematik. *Empirische Sonderpädagogik*, 9(2):123–142.

Scornet, E. (2017). Tuning parameters in random forests. *ESAIM: Proceedings and Surveys*, 60:144–162.

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

Segal, M. and Xiao, Y. (2011). Multivariate random forests. *Wiley Interdisciplinary Reviews: Data mining and knowledge discovery*, 1(1):80–87.

Shamout, M., Ben-Abdallah, R., Alshurideh, M., Alzoubi, H., Kurdi, B. a., and Hamadneh, S. (2022). A conceptual model for the adoption of autonomous robots in supply chain and logistics industry. *Uncertain Supply Chain Management*, 10(2):577–592.

Shen, R., Zou, F., Song, J., Yan, K., and Zhou, K. (2018). Efui: An ensemble framework using uncertain inference for pornographic image recognition. *Neurocomputing*, 322:166–176.

Shumway, R. H. and Stoffer, D. S. (2017). Arima models. *Time Series Analysis and Its Applications: With R Examples*, pages 75–163.

Siami-Namini, S., Tavakoli, N., and Namin, A. S. (2018). A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE International Conference on Machine learning and Applications*, pages 1394–1401. IEEE.

Simm, J., Magrans de Abril, I., and Sugiyama, M. (2014). Tree-based ensemble multi-task learning method for classification and regression. *IEICE TRANSACTIONS on Information and Systems*, 97(6):1677–1681.

Syntetos, A. A., Babai, Z., Boylan, J. E., Kolassa, S., and Nikolopoulos, K. (2016). Supply chain forecasting: Theory, practice, their gap and the future. *European Journal of Operational Research*, 252(1):1–26.

Tabanelli, E., Tagliavini, G., and Benini, L. (2022). Optimizing random forest-based inference on risc-v mcus at the extreme edge. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(11):4516–4526.

Tetzlaff, L., Schmiedek, F., and Brod, G. (2021). Developing personalized education: A dynamic framework. *Educational Psychology Review*, 33:863–882.

Therneau, T. M. and Atkinson, E. J. (1997). An introduction to recursive partitioning using the rpart routines. Technical report, Mayo Foundation.

Tian, Y., Sukthankar, R., and Shah, M. (2013). Spatiotemporal deformable part models for action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., and Vlahavas, I. (2011). Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414.

Vairagade, N., Logofatu, D., Leon, F., and Muharemi, F. (2019). Demand forecasting using random forest and artificial neural network for supply chain management. In *Computational Collective Intelligence: 11th International Conference, ICCCI 2019, Hendaye, France, September 4–6, 2019, Proceedings, Part I 11*, pages 328–339. Springer.

Van Kasteren, T., Englebienne, G., and Kröse, B. J. (2010). An activity monitoring system for elderly care using generative and discriminative models. *Personal and Ubiquitous Computing*, 14:489–498.

Venkatapathy, A. K. R., Bayhan, H., Zeidler, F., and ten Hompel, M. (2017). Human machine synergies in intra-logistics: Creating a hybrid network for research and technologies. In *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 1065–1068. IEEE.

Wang, Y., Cang, S., and Yu, H. (2019). A survey on wearable sensor modality centred human activity recognition in health care. *Expert Systems with Applications*, 137:167–190.

Wang, Z., Wang, Y., Zeng, R., Srinivasan, R. S., and Ahrentzen, S. (2018). Random forest based hourly building energy prediction. *Energy and Buildings*, 171:11–25.

West, M. and Harrison, J. (2006). *Bayesian Forecasting and Dynamic Models*. Springer Science & Business Media.

Wright, M. N. and Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17.

Wu, B., Jia, F., Liu, W., Ghanem, B., and Lyu, S. (2018). Multi-label learning with missing labels using mixed dependency graphs. *International Journal of Computer Vision*, 126(8):875–896.

Xia, Y., Chen, K., and Yang, Y. (2021). Multi-label classification with weighted classifier selection and stacked ensemble. *Information Sciences*, 557:421–442.

Xian, Y., Lampert, C. H., Schiele, B., and Akata, Z. (2019). Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2251–2265.

Yang, J., Lee, J., and Choi, J. (2011). Activity recognition based on rfid object usage for smart mobile devices. *Journal of Computer Science and Technology*, 26(2):239–246.

Yoon, J. (2021). Forecasting of real gdp growth using machine learning models: Gradient boosting and random forest approach. *Computational Economics*, 57(1):247–265.

Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press.

Zhang, C. and Ma, Y. (2012). *Ensemble machine learning: methods and applications*. Springer.

Zhang, G. P., Patuwo, B. E., and Hu, M. Y. (2001). A simulation study of artificial neural networks for nonlinear time-series forecasting. *Computers & Operations Research*, 28(4):381–396.

Zhang, G. P. and Qi, M. (2005). Neural network forecasting for seasonal and trend time series. *European Journal of Operational Research*, 160(2):501–514.

Zhang, L., Bian, W., Qu, W., Tuo, L., and Wang, Y. (2021). Time series forecast of sales volume based on xgboost. In *Journal of Physics: Conference Series*, volume 1873, page 012067. IOP Publishing.

Zhang, L., Wang, S., and Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253.

Zhou, Z.-H. (2012). *Ensemble methods: foundations and algorithms*. CRC press.

Part II

Publications

Article 1

Schmid, L., Gerharz, A., Groll, A. & Pauly, M. (2023). Tree-based ensembles for multi-output regression: Comparing multivariate approaches with separate univariate ones. *Computational Statistics & Data Analysis*, **179**, 107628, https://doi.org/10.1016/j.csda.2022.107628.

Article 2

Schmid, L., Roidl, M. & Pauly, M. (2023). Comparing statistical and machine learning methods for time series forecasting in data-driven logistics - A simulation study. *arXiv preprint*, https://doi.org/10.48550/arXiv.2303.07139

Comparing statistical and machine learning methods for time series forecasting in data-driven logistics – A simulation study

Lena Schmid[a,*], Moritz Roidl[b], Markus Pauly[a,c]

[a] *Department of Statistics, TU Dortmund University, 44227 Dortmund, Germany*
[b] *Chair of Material Handling and Warehousing, TU Dortmund University, 44221 Dortmund, Germany*
[c] *UA Ruhr, Research Center Trustworthy Data Science and Security, 44227 Dortmund, Germany*

Abstract

Many planning and decision activities in logistics and supply chain management are based on forecasts of multiple time dependent factors. Therefore, the quality of planning depends on the quality of the forecasts. We compare different state-of-the art forecasting methods in terms of out-of-the-box forecasting performance. Different to most existing research in logistics, we do not do this case-dependent but consider a broad set of simulated time series to give more general recommendations. We therefore simulate various linear and nonlinear time series that reflect different situations.

Keywords: Machine Learning, Time Series, Forecasting, Simulation Study

1. Introduction

Forecasting methods are essential for efficient planning in various logistics domains such as warehousing, transport, and supply chain management. They enable companies to anticipate and plan for future demand, capacity needs, and supply chain requirements. Thereby, different logistics applications require different forecasts due to their unique characteristics. In the transport domain, e.g., accurate transportation forecasting enables logistics companies to optimize their transportation networks, reduce transportation costs and enhance delivery reliability (Huang et al., 2020; Wu et al., 2004; Lin et al., 2005; Garrido and Mahmassani, 2000; Wu and Levinson, 2021). Precise forecasting allows warehouse managers to optimize space use, reduce stock-out risk, and improve overall efficiency (Shi et al., 2018; Ribeiro et al., 2022). In supply chain management, accurate forecasts are, e.g., used to optimize resource use across the entire supply chain (Feizabadi, 2022; Kuhlmann and Pauly, 2023; Syntetos et al., 2016). The above references show that the use of forecasting techniques such as time series models and machine learning methods has become increasingly popular in logistics in recent years. However, there is still a lack of consensus on which method is more effective, especially as most methods of comparison in logistics solely rely on comparing the performance on a few data sets (Ensafi et al., 2022; Ribeiro et al., 2022). In fact, different to other fields (e.g. Wu et al., 2018; Weber et al., 2019) there do not exist rigorous benchmark studies in data-driven logistics to the best of our knowledge. In our opinion, the key reason for this is that, outside of specific examples (e.g. Niemann

*Corresponding Author
Email address: lena.schmid@tu-dortmund.de (Lena Schmid)

et al., 2020; Arora et al., 2022), there is a lack of freely accessible and well-characterized data sets for benchmarking (e.g. Reining et al., 2019; Awasthi et al., 2023) in the logistics research domain. This hampers the analysis of domain-specific pros and cons of method choices or the formulation of general recommendations. To overcome this and to be in line with recent recommendations (Friedrich and Friede, 2023), we therefore focus on simulating data from various statistical time series models that reflect potential logistic scenarios.

Time series models have been used in forecasting for several decades and are widely used in logistics for sales or demand forecasting, see e.g. Kuhlmann and Pauly (2023); Shukla and Jharkharia (2011) and the references cited therein. These models are based on historical data and use statistical techniques to identify patterns and trends in the data, which can then be used to make predictions about future demand. Some commonly used time series models in logistics include (seasonal) autoregressive integrated moving averages (ARIMA) and exponential smoothing models. For example, Gilbert (2005) developed an ARIMA multistage supply chain model that is based on time series models. Another example is Prophet (Taylor and Letham, 2018), a forecasting tool for time series analysis developed by Facebook, which includes additive modeling with components such as seasonality, holidays and trend flexibility. Kumar Jha and Pande (2021) examined ARIMA and Prophet models for predicting supermarket sales. The Prophet models showed superior predictive performance in terms of lower errors. Hasmin and Aini (2020) investigated the performance of double exponential smoothing for inventory forecasting.

More recently, machine learning (ML) methods have become increasingly popular for demand forecasting in logistics due to their ability to handle large and complex data sets. There are many literature reviews (Carbonneau et al., 2008; Wenzel et al., 2019; Sharma et al., 2020; Ni et al., 2020; Baryannis et al., 2019), that discuss the use of machine learning techniques in forecasting for supply chain management, including an overview of the various techniques used and their advantages and limitations. However, our comment regarding a lack of neutral benchmarking studies still applies.

Several studies have shown that ML methods such as neural networks, support vector regression, and Random Forests can outperform traditional time series models for specific demand forecasting problems. For example, a study by Ensafi et al. (2022) compared the prediction power of more than ten different forecasting models, including classical methods such as ARIMA and ML techniques such as long short-term memory (LSTM) and convolution neural networks, using a single data set containing the sales history of furniture in a retail store. The results showed that the LSTM outperformed the other models in terms of prediction performance. Another study by Kohzadi et al. (1996) also compared the forecasting power of ARIMA and neural networks using a single commodity prices data set. Again the neural network performed better than the ARIMA model. Similar results were obtained in Weng et al. (2019) or Siami-Namini et al. (2018). However, other studies have found mixed results, with some suggesting that time series models perform better than ML methods. For instance, Palomares-Salas et al. (2009) compared the forecasting accuracy of ARIMA and neural network models in predicting wind speed for short time intervals. The results showed that the performance of both can be very similar, indicating that a more simple and interpretable forecasting model could be used to administrate energy sources. A comparison of daily hotel demand forecasting performance of SARIMAX, GARCH and neutral networks also showed that both time series approaches outperformed the neural networks (Ampountolas, 2021). In the latter examples, one reason may also be the difficulty of tuning complex machine learning procedures. That's one reason why we focus on out-of-the-box machine learning methods in our study.

The comparison of the forecasting performance of ML methods and time series models in logistics has significant implications for businesses seeking to improve their forecasting accuracy. By identifying the most effective forecasting methods, businesses can make better-informed decisions about production, inventory management, and resource allocation. Thus, this work aims to provide a comprehensive comparison of the forecasting performance of time series models and ML methods. Different from the above-mentioned works that merely focus on single use cases, this task needs more variation in the data sets under study. To this end, we compare various forecasting methods in terms of out-of-the-box forecasting performance on a broad set of simulated time series. We thereby simulate various linear and nonlinear time series that are of importance for logistics and study the one-step forecast performance of different statistical learning methods.

This work is structured as follows: Section 2 presents the different used forecasting methods. More precisely, the (seasonal) ARIMA and TBATS models are presented. In addition, the machine learning approaches (Random Forest and XGBoost) are described in more detail. Section 3 presents the simulation design and framework, while Section 4 summarizes the main simulation results. In Section 5, an illustrative real-world data example is analyzed before the manuscript concludes with a discussion of our findings and an outlook for future research (Section 6).

2. Methods

In this section, we explain the one-step forecasting methods under investigation. There are various strategies for modeling and forecasting time series. Traditional time series models, including moving averages and exponential smoothing, follow a linear approach in which the predictions of future values are linear functions of past observations. Due to their relative simplicity in terms of understanding and implementation, linear models have found application in many forecasting problems (Fan et al., 2021; Nyoni, 2018; Benvenuto et al., 2020). To overcome the limitations of linear models and account for certain nonlinear patterns observed in real-world problems, several classes of nonlinear models have been proposed in the literature. Examples cover the threshold autoregressive model (TAR) (Tsay, 1989) or the generalized autoregressive conditional heteroscedastic model (GARCH) (Francq and Zakoian, 2019). Although some improvements have been noted, the utility of their application to general prediction problems is limited (De Gooijer and Kumar, 1992): Since these models were developed for specific nonlinear patterns, they are often unable to model other types of nonlinearities. Here, machine learning methods have been proposed as an alternative for time series forecasting (Bontempi et al., 2012; Ahmed et al., 2010). Since it is impossible to cover the entire spectrum of machine learning models and time series methods in our simulation study, we limit ourselves to a selection of what we consider the most common algorithms in data-driven logistics. To evaluate the performance, we compare these methods with a naive approach, where the last observation of the time series is used as a prediction. The time series (Subsection 2.1) and machine learning methods (Subsection 2.2) under study are explained in more detail in the next two subsections.

2.1. Time Series Methods

We focus on three different time series models: ARIMA, SARIMA, and TBATS. The first two models are among the most popular models in traditional time series forecasting (Brockwell and Davis, 2002; Hyndman and Athanasopoulos, 2018) and are often used as benchmark models for comparison with machine learning algorithms (Al-Saba and El-Amin, 1999; Zhang et al., 2001b; Hwarng, 2001). In addition, TBATS models combine many different approaches that are commonly used in forecasting.

ARIMA. Autoregressive integrated moving average (ARIMA)(Box et al., 2015) model is a generalized model of the autoregressive moving average (ARMA) model and builds a composite model of the time series (Shumway et al., 2000). Denoted as ARIMA(p, d, q), $p, q, d \in \mathbf{N}$, the model is characterized by three key components:

- AR (Autoregression): Represents the regression of the time series on its own past values, capturing dependencies through lagged observations. The number of lagged observations included in the models is given by p.
- I (Integrated): The differencing order (d) indicates the number of times the time series is differenced to achieve stationarity. This transformation involves subtracting the current observation from its d-th lag, which is crucial for stabilizing the mean and addressing trends.
- MA (Moving Average): Incorporates a moving average model to account for dependencies between observations and the residual errors of the lagged observations (q).

In general a time series $\{x_t\}_t$ generated from an ARIMA(p, d,q) model has the form

$$\sum_{i=1}^{p} \phi_i \Delta^d x_{t-i} = \sum_{j=0}^{q} \theta_j \varepsilon_{t-j},$$

where $p, d, q \in \mathbb{N}$, $\phi_1, \ldots, \phi_p \in \mathbb{R}$ are the autoregressive coefficients, $\theta_1, \ldots \theta_q \in \mathbb{R}$ are the moving average coefficients and ε_t denotes the residuals or the errors at time t. The residuals are often assumed to follow a white noise process, represented by a sequence of uncorrelated random variables with zero mean and finite second moment. The difference operator Δ is defined as $\Delta : \mathbb{R} \to \mathbb{R}$ with $x_t \to x_t - x_{t-1}$.

SARIMA. With seasonal time series data, short-term non-seasonal components likely contribute to the model. Therefore, we need to estimate a seasonal ARIMA model incorporating non-seasonal and seasonal factors into a multiplicative model (Shumway et al., 2000). The general form of a seasonal ARIMA model is denoted as SARIMA$(p, d, q)(P, D, Q)_m$, where p is the non-seasonal AR order, d is the non-seasonal differentiation, q is the non-seasonal MA order, P, D and Q are the similar parameters for the seasonal part. The parameter m denotes the number of time steps for a single period.

TBATS. For time series data exhibiting complex and diverse seasonal patterns, TBATS (Trigonometric Seasonal Exponential Smoothing) is a robust modeling approach. Introduced as an extension of exponential smoothing methods, TBATS accounts for different seasonalities through a combination of trigonometric functions and exponential smoothing (De Livera et al., 2011). The model is particularly effective in handling multiple seasonal cycles, making it suitable for data sets with intricate temporal structures.

The general form of a TBATS model consists of several components as described below:

- T (Trend): Captures the overall trend in the time series using an exponential smoothing mechanism.
- B (Box-Cox Transformation): Applies the Box-Cox transformation (Box and Cox, 1964) to stabilize variance and ensure the homogeneity of variances.
- A (ARIMA Errors): Incorporates ARIMA errors to capture any remaining non-seasonal dependencies.
- S (Seasonal): Utilizes trigonometric functions to model multiple seasonal components, accommodating various seasonal patterns.

2.2. Machine Learning Methods

Machine learning methods are increasingly being used to address time series prediction problems. In fact, there exist too many approaches to consider in a comparison study like ours. We therefore restricted ourselves to a class that has already been successfully used for predictions in the logistics context (Ji et al., 2019; Islam and Amin, 2020; Ma et al., 2018; Huang et al., 2020; Kuhlmann et al., 2023): Tree-based ensemble learners. We thereby focus on two models, each studied with and without differencing: Random Forest and XGBoost on trees which are briefly introduced below.

XGBoost. Gradient boosting is an ensemble machine learning technique often used in classification and regression problems and is particularly popular in predictive scenarios (Aguilar Madrid and Antonio, 2021). As an ensemble technique, gradient boosting combines the results of several weak learners, referred to as base learners, with the aim of building a model that generally performs better than the conventional single machine learning models. Typically, gradient boosting utilizes decision trees as base learners. Like other boosting methods, the core idea of gradient boosting is that during the learning procedure, new models are built and fitted consecutively and not independently to provide better predictions of the output variable. Thereby, new base learners are constructed with the aim of minimizing a loss function associated with the whole ensemble. Instances that are not predicted correctly in previous steps and score higher errors are correlated with larger weight values so that the model can focus on them and learn from its mistakes.

XGBoost stands for Extreme Gradient Boosting and is a specific implementation of gradient boosting (Chen and Guestrin, 2016). It incorporates randomization and regularization techniques to reduce overfitting while increasing training speed. Moreover, it computes second-order gradients of the loss function, which provides more information about the gradient's direction, making it easier to minimize the loss function.

In general, the hyperparameters for XGBoost can be divided into two categories (Chen and Guestrin, 2016): General boosting parameters, including the number of iterations and the learning rate, which controls how much information from a new tree will be used in the boosting step. Second, in base learner dependent parameters. When trees are used as base learners, the additional hyperparameters are used to control the complexity of the individual trees. Examples include limiting the maximum tree depth or specifying a minimum number of samples in each leaf (Therneau and Atkinson, 1997). There also exists other boosting variants (Schapire and Freund, 2013; Friedman, 2002; Mayr et al., 2014), but we concentrate on XGBoost as it has emerged as one of the key machine learning models for prediction and was also referred to as *'the*

Queen of Machine Learning' (Morde) in this context. XGBoost models have also been used for time series forecasting, e.g., Luo et al. (2021); Alim et al. (2020). For example, in Zhang et al. (2021) the potential of XGBoost in retail for predicting store sales was investigated while (Huang et al., 2020) studied this for predicting the travel time of NYC cabs.

Random Forest. A Random Forest (Breiman, 2001) is a machine learning method based on building ensembles of decision trees. It was developed to address predictive shortcomings of traditional Classification and Regression Trees (CARTs) (Breiman et al., 2017). Random Forests consist of a large number of weak decision tree learners, which are grown in parallel to reduce the bias and variance of the model at the same time (Breiman, 2001). For training a Random Forest, bootstrap samples are drawn from the training data set. Each bootstrap sample is then used to grow a(n unpruned) tree. Instead of using all available features in this step, only a small and fixed number of randomly sampled m_{try} features are selected as split candidates. A split is chosen by the CART-split criterion for regression, i.e., by minimizing the sum of squared errors in both child nodes. Instead of the CART-split criterion, many other distances, such as the least absolute deviations of the mean (L1-norm), can also be used. These steps are repeated until B such trees are grown, and new data is predicted by taking the mean of all B tree predictions. The most important hyperparameters for the Random Forest (Wright and Ziegler, 2017) are:

- B as the number of grown trees. Note that this parameter is usually not tuned since it is known that more trees are better.
- The cardinality of the sample of features at every node is m_{try}.
- The minimum number of observations that each terminal node should contain (stopping criteria).

Though there exist other variants of bagged tree-based ensembles (Geurts et al., 2006; Goehry et al., 2023), we concentrate on the Random Forest as it is the best known method that is often seen as the machine learning benchmark procedure (e.g. Pórtoles et al., 2018). In addition, Random Forests have also been frequently used for time series forecasting (Huang et al., 2020; Kane et al., 2014). For example, in Salari et al. (2022), a Random Forest approach was used to model real-time delivery time forecasts in online retailing while Vairagade et al. (2019) applied Random Forest to predict product demand for grocery items.

While machine learning methods are quite en vogue, we should not neglect the advantages of time series methods in terms of interpretability. Here, time series approaches enable a clearer understanding of the factors influencing the predictions.

3. Simulation Set-up

In our simulation study, we compare the one-step forecast prediction performance of the methods described in Section 2. All simulations were conducted in the statistical computing software `R` (R Core Team, 2022). We use the `forecast` package (Hyndman and Khandakar, 2008) for all time series approaches under consideration. For the machine learning methods, we used the `ranger` (Wright and Ziegler, 2017) and `xgboost`(Chen et al., 2022) packages for Random Forest and XGBoost, respectively. The concrete simulation settings and data generating processes (DGPs) are described below.

Data Generating Processes. We consider twelve DGPs in total - an autoregressive model (AR), two bilinear models (BL), two nonlinear autoregressive models (NAR), a nonlinear moving average model (NMA), two sign autoregressive models (SAR), two smooth transition autoregressive models (STAR) and two TAR models. They are summarized in Table 1, where the error terms ε_t are independent and identically distributed with a standard normal distribution.

Table 1: Data generating processes (DGPs) used in the simulation study. The error terms ε_t are i.i.d $\mathcal{N}(0,1)$.

Model Type	Variant(s)	Data generating process
Autoregressive	AR	$x_t = 0.5x_{t-1} + 0.45x_{t-2} + \varepsilon_t,$
Bilinear	BL 1	$x_t = 0.7x_{t-1} \cdot \varepsilon_{t-2} + \varepsilon_t,$
	BL2	$x_t = 0.4x_{t-1} - 0.3x_{t-2} + 0.5x_{t-2} \cdot \varepsilon_{t-1} + \varepsilon_t,$
Nonlinear Autoregressive	NAR 1	$x_t = \frac{0.7\lvert x_{t-1}\rvert}{\lvert x_{t-1}\rvert+2} + \varepsilon,$
	NAR2	$x_t = \frac{0.7\lvert x_{t-1}\rvert}{\lvert x_{t-1}\rvert+2} + \frac{0.35\lvert x_{t-2}\rvert}{\lvert x_{t-2}\rvert+2} + \varepsilon,$
Nonlinear Moving Average	NMA	$x_t = \varepsilon_t - 0.3\varepsilon_{t-1} + 0.2\varepsilon_{t-2} + 0.4\varepsilon_{t-1}\varepsilon_{t-2} - 0.25\varepsilon_{t-2}^2,$
Sign Autoregressive	SAR 1	$x_t = \text{sign}(x_{t-1}) + \varepsilon_t,$
	SAR 2	$x_t = \text{sign}(x_{t-1} + x_{t-2}) + \varepsilon_t,$
Smooth Transition	STAR 1	$x_t = 0.8\varepsilon_t - \frac{0.8\varepsilon_{t-1}}{1+\exp(-10x_{t-1})} + \varepsilon_t,$
Autoregressive	STAR 2	$x_t = 0.3x_t + 0.6x_{t-2} + \frac{0.1-0.9x_{t-1}+0.8x_{t-2}}{1+\exp(-10x_{t-1})} + \varepsilon_t,$
Threshold Autoregressive	TAR 1	$x_t = \begin{cases} 0.9x_{t-1} + \varepsilon_t & \text{if } \lvert x_{t-1}\rvert \leq 1 \\ -0.3x_{t-1} - \varepsilon_t & \text{if } \lvert x_{t-1}\rvert > 1 \end{cases}$
	TAR 2	$x_t = \begin{cases} 0.9x_{t-1} + 0.05x_{t-2} + \varepsilon_t & \text{if } \lvert x_{t-1}\rvert \leq 1 \\ -0.3x_{t-1} + 0.65x_{t-2} - \varepsilon_t & \text{if } \lvert x_{t-1}\rvert > 1. \end{cases}$

Similar models have been used to evaluate time series forecasts (Zhang et al., 2001a) and are of importance in data-driven logistics. In particular, autoregressive models (AR, NAR1, NAR2) were chosen to capture the persistence observed in historical logistics demand (Luong, 2007). Bilinear models (BL1, BL2) reflect the complex interactions within logistics networks where different components contribute to the observed patterns. The non-linear moving average (NMA) model is suitable for scenarios with complex interdependencies between multiple factors. Sign autoregressive models (SAR1, SAR2) are suitable for situations in which events or conditions have a directional influence on future events. Smooth transition autoregressive models (STAR1, STAR2) mimic logistics systems where demand changes gradually due to external factors (Ubilava, 2012) and threshold autoregressive models (TAR1, TAR2) represent logistics processes with different regimes based on specific conditions (Ricky Rambharat et al., 2005). This diverse set of DGPs depicts many aspects of the multi-layered nature of logistics data, which includes persistence, interactions, complicated dependencies, directional influences, smooth transitions and different regimes. In the absence of comprehensive benchmark problems, this set-up allows us to evaluate the adaptability of forecasting methods in dynamic logistics scenarios.

Additional Complexities. To add additional complexity to the analysis, we have incorporated settings with a jump process and a random walk (Shumway et al., 2000) into each DGP. The jump process introduces sudden regime changes (which may occur in logistics due to unforeseeable events), while the random walk adds noise to the data (which may occur in logistics settings with increased complexity or less accurate measurements). Thus, our study considers four different scenarios: (1) the DGP without additional complexity, (2) the DGP superposed with the jump process, (3) the DGP superposed with random noise, and (4) the DGP superposed with both the jump process and random noise. The *jumps* are modeled using a compound Poisson process $\{p_t\}_t$ (Kingman, 1992). The original DGP $\{x_t\}_t$ is then superposed by p_t as follows

$$x_t^* = x_t + p_t,$$

where x_t^* denotes the resulting DGP, and the compound Poisson process is given by

$$p_t = \sum_{i=1}^{N_t} Z_i,$$

where N_t follows a Poisson distribution with parameter λ and $Z_i \sim \mathcal{N}(0, \sigma_p^2)$. For the jump experiments we set σ_p^2 to 1. A larger σ_p^2 results in larger jumps in magnitude, while the mean over positive and negative jumps remains zero. The parameter λ is set to $\frac{n}{10}$, where n denotes the length of the generated time series. This means that, on average, a jump is expected to occur after every λ period. Superposing the DGP with the compound Poisson process results in a mean shift by the actual jump size that occurred at each jump event. As mentioned before, the noise is modeled by a *random walk* $\{w_t\}_t$ with

$$w_t = w_{t-1} + e_t,$$

where $e_t \sim \mathcal{N}(0, \sigma_{rw}^2)$. In our study, we choose σ_{rw}^2 in such a way that we obtain a setting with medium noise, i.e., a signal-to-noise ratio (SNR) of four. The SNR (Box, 1988) is a measure that characterizes the strength of the signal relative to the background noise. A higher SNR indicates a clearer and more discernible signal amidst the noise. By including the random walk, we achieve a resulting DGP that is globally nonstationary due to the random walk overlay.

Additional Queueing Models. Beyond these 48 simulation models, we include the M/M/1 and M/M/2 queueing models (Cooper, 1981) in our study. Queueing models are commonly used in logistics, operations research and industrial engineering to study the behavior of waiting lines or queues (Artalejo and Lopez-Herrero, 2001; Schwarz et al., 2006; Kobayashi and Konheim, 1977; Gautam, 2012). Both models have numerous real-world applications, such as in call centers (Brown et al., 2005), healthcare facilities (Green, 2006), and transportation systems (Radmilovic et al., 1996). The M/M/1 model is a classic queueing model that assumes a single queue and one server. It is a stochastic model, where customer arrivals are assumed to follow a Poisson process, and service times are exponentially distributed. The M/M/1 model can be used to analyze the expected waiting time, the number of customers in the queue, and the expected server utilization. The M/M/2 model is a variation of the M/M/1 model that assumes two parallel servers. According to Gautam (2012), we set the arrival rate to four and the service rate to two.

Number of different Settings. For each setting, we generate time series of length n from the respective DGPs with $n \in \{100, 500, 1000\}$. In total, this results in 150 (= 12 (time series DGPs) $\times$ 4 (further complexity)+2 queueing models) $\times$ 3 (lengths)) different simulation settings for each forecasting method.

Data Preprocessing. To forecast time series using a machine learning algorithm, the sliding window approach (Dietterich, 2002) is used. In this approach, a fixed-sized window is moved over the time series data, and at each step, the data within the window is used as input to a machine learning algorithm for prediction. One advantage of the sliding window approach is that it allows the machine learning algorithm to capture the temporal dependencies and patterns in the data. The window size is an important parameter in this approach (Savva et al., 2020). If the window size is too small, it may not capture the relevant information in the data, while if it is too large, it may introduce unnecessary noise and reduce the accuracy of the model. We consider sliding window sizes of 2, 4, 8 and 16 and study which size is best suited for the different time series lengths 100, 500 and 1000. Furthermore, in machine learning-based time series forecasting, we explore two approaches: one using the original time series and the other using the differentiated time series as input. The latter is essential as trees cannot forecast outside the range observed so far and to enhance stationarity in the time series.

Choice of Parameters. In order not to have to discuss the different possibilities for hyperparameter tuning of the machine learning algorithm, we use the default values recommended in the literature (Breiman, 2001; Wright and Ziegler, 2017; Hastie et al., 2009). This has the additional advantage of a reduced runtime. Thus, each ensemble learner consists of 500 trees, the inner bootstrap sample is equal to $m_{try} = \lfloor \frac{p}{3} \rfloor$, where p denotes the number of features, the number of sample points in the bagging step is equal to the sample size. Each terminal node should at least contain five observations. For XGBoost, we use a learning rate of 0.3 and a maximal depth of 6. To estimate the parameters of the time series approaches, we use the algorithms implemented in the R-package `forecast`.

Evaluation Measure. Since the mean square error (MSE) and the mean absolute percentage error (MAPE) are widely used in the forecasting of time series in logistics (Kuhlmann and Pauly, 2023), we use them as evaluation measures, which are calculated over 1,000 repeated forecasting steps. The MSE measures the model's accuracy, expressed as the average squared difference between observed and predicted values. Simultaneously, the MAPE, calculated as the average percentage difference between observed and predicted values, offers insights into the model's relative performance.

4. Results

In this section, we describe the results of the simulation study. In particular, we present the MSE of the different forecasting algorithms under various simulation configurations. The analysis of the MAPE results can be found in the Appendix. We start with the performance of the methods for queueing models.

4.1. Predictive Power in Queueing Models

The influence of the different sliding window sizes and the differentiation is shown in Figures 1 and 2.

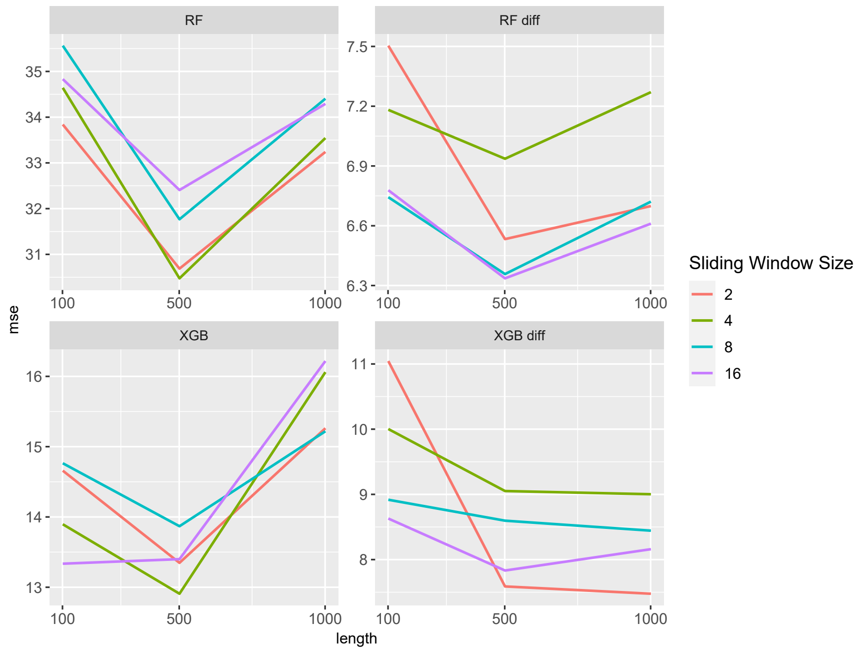


Figure 1: MSE of ML approaches separated by the sliding window size for the M/M/1 setting. XGB stands for XGBoost and RF for Random Forest; diff in the method name indicates that the data were differentiated.

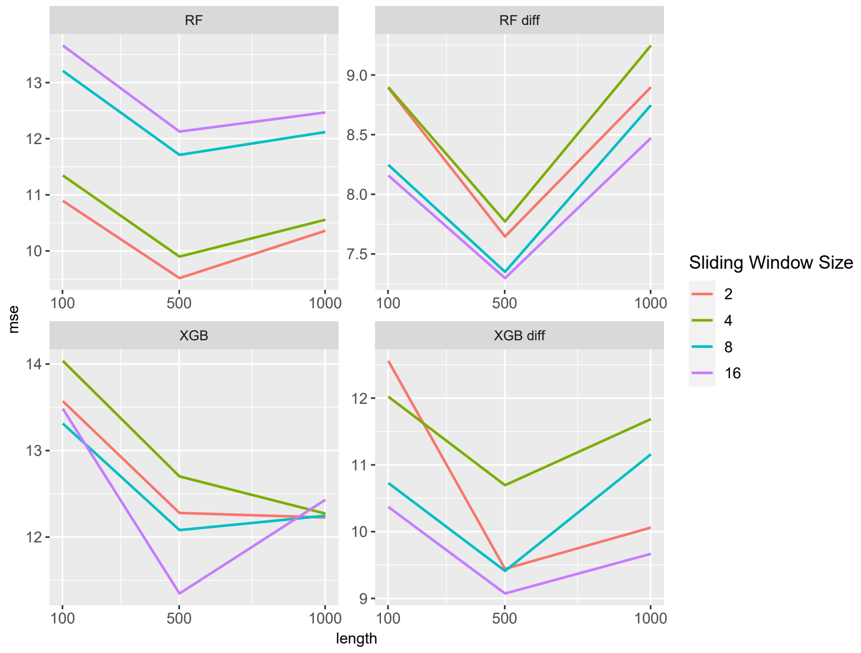


Figure 2: MSE of ML approaches separated by the sliding window size for the M/M/2 setting. XGB stands for XGBoost and RF for Random Forest; diff in the method name indicates that the data were differentiated.

Generally, differentiation improves the prediction power of both ML approaches in both settings. Especially for the Random Forest, the MSE decreases by one-fifth after differentiation. The lengths of the time series only have a minor influence on the MSE. The Random Forest with differentiated data outperformed the other methods for all lengths. Comparing the effects of sliding window sizes, we find slight differences in performance. Random Forests have smaller MSE values with smaller sliding windows in both settings, while larger window sizes slightly improve performance in the other approaches.

The predictive power of the time series and naive approaches are given in Figure 3. Note that both ARIMA and SARIMA models have identical MSE values. In both cases, the time series approach performs better than the naive approach. However, the difference in performance is smaller for M/M/2. Again, the influence of the time series length is marginal. While all time series approaches perform similarly in the M/M/1 setting, the TBATS method has slightly smaller values in the M/M/2 setting.

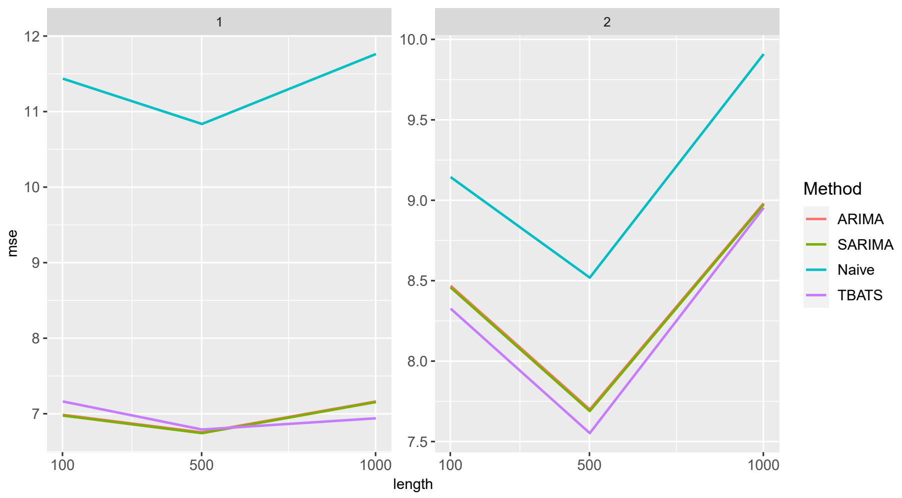


Figure 3: MSE of time series and naive approaches for the M/M/1 (left) and M/M/2 (right) setting. ARIMA and SARIMA models have identical MSE values, as no seasonality was present.

In both scenarios, the Random Forest approach with differenced data consistently showed the smallest MSE. However, the differences between this method and the time series approaches were not great.

4.2. Predictive Power in the Different Time Series Settings

In the following, we analyze the performance of the methods for the DGPs described in Table 1. When comparing the influence of sliding window size and differentiation on the performance of Random Forest across all settings (Figure 4), we observed that non-differentiation resulted in smaller MSE values except for the AR setting. In the AR setting, differentiation slightly outperformed non-differentiation. However, it should be noted that as the length of the time series increases, the differences between the two approaches become negligible. In all settings, the MSE values slightly decrease with an increase in time series length. The sliding window size has a small influence on the prediction power and shows similar behavior across different time series lengths.

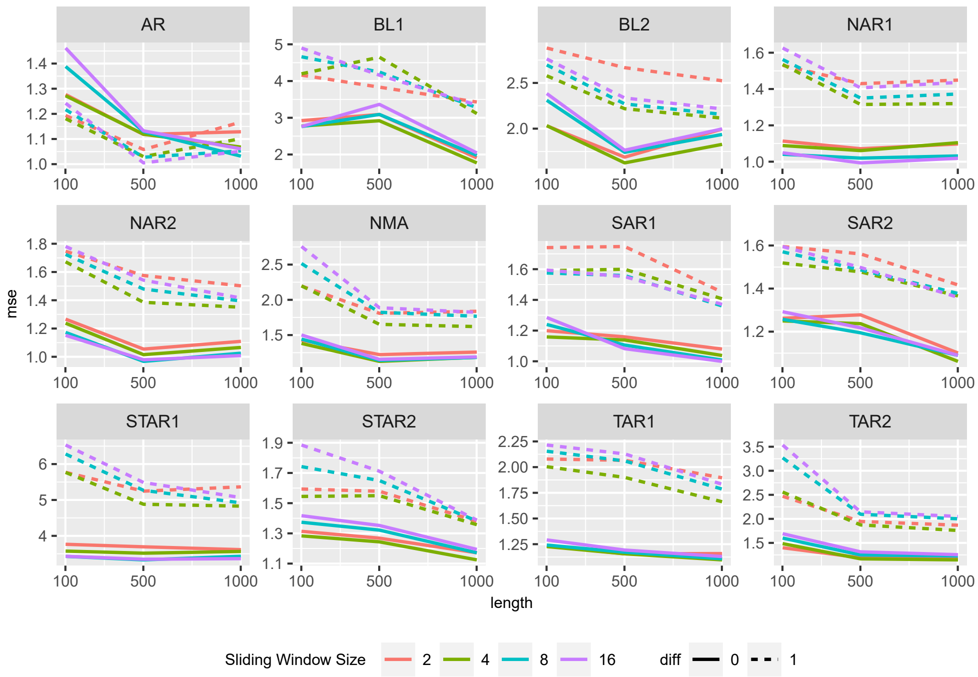


Figure 4: MSE of the Random Forest approaches separated by the sliding window size and differentiation for the different data generating processes.

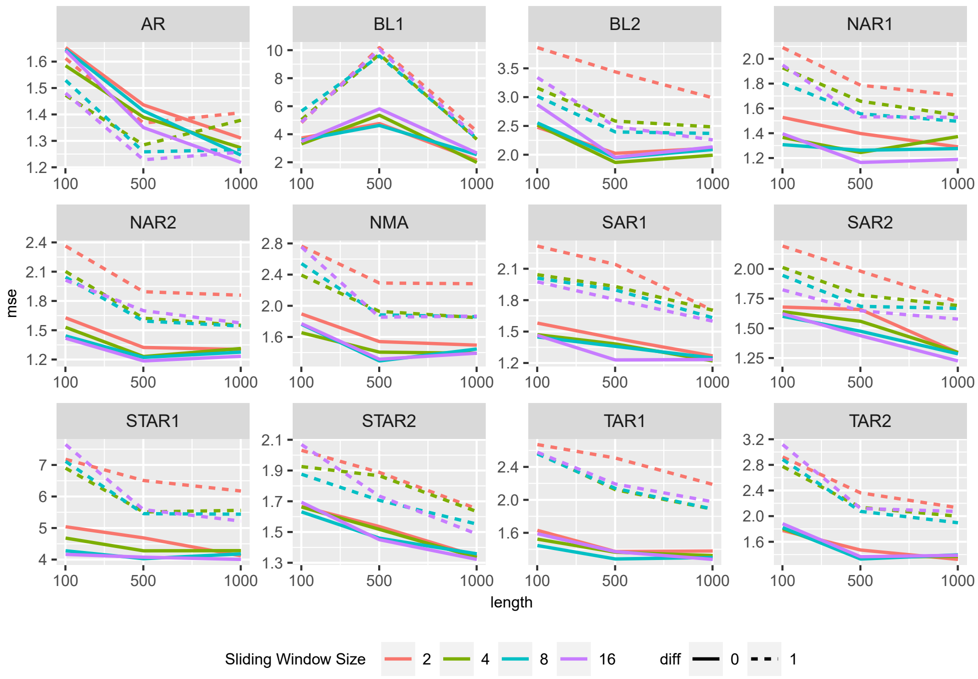


Figure 5: MSE of XGBoost approaches separated by the sliding window size and differentiation for the different data generating processes.

Similar observations can be made for XGBoost, see Figure 5. The sliding window's size and the time series length have a small effect on the performance quality. For all DGPs, the MSE values decrease slightly with increasing time series length, except for BL1. Here, the MSE values first increase. The XGBoost approaches generally have slightly larger MSE values than the Random Forest approaches.

Figure 6 shows the MSE values for the time series approaches. The performance of the time series approaches is comparable to that of the Random Forest. All methods have very similar MSE values. The time series length has only a minor impact on the predictive power, except for the BL1 setting. As observed for the XGBoost approaches, MSE values in this setting first increase and then decrease with increasing time series length.

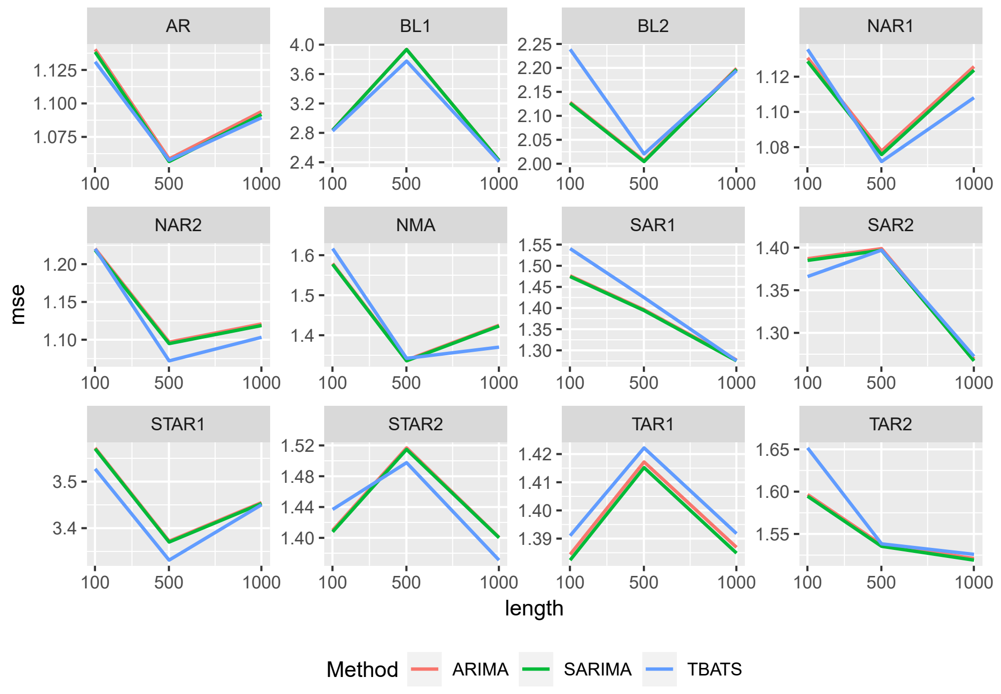


Figure 6: MSE of the time series approaches for the different data generating processes.

Additional results can be found in the Appendix. Figure A.10 therein, e.g., shows that the naive approach exhibits the largest MSE values compared to all methods. Thereby the performance of the naive approach is dependent on the DGP and the length of the time series. For BL2, longer time series lengths generally lead to better performance, but for NAR1 the performance may slightly decrease. For AR, BL1, and NMA models, the MSE values typically decrease initially and then slightly increase as the time series length increases. Conversely, NAR2, SAR1, SAR2, STAR1, STAR2, TAR1 and TAR2 tend to show the opposite trend.

4.3. Influence of the Additional Complexities on the Predictive Power

Based on the findings of the previous sections, we focus on the simulation results obtained with a sliding window size of 8, as the choice of this size is due to the consistent performance observed with different sizes. Details of the results with other window sizes can be found in the Appendix, but a moderate size of 8 balances computational efficiency and information incorporation. Below we first consider the influence of an additional jump process before discussing the white noise results.

The influence of the jumping process can be seen in Figure 7. All MSE values increase monotonically with increasing sample size, indicating that the jump process significantly impacts predictive performance. Note that as time series length increases, the Random Forest approach with differentiated data outperforms all other approaches. Using the differenced data significantly improves the MSE values for both ML approaches, particularly for increasing time series length. The predictive performance of the time series approaches is similar for all DGPs and slightly better than that of the naive approach.

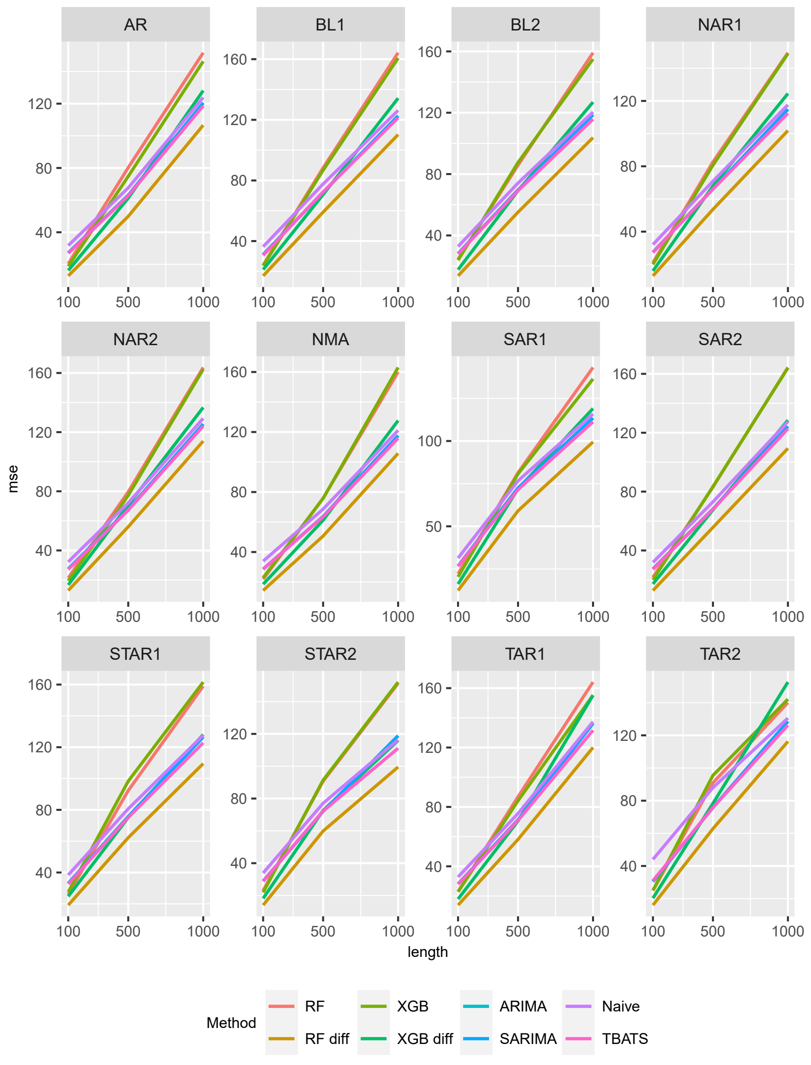


Figure 7: MSE values of all methods and data generating processes superposed by a compound Poisson process.

Figure 8 summarizes the prediction results for all methods and all DGPs superposed by a random walk.

Here, the time series length has only a minor influence on the prediction performance of the data overlaid

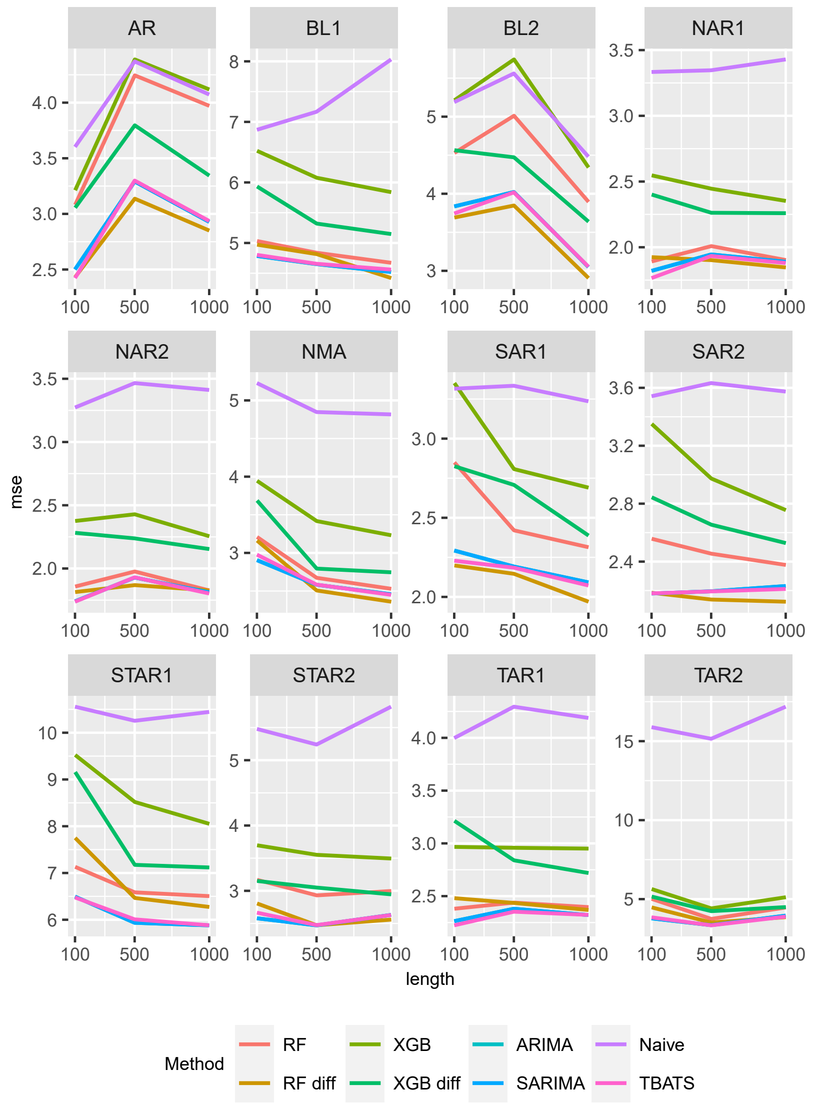


Figure 8: MSE values of all methods and data generating processes superposed by a random walk.

with a random walk. For the AR and BL2 settings, the MSE values increase slightly when the time series length is increased from 100 to 500. For all other DGPs, the MSE values decrease slightly, except for the naive approach. The naive approach has the highest MSE values for all settings, followed by XGBoost, except for BL2. Here, both approaches have similar values. The performance of the other methods depends on the respective setting.

For the settings, AR, BL2, SAR1 and SAR2, Random Forest with differenced data again shows the smallest MSE values, while the time series approaches show slightly larger values. Note that the XGBoosts with differentiated data perform better in these settings than the Random Forests with non-differentiated data. In the BL1, NAR1, NAR2, NMA and STAR2 settings, only minor differences in the performance of the Random Forests and time series approaches can be observed. When comparing the two XGBoost approaches in these settings, the differentiation reduces the MSE. The ML approaches show larger MSE values in the STAR1, TAR1 and TAR2 settings than the time series approaches, with Random Forests performing better than the XGBoost method.

The influence of both complexities, the random walk and the Poisson process, on the prediction performance is shown in Figure A.15 in the Appendix. Similar to the case where a composite Poisson process is superposed on the data, we observe an increase in MSE values with increasing time series length for all settings. In particular, for time series lengths of 500, we obtain MSE values of more than 2,000.

4.4. Summarizing all Results

To evaluate the prediction performance across the spectrum of simulation settings, we calculate the median rank for each prediction method in Table 2. The ranking is based on the MSE values, with rank 1 indicating the method with the lowest MSE. Each entry in the table represents the median rank of a particular prediction method in all settings of a particular DGP model described in Section 2. Furthermore, the results for the ranking take into account the performance of machine learning algorithms with a sliding window size of 8.

Table 2: Median performance rank of forecasting methods across different simulation settings and different time series lengths. Rankings are based on MSE values, with rank 1 indicating the method with the lowest MSE.

Method	Queueing	DGPs (Table 1) with			
	models	no add. complexity	jump	random walk	both
Random Forest	7	1	7	5	7
Random Forest Diff	1	6	1	1	1
XGBoost	7	5	7	7	7
XGBoost Diff	5	7	5	6	6
ARIMA	2.5	3	3	3	3
SARIMA	2.5	3	3	3	3
TBATS	3	3	3	3	3
Naive	6	8	6	8	5

The results in Table 2 provide useful insights into the relative predictive performance of the different methods in different simulation scenarios. In particular, Random Forest with differentiated inputs proves to be the best performing method, achieving the lowest median value across different complexities, including scenarios with jumps, random walks or a combination of both. While XGBoost is competitive, it tends to have a slightly higher median value under these conditions. Traditional time series methods such as ARIMA, SARIMA and TBATS consistently show robust and similar performance.

5. Real-World Data Example

As explained at the onset, there is a lack of freely available and good documented data sets in logistics research. We therefore use a rather simple real-world data example for illustration. The data set contains daily demand orders from a Brazilian logistics company (Ferreira et al., 2017) and was sourced from the UCI

Machine Learning Repository (Dua and Graff, 2017). Covering a span of 60 consecutive days, the data set consists of three time series that capture orders for products A, B, and C. Figure 9 shows the corresponding time series in which specific shocks in the data can be identified. This observation puts us in a similar setting to the simulation study where the DGP was overlaid with a Poisson process. Given this context, it is of interest to evaluate whether the robust performance of (differentiated) machine learning algorithms observed in the simulation study is also apparent in for this dataset.

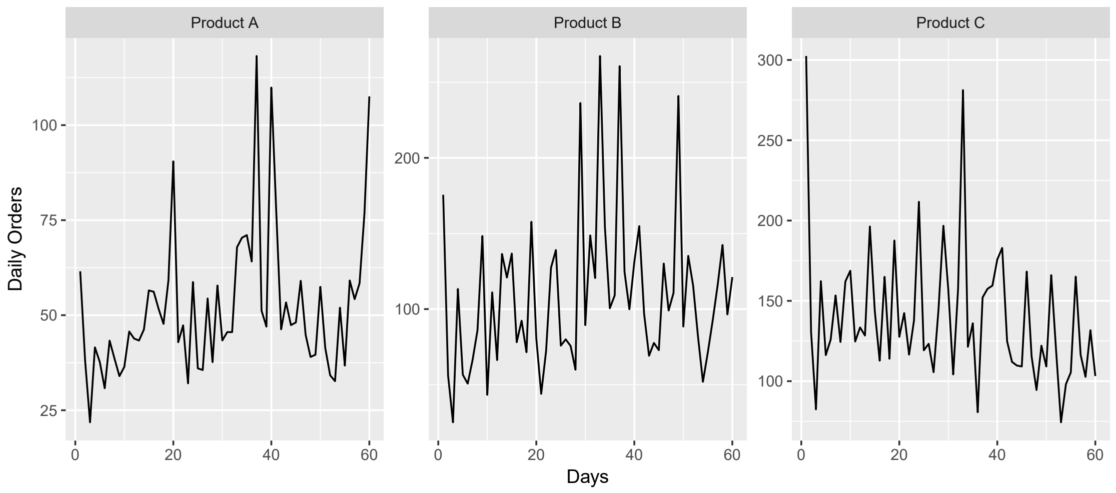


Figure 9: Daily orders of a Brazilian logistics company separated by the different products.

The machine learning algorithms adhere to the hyperparameters outlined in Section 3, with a sliding window size of eight, as informed by insights from our simulation study. We use the first 50 observations to train all methods and the last ten observations to test the performance via time series cross validation (Hyndman and Athanasopoulos, 2018, Chapter 5.10). The MSE and MAPE are again used as evaluation measures. The summarized results are presented in Table 3. Note that the results of SARIMA and ARIMA are identical due to the absence of seasonality and are therefore combined into one method.

Table 3: Mean MAPE and MSE of the methods considered in Section 2 using daily demand order data set.

	MAPE			MSE		
Method	Prod. A	Prod. B	Prod. C	Prod. A	Prod. B	Prod. C
Random Forest	24.30	35.05	30.79	22.39	262.41	695.70
Random Forest Diff	6.67	21.80	15.84	4.91	197.23	1.97
XGBoost	25.06	41.62	19.51	22.34	376.62	147.20
XGBoost Diff	10.70	37.98	27.15	13.10	841.56	41.00
(S)ARIMA	28.57	49.30	33.56	29.48	1,142.14	655.88
TBATS	28.37	36.17	33.56	43.14	446.18	663.78
Naive	33.18	30.71	30.59	25.10	194.21	82.03

The results show that the performance of the forecasting methods is different in the various product categories. In general, the machine learning algorithms deliver consistently better results than the traditional time series methods. This is in line with our simulation study, where ML methods showed better performances when additional complexities were present. Random Forest with differentiation performed best for all three time series and evaluation measures, again confirming the results obtained in the simulation study for such settings. It should be noted that the introduction of differentiation is beneficial for Random Forest in all predictions. For XGBoost, however, performance on product A improves significantly when differenced data is used, but in the other two time series differentiation leads to worse forecasting performance.

6. Summary, Discussion and Outlook

Summary with Higlights. The main objective of this simulation study was to perform a one-step comparative analysis of prediction accuracy and evaluate the performance of tree-based machine learning and time series approaches that are typically used in data-driven logistics. Through a comprehensive investigation of different data generating processes, queueing models, and additional complexities, we aimed to determine each method's inherent strengths and limitations. Our analysis included conventional time series methods, including (seasonal) ARIMA models and TBATS, as well as machine learning methods such as Random Forest and XGBoost. In addition, we investigated the impact of data differencing on the performance of the two latter algorithms. The key findings from our study are as follows:

- The out-of-the-box Random Forest emerged as the ML benchmark method.
- Training on differentiated time series can significantly improve the ML resilience.
- ML models are more robust with respect to additional (nonlinear) complexity, settings in which they outperformed statistical time series approaches.
- In all other settings, the time series approaches were at least competitive or even performed better.

Detailed Discussion and Outlook. In our study, the Random Forest approach performed consistently better in all simulation settings than the XGBoost approaches. It is worth noting that no hyperparameter tuning was made in our study. Random Forests are known to be robust to hyperparameter settings and often perform well with default values (Probst et al., 2019; Fernández-Delgado et al., 2014). This robustness can be a crucial factor contributing to their superior performance compared to XGBoost. Applying techniques such as Bayesian Optimization or more simple grid or random search for hyperparameter tuning could change this observation and should be investigated in future studies. Regarding the effect of data differentiation on the performance of the two machine learning methods, we observed similar patterns. Differentiation improved performance, especially in queueing scenarios and situations where additional complexity was introduced into the data generation process. Without additional complexity, differentiation showed minimal impact, with the performance of both methods deteriorating slightly when the differentiated data was used, except for very linear data generation processes. Here, only a slight improvement was observed. This suggests that differentiation plays a crucial role in improving the resilience of machine learning methods, especially Random Forests when the data is overlaid with additional noise like a random walk. When comparing the performance of the different time series approaches, we found subtle differences between them. ARIMA and SARIMA showed relatively similar performance in all simulation settings under consideration. Their prediction accuracy was quite consistent without big differences in most situations. Comparing their performance with that of TBATS, the differences are also small and not substantial, suggesting that ARIMA, SARIMA and TBATS had comparable predictive power in our simulation settings. The additional complexity induced, such as a jump process or random noise, significantly impacts the predictive power. Introducing a jump process leads to increased MSE values for all methods and settings, indicating a significant impact on prediction accuracy. In this scenario, all methods show consistent behavior with strong increasing MSE values for increasing time series lengths. When a noise process is introduced, a more nuanced pattern emerges. For the machine learning approaches, differentiating the data proves beneficial and improves the overall performance. The Random Forest approach with differenced data as input outperforms the other approaches in most scenarios, closely followed by all three time series approaches. A comparison between Random forests and the time series approaches shows different performance patterns in the different simulation environments. In queueing situations, where the underlying processes are often characterized by complicated dynamics, the Random Forest approach shows superior performance. Furthermore, a notable trend emerges in simulation settings where a Poisson process complements the data generating processes. In these cases, ML methods show improved performance, indicating robustness to the inherent complexity introduced by the Poisson process. The adaptability of ML models to capture and learn from nonlinear patterns may contribute to their effectiveness in scenarios with Poisson process or random walk overlays. However, it is essential to recognize that this beneficial performance of ML methods is not universal. In all other simulation settings, the Random Forest approaches perform comparable or slightly worse than all three time series approaches. In addition to the simulation study, our illustrative data analyses were conducted with a focus on one-step demand forecasting for different products of a logistics company. The results indicate that machine learning algorithms can improve the forecasting performance in this context. In particular, the machine learning methods perform better or equally well as the time series methods for most products.

In the context of data-driven logistics, our results underscore the importance of tailoring time series forecasting methods to the specific characteristics of data sets encountered in different logistics areas. The Random Forest approach, especially when using differentiated data as input, is recommended as an initial benchmark prediction tool, particularly for data sets with a lot of noise or complex patterns. The robustness of Random Forests, combined with their ability to achieve good results without extensive tuning of hyperparameters, makes them a pragmatic choice for various prediction scenarios. Conversely, in situations where interpretability is paramount (e.g., to gain understanding or trust of users in warehouses or decision makers in SCM) and the data exhibit clear patterns, traditional time series approaches remain a valuable and interpretable option. These approaches often come with faster runtimes and greater resource efficiency, which is also essential in the development of data-driven logistics, e.g. in case of resource constraints (Venkatapathy et al., 2015; Gouda et al., 2023). As only one-step forecasts were considered, future simulation studies should investigate whether the same observations can be found for more step forecasting. Also, additional or hybrid methods must be investigated (Aladag et al., 2009; Zhang, 2003; Smyl, 2020). Another line of future research needs to compare the methods with respect to uncertainty quantification, i.e., point-wise or simultaneous prediction intervals and regions.

References

Aguilar Madrid, E., Antonio, N., 2021. Short-term electricity load forecasting with machine learning. Information 12, 50.

Ahmed, N.K., Atiya, A.F., Gayar, N.E., El-Shishiny, H., 2010. An empirical comparison of machine learning models for time series forecasting. Econometric reviews 29, 594–621.

Al-Saba, T., El-Amin, I., 1999. Artificial neural networks as applied to long-term demand forecasting. Artificial Intelligence in engineering 13, 189–197.

Aladag, C.H., Egrioglu, E., Kadilar, C., 2009. Forecasting nonlinear time series with a hybrid methodology. Applied Mathematics Letters 22, 1467–1470.

Alim, M., Ye, G.H., Guan, P., Huang, D.S., Zhou, B.S., Wu, W., 2020. Comparison of arima model and xgboost model for prediction of human brucellosis in mainland china: a time-series study. BMJ open 10, e039676.

Ampountolas, A., 2021. Modeling and forecasting daily hotel demand: A comparison based on sarimax, neural networks, and garch models. Forecasting 3, 580–595.

Arora, K., Abbi, P., Gupta, P.K., 2022. Analysis of supply chain management data using machine learning algorithms, in: Innovative Supply Chain Management via Digitalization and Artificial Intelligence. Springer, pp. 119–133.

Artalejo, J.R., Lopez-Herrero, M., 2001. Analysis of the busy period for the m/m/c queue: An algorithmic approach. Journal of applied probability 38, 209–222.

Awasthi, S., Fernandez-Cortizas, M., Reining, C., Arias-Perez, P., Luna, M.A., Perez-Saura, D., Roidl, M., Gramse, N., Klokowski, P., Campoy, P., 2023. Micro uav swarm for industrial applications in indoor environment – a systematic literature review. Logistics Research .

Baryannis, G., Dani, S., Antoniou, G., 2019. Predicting supply chain risks using machine learning: The trade-off between performance and interpretability. Future Generation Computer Systems 101, 993–1004.

Benvenuto, D., Giovanetti, M., Vassallo, L., Angeletti, S., Ciccozzi, M., 2020. Application of the arima model on the covid-2019 epidemic dataset. Data in brief 29, 105340.

Bontempi, G., Ben Taieb, S., Borgne, Y.A.L., 2012. Machine learning strategies for time series forecasting, in: European business intelligence summer school, Springer. pp. 62–77.

Box, G., 1988. Signal-to-noise ratios, performance criteria, and transformations. Technometrics 30, 1–17.

Box, G.E., Cox, D.R., 1964. An analysis of transformations. Journal of the Royal Statistical Society Series B (Methodology) 26, 211–243.

Box, G.E., Jenkins, G.M., Reinsel, G.C., Ljung, G.M., 2015. Time series analysis: forecasting and control. John Wiley & Sons.

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

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 2017. Classification and Regression Rrees. Routledge.

Brockwell, P.J., Davis, R.A., 2002. Introduction to time series and forecasting. Springer.

Brown, L., Gans, N., Mandelbaum, A., Sakov, A., Shen, H., Zeltyn, S., Zhao, L., 2005. Statistical analysis of a telephone call center: A queueing-science perspective. Journal of the American statistical association 100, 36–50.

Carbonneau, R., Laframboise, K., Vahidov, R., 2008. Application of machine learning techniques for supply chain demand forecasting. European Journal of Operational Research 184, 1140–1154.

Chen, T., Guestrin, C., 2016. Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery. pp. 785–794.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., Li, M., Xie, J., Lin, M., Geng, Y., Li, Y., Yuan, J., 2022. xgboost: Extreme Gradient Boosting. URL: https://CRAN.R-project.org/package=xgboost. r package version 1.6.0.1.

Cooper, R.B., 1981. Queueing theory, in: Proceedings of the ACM'81 conference, pp. 119–122.

De Gooijer, J.G., Kumar, K., 1992. Some recent developments in non-linear time series modelling, testing, and forecasting. International Journal of Forecasting 8, 135–156.

De Livera, A.M., Hyndman, R.J., Snyder, R.D., 2011. Forecasting time series with complex seasonal patterns using exponential smoothing. Journal of the American statistical association 106, 1513–1527.

Dietterich, T.G., 2002. Machine learning for sequential data: A review, in: Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops SSPR 2002 and SPR 2002 Windsor, Ontario, Canada, August 6–9, 2002 Proceedings, Springer. pp. 15–30.

Dua, D., Graff, C., 2017. Uci machine learning repository. URL: http://archive.ics.uci.edu/ml.

Ensafi, Y., Amin, S.H., Zhang, G., Shah, B., 2022. Time-series forecasting of seasonal items sales using machine learning–a comparative analysis. International Journal of Information Management Data Insights 2, 100058.

Fan, D., Sun, H., Yao, J., Zhang, K., Yan, X., Sun, Z., 2021. Well production forecasting based on arima-lstm model considering manual operations. Energy 220, 119708.

Feizabadi, J., 2022. Machine learning demand forecasting and supply chain performance. International Journal of Logistics Research and Applications 25, 119–142.

Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D., 2014. Do we need hundreds of classifiers to solve real world classification problems? Journal of Machine Learning Research 15, 3133–3181.

Ferreira, R., Martiniano, A., Ferreira, A., Ferreira, A., Sassi, R., 2017. Daily Demand Forecasting Orders. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5BC8T.

Francq, C., Zakoian, J.M., 2019. GARCH models: structure, statistical inference and financial applications. John Wiley & Sons.

Friedman, J.H., 2002. Stochastic gradient boosting. Computational statistics & data analysis 38, 367–378.

Friedrich, S., Friede, T., 2023. On the role of benchmarking data sets and simulations in method comparison studies. Biometrical Journal , 2200212.

Garrido, R.A., Mahmassani, H.S., 2000. Forecasting freight transportation demand with the space–time multinomial probit model. Transportation Research Part B: Methodological 34, 403–418.

Gautam, N., 2012. Analysis of queues: methods and applications. CRC Press.

Geurts, P., Ernst, D., Wehenkel, L., 2006. Extremely randomized trees. Machine learning 63, 3–42.

Gilbert, K., 2005. An ARIMA supply chain model. Management Science 51, 305–310. URL: `http://search.ebscohost.com/login.aspx?direct=true&db=buh&AN=16292291&site=ehost-live`, doi:`10.1287/mnsc.1040.0308`. publisher: INFORMS: Institute for Operations Research.

Goehry, B., Yan, H., Goude, Y., Massart, P., Poggi, J.M., 2023. Random forests for time series. REVSTAT-Statistical Journal 21, 283–302.

Gouda, A., Heinrich, D., Hünnefeld, M., Priyanta, I.F., Reining, C., Roidl, M., 2023. A grid-based sensor floor platform for robot localization using machine learning, in: 2023 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), IEEE. pp. 1–6.

Green, L., 2006. Queueing analysis in healthcare. Patient flow: reducing delay in healthcare delivery , 281–307.

Hasmin, E., Aini, N., 2020. Data mining for inventory forecasting using double exponential smoothing method, in: 2020 2nd International Conference on Cybernetics and Intelligent System (ICORIS), pp. 1–5. doi:`10.1109/ICORIS50180.2020.9320765`.

Hastie, T., Tibshirani, R., Friedman, J.H., Friedman, J.H., 2009. The elements of statistical learning: data mining, inference, and prediction. volume 2. Springer.

Huang, H., Pouls, M., Meyer, A., Pauly, M., 2020. Travel time prediction using tree-based ensembles, in: Computational Logistics: 11th International Conference, ICCL 2020, Enschede, The Netherlands, September 28–30, 2020, Proceedings 11, Springer. pp. 412–427.

Hwarng, H.B., 2001. Insights into neural-network forecasting of time series corresponding to arma (p, q) structures. Omega 29, 273–289.

Hyndman, R.J., Athanasopoulos, G., 2018. Forecasting: principles and practice. OTexts.

Hyndman, R.J., Khandakar, Y., 2008. Automatic time series forecasting: the forecast package for R. Journal of Statistical Software 26, 1–22. doi:`10.18637/jss.v027.i03`.

Islam, S., Amin, S.H., 2020. Prediction of probable backorder scenarios in the supply chain using distributed random forest and gradient boosting machine learning techniques. Journal of Big Data 7, 1–22.

Ji, S., Wang, X., Zhao, W., Guo, D., 2019. An application of a three-stage xgboost-based model to sales forecasting of a cross-border e-commerce enterprise. Mathematical Problems in Engineering 2019.

Kane, M.J., Price, N., Scotch, M., Rabinowitz, P., 2014. Comparison of arima and random forest time series models for prediction of avian influenza h5n1 outbreaks. BMC bioinformatics 15, 1–9.

Kingman, J.F.C., 1992. Poisson processes. volume 3. Clarendon Press.

Kobayashi, H., Konheim, A., 1977. Queueing models for computer communications system analysis. IEEE Transactions on Communications 25, 2–29.

Kohzadi, N., Boyd, M.S., Kermanshahi, B., Kaastra, I., 1996. A comparison of artificial neural network and time series models for forecasting commodity prices. Neurocomputing 10, 169–181.

Kuhlmann, L., Pauly, M., 2023. A dynamic systems model for an economic evaluation of sales forecasting methods. Tehnički glasnik 17, 397–404.

Kuhlmann, L., Wilmes, D., Müller, E., Pauly, M., Horn, D., 2023. Rodd: Robust outlier detection in data cubes. arXiv preprint arXiv:2303.08193 .

Kumar Jha, B., Pande, S., 2021. Time series forecasting model for supermarket sales using fb-prophet, in: 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), pp. 547–554. doi:10.1109/ICCMC51019.2021.9418033.

Lin, H.E., Zito, R., Taylor, M., et al., 2005. A review of travel-time prediction in transport and logistics, in: Proceedings of the Eastern Asia Society for transportation studies, Bangkok, Thailand. pp. 1433–1448.

Luo, J., Zhang, Z., Fu, Y., Rao, F., 2021. Time series prediction of covid-19 transmission in america using lstm and xgboost algorithms. Results in Physics 27, 104462.

Luong, H.T., 2007. Measure of bullwhip effect in supply chains with autoregressive demand process. European Journal of Operational Research 180, 1086–1097.

Ma, Y., Zhang, Z., Ihler, A., Pan, B., 2018. Estimating warehouse rental price using machine learning techniques. International Journal of Computers Communications & Control 13, 235–250.

Mayr, A., Binder, H., Gefeller, O., Schmid, M., 2014. The evolution of boosting algorithms. Methods of information in medicine 53, 419–427.

Morde, V., . Xgboost algorithm: Long may she reign! URL: https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d. "Accesssed 13 Dec 2023".

Ni, D., Xiao, Z., Lim, M.K., 2020. A systematic review of the research trends of machine learning in supply chain management. International Journal of Machine Learning and Cybernetics 11, 1463–1482.

Niemann, F., Reining, C., Moya Rueda, F., Nair, N.R., Steffens, J.A., Fink, G.A., Ten Hompel, M., 2020. Lara: Creating a dataset for human activity recognition in logistics using semantic attributes. Sensors 20, 4083.

Nyoni, T., 2018. Modeling and forecasting inflation in kenya: Recent insights from arima and garch analysis. Dimorian Review 5, 16–40.

Palomares-Salas, J., De La Rosa, J., Ramiro, J., Melgar, J., Aguera, A., Moreno, A., 2009. Arima vs. neural networks for wind speed forecasting, in: 2009 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, IEEE. pp. 129–133.

Pórtoles, J., González, C., Moguerza, J.M., 2018. Electricity price forecasting with dynamic trees: A benchmark against the random forest approach. Energies 11, 1588.

Probst, P., Boulesteix, A.L., Bischl, B., 2019. Tunability: Importance of hyperparameters of machine learning algorithms. Journal of Machine Learning Research 20, 1–32.

R Core Team, 2022. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria. URL: https://www.R-project.org/.

Radmilovic, Z., Colic, V., Hrle, Z., 1996. Some aspects of storage and bulk queueing systems in transport operations. Transportation planning and technology 20, 67–81.

Reining, C., Niemann, F., Moya Rueda, F., Fink, G.A., ten Hompel, M., 2019. Human activity recognition for production and logistics—a systematic literature review. Information 10, 245.

Ribeiro, A.M.N., do Carmo, P.R.X., Endo, P.T., Rosati, P., Lynn, T., 2022. Short-and very short-term firm-level load forecasting for warehouses: a comparison of machine learning and deep learning models. Energies 15, 750.

Ricky Rambharat, B., Brockwell, A.E., Seppi, D.J., 2005. A threshold autoregressive model for wholesale electricity prices. Journal of the Royal Statistical Society Series C: Applied Statistics 54, 287–299.

Salari, N., Liu, S., Shen, Z.J.M., 2022. Real-time delivery time forecasting and promising in online retailing: When will your package arrive? Manufacturing & Service Operations Management 24, 1421–1436.

Savva, A.D., Kassinopoulos, M., Smyrnis, N., Matsopoulos, G.K., Mitsis, G.D., 2020. Effects of motion related outliers in dynamic functional connectivity using the sliding window method. Journal of Neuroscience Methods 330, 108519.

Schapire, R.E., Freund, Y., 2013. Boosting: Foundations and algorithms. Kybernetes 42, 164–166.

Schwarz, M., Sauer, C., Daduna, H., Kulik, R., Szekli, R., 2006. M/m/1 queueing systems with inventory. Queueing Systems 54, 55–78.

Sharma, R., Kamble, S.S., Gunasekaran, A., Kumar, V., Kumar, A., 2020. A systematic literature review on machine learning applications for sustainable agriculture supply chain performance. Computers & Operations Research 119, 104926.

Shi, Y., Guo, X., Yu, Y., 2018. Dynamic warehouse size planning with demand forecast and contract flexibility. International Journal of Production Research 56, 1313–1325.

Shukla, M., Jharkharia, S., 2011. Arima models to forecast demand in fresh supply chains. International Journal of Operational Research 11, 1–18.

Shumway, R.H., Stoffer, D.S., Stoffer, D.S., 2000. Time series analysis and its applications. volume 3. Springer.

Siami-Namini, S., Tavakoli, N., Namin, A.S., 2018. A comparison of arima and lstm in forecasting time series, in: 2018 17th

IEEE international conference on machine learning and applications (ICMLA), IEEE. pp. 1394–1401.

Smyl, S., 2020. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. International Journal of Forecasting 36, 75–85.

Syntetos, A.A., Babai, Z., Boylan, J.E., Kolassa, S., Nikolopoulos, K., 2016. Supply chain forecasting: Theory, practice, their gap and the future. European Journal of Operational Research 252, 1–26.

Taylor, S.J., Letham, B., 2018. Forecasting at scale. The American Statistician 72, 37–45.

Therneau, T.M., Atkinson, E.J., 1997. An Introduction to Recursive Partitioning Using the RPART Routines. Technical Report. Mayo Foundation.

Tsay, R.S., 1989. Testing and modeling threshold autoregressive processes. Journal of the American statistical association 84, 231–240.

Ubilava, D., 2012. Modeling nonlinearities in the us soybean-to-corn price ratio: a smooth transition autoregression approach. Agribusiness 28, 29–41.

Vairagade, N., Logofatu, D., Leon, F., Muharemi, F., 2019. Demand forecasting using random forest and artificial neural network for supply chain management, in: Computational Collective Intelligence: 11th International Conference, ICCCI 2019, Hendaye, France, September 4–6, 2019, Proceedings, Part I 11, Springer. pp. 328–339.

Venkatapathy, A.K.R., Riesner, A., Roidl, M., Emmerich, J., ten Hompel, M., 2015. Phynode: An intelligent, cyber-physical system with energy neutral operation for phynetlab, in: Smart SysTech 2015; European Conference on Smart Objects, Systems and Technologies, VDE. pp. 1–8.

Weber, L.M., Saelens, W., Cannoodt, R., Soneson, C., Hapfelmeier, A., Gardner, P.P., Boulesteix, A.L., Saeys, Y., Robinson, M.D., 2019. Essential guidelines for computational method benchmarking. Genome biology 20, 1–12.

Weng, Y., Wang, X., Hua, J., Wang, H., Kang, M., Wang, F.Y., 2019. Forecasting horticultural products price using arima model and neural network based on a large-scale data set collected by web crawler. IEEE Transactions on Computational Social Systems 6, 547–553.

Wenzel, H., Smit, D., Sardesai, S., 2019. A literature review on machine learning in supply chain management, in: Artificial Intelligence and Digital Transformation in Supply Chain Management: Innovative Approaches for Supply Chains. Proceedings of the Hamburg International Conference of Logistics (HICL), Vol. 27, Berlin: epubli GmbH. pp. 413–441.

Wright, M.N., Ziegler, A., 2017. ranger: A fast implementation of random forests for high dimensional data in C++ and R. Journal of Statistical Software 77, 1–17. doi:`10.18637/jss.v077.i01`.

Wu, C.H., Ho, J.M., Lee, D.T., 2004. Travel-time prediction with support vector regression. IEEE transactions on intelligent transportation systems 5, 276–281.

Wu, H., Levinson, D., 2021. The ensemble approach to forecasting: a review and synthesis. Transportation Research Part C: Emerging Technologies 132, 103357.

Wu, Z., Ramsundar, B., Feinberg, E.N., Gomes, J., Geniesse, C., Pappu, A.S., Leswing, K., Pande, V., 2018. Moleculenet: a benchmark for molecular machine learning. Chemical science 9, 513–530.

Zhang, G., Patuwo, B., Hu, M.Y., 2001a. A simulation study of artificial neural networks for nonlinear time-series forecasting. Computers & Operations Research 28, 381–396. URL: `https://www.sciencedirect.com/science/article/pii/S0305054899001239`, doi:`https://doi.org/10.1016/S0305-0548(99)00123-9`.

Zhang, G.P., 2003. Time series forecasting using a hybrid arima and neural network model. Neurocomputing 50, 159–175.

Zhang, G.P., Patuwo, B.E., Hu, M.Y., 2001b. A simulation study of artificial neural networks for nonlinear time-series forecasting. Computers & Operations Research 28, 381–396.

Zhang, L., Bian, W., Qu, W., Tuo, L., Wang, Y., 2021. Time series forecast of sales volume based on xgboost, in: Journal of Physics: Conference Series, IOP Publishing. p. 012067.

Appendix A. Additional Simulation Results

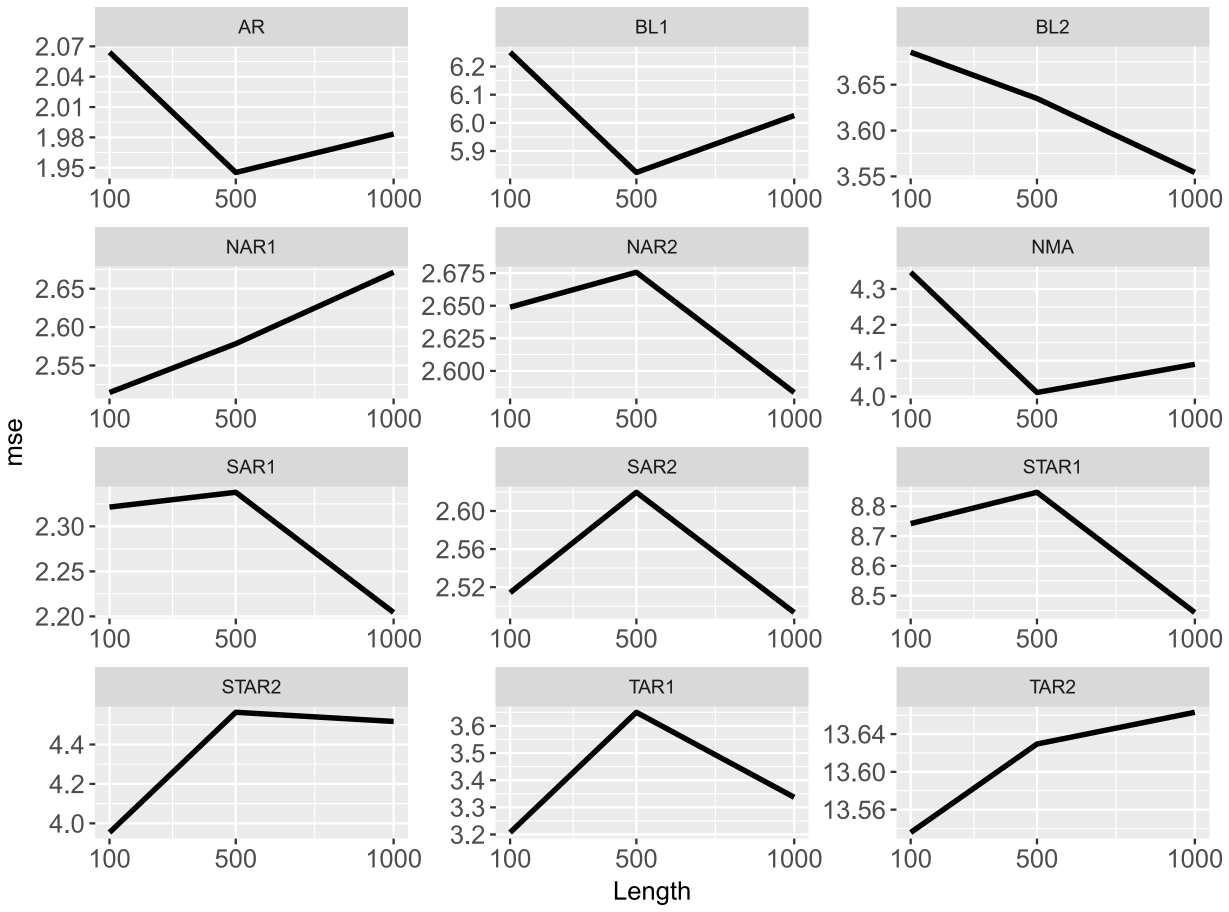


Figure A.10: MSE of the naive approach for the different data generating processes.

Appendix A.1. Influence of Jump Process

Figure A.11 and A.12 summarize the prediction results for all sliding window sizes and data generating processes using the ML methods. For both methods applied to differenced data, the performance is quite similar across the different windows sizes. However, a small difference in MSE values can be observed for the Random Forests, where a smaller window size slightly improves the prediction power.

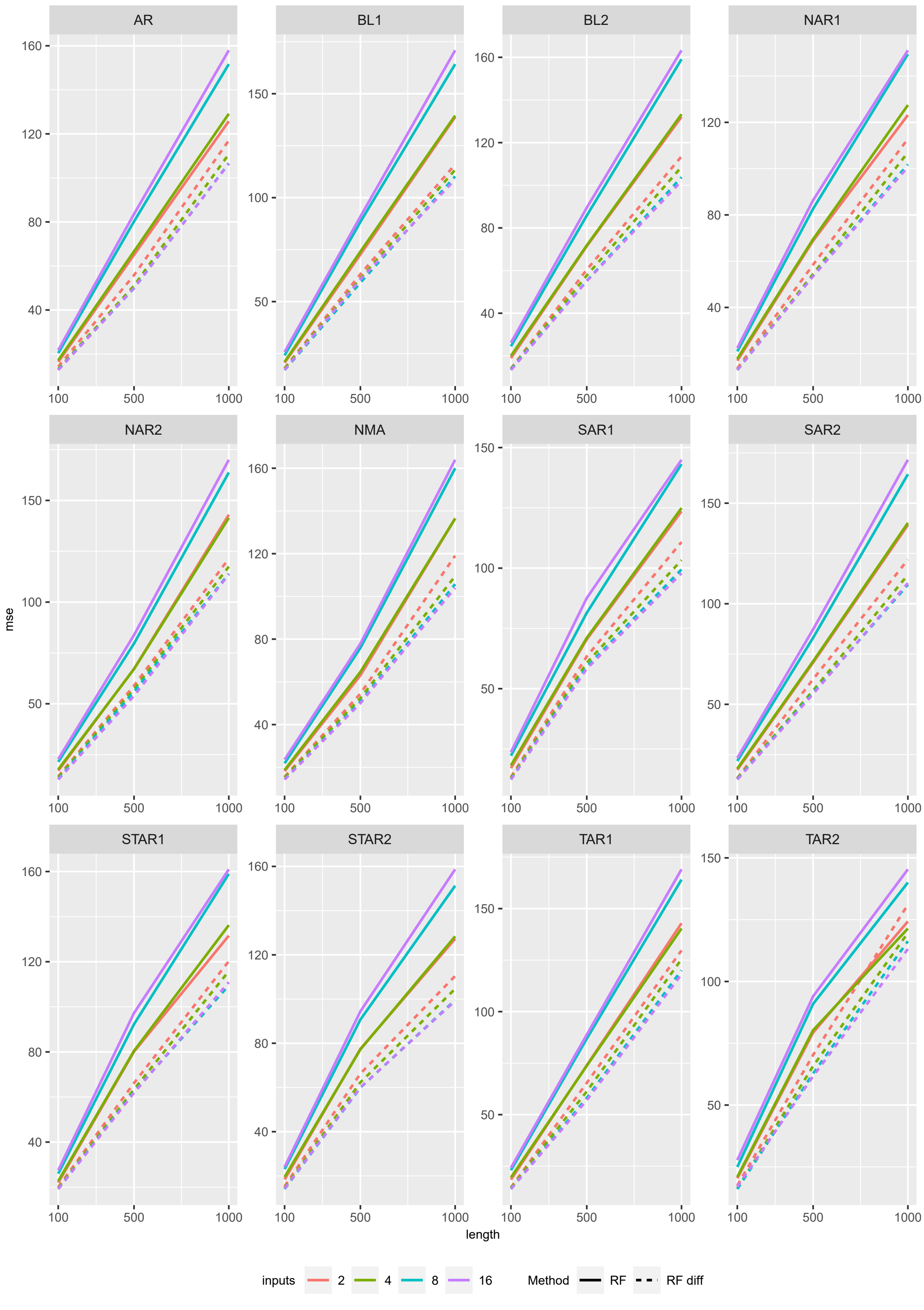


Figure A.11: MSE values of all Random Forest approaches, sliding window sizes and data generating processes superposed by a compound Poisson process.

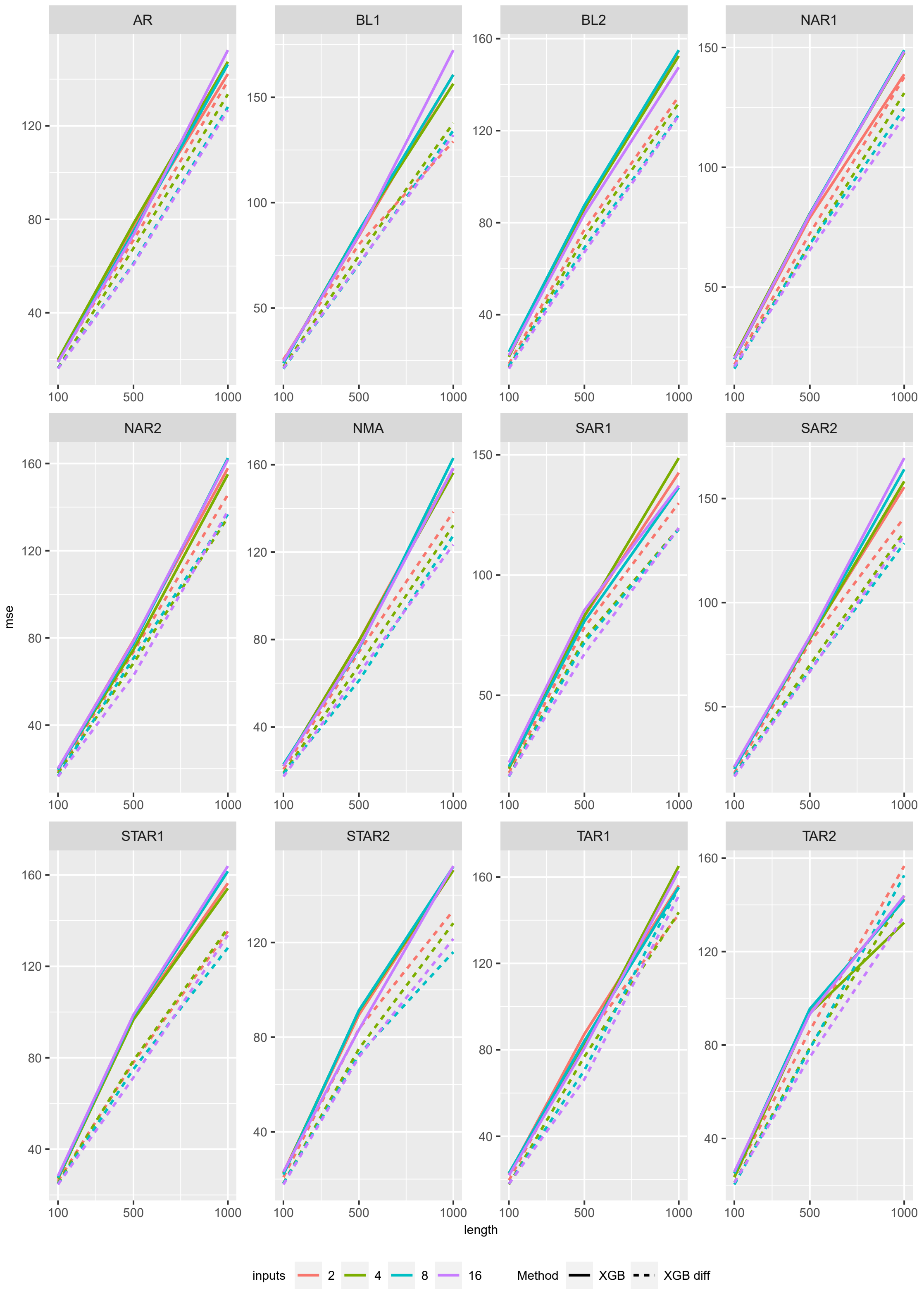


Figure A.12: MSE values of all Random Forest approaches, sliding window sizes and data generating processes superposed by a compound Poisson process.

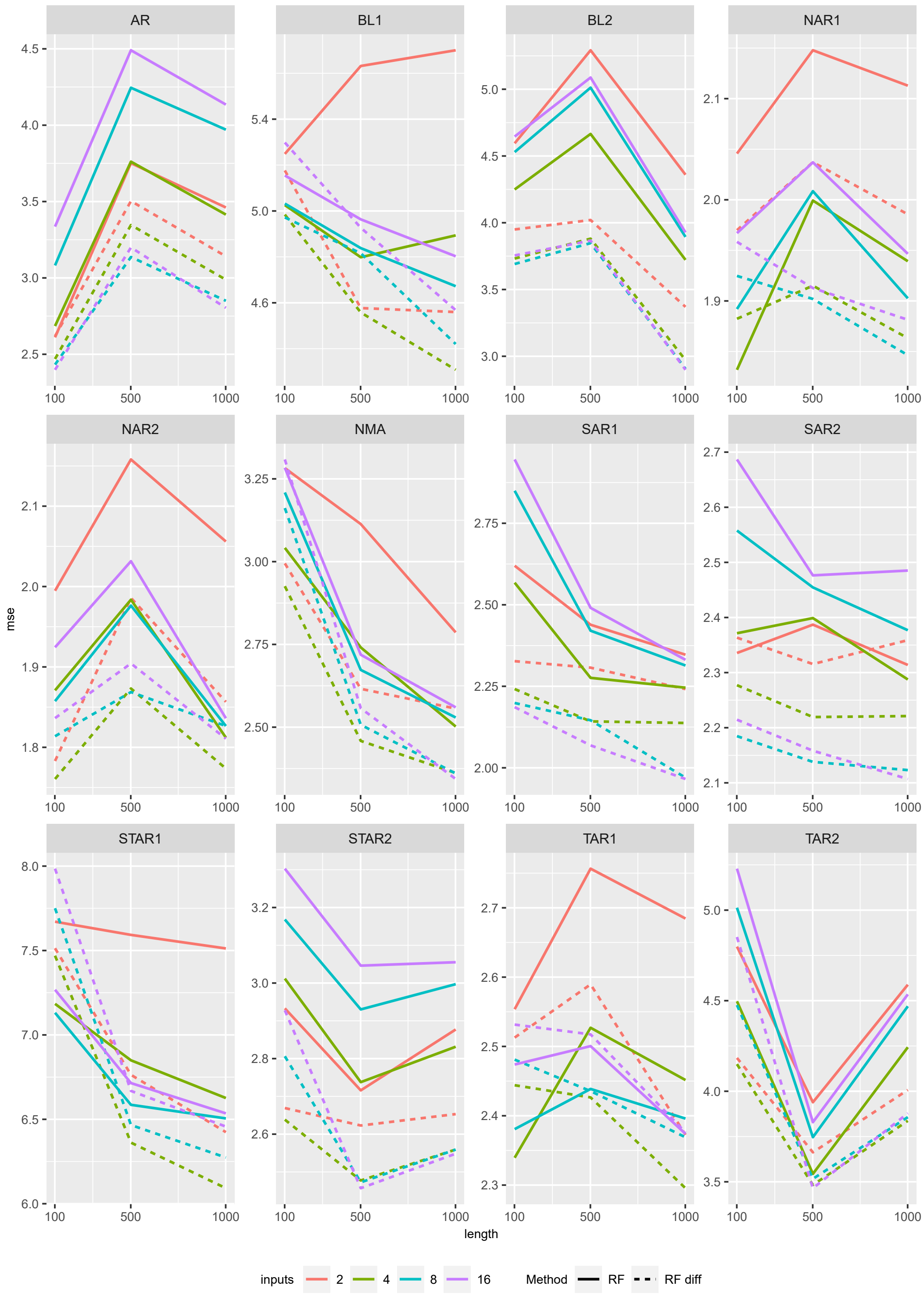


Figure A.13: MSE values of all Random Forest approaches, sliding window sizes and data generating processes superposed by a random walk.

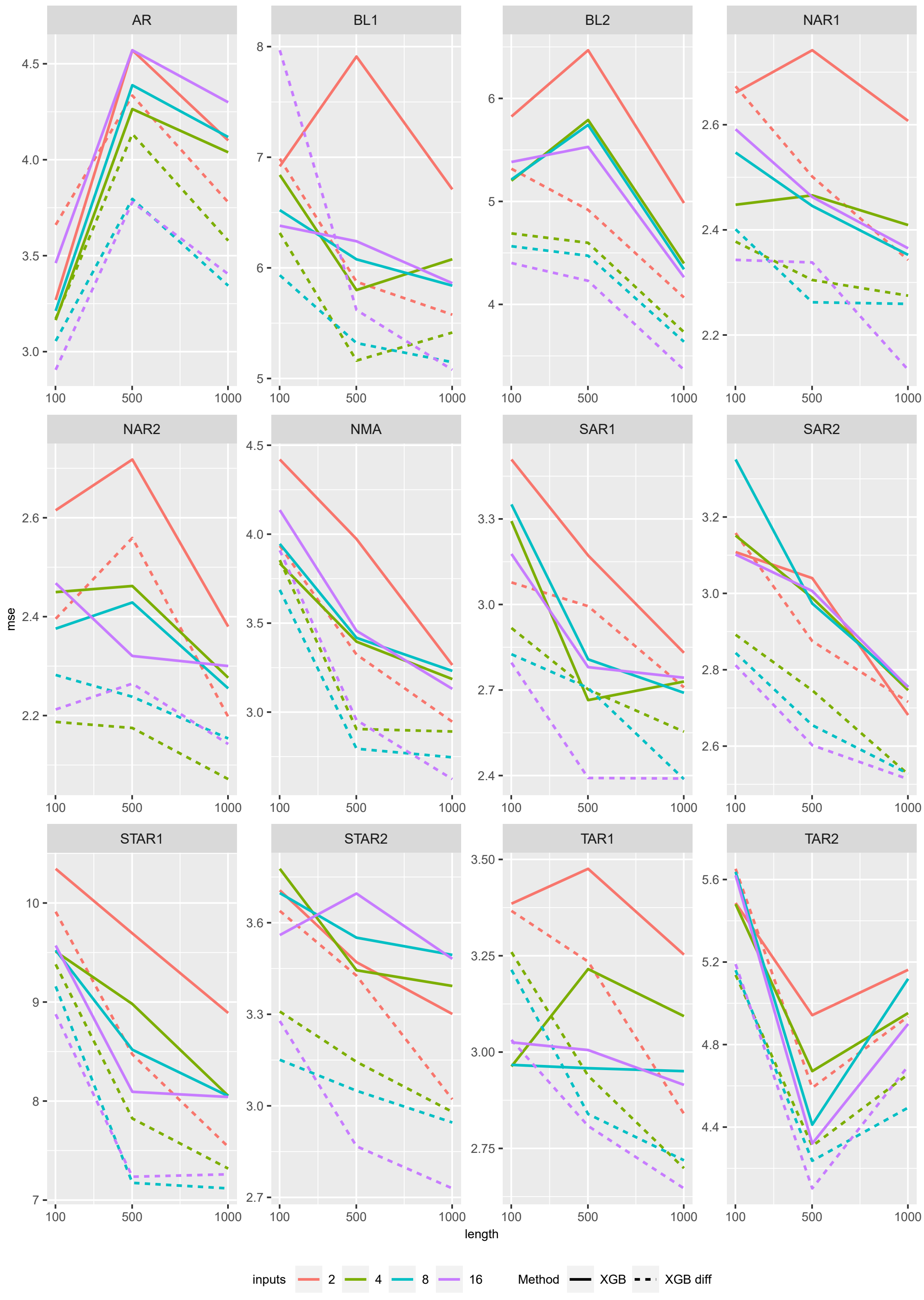


Figure A.14: MSE values of all XGBoost approaches, sliding window sizes and data generating processes superposed by a random walk.

Appendix A.3. Influence of Additional Noise and Jump Process

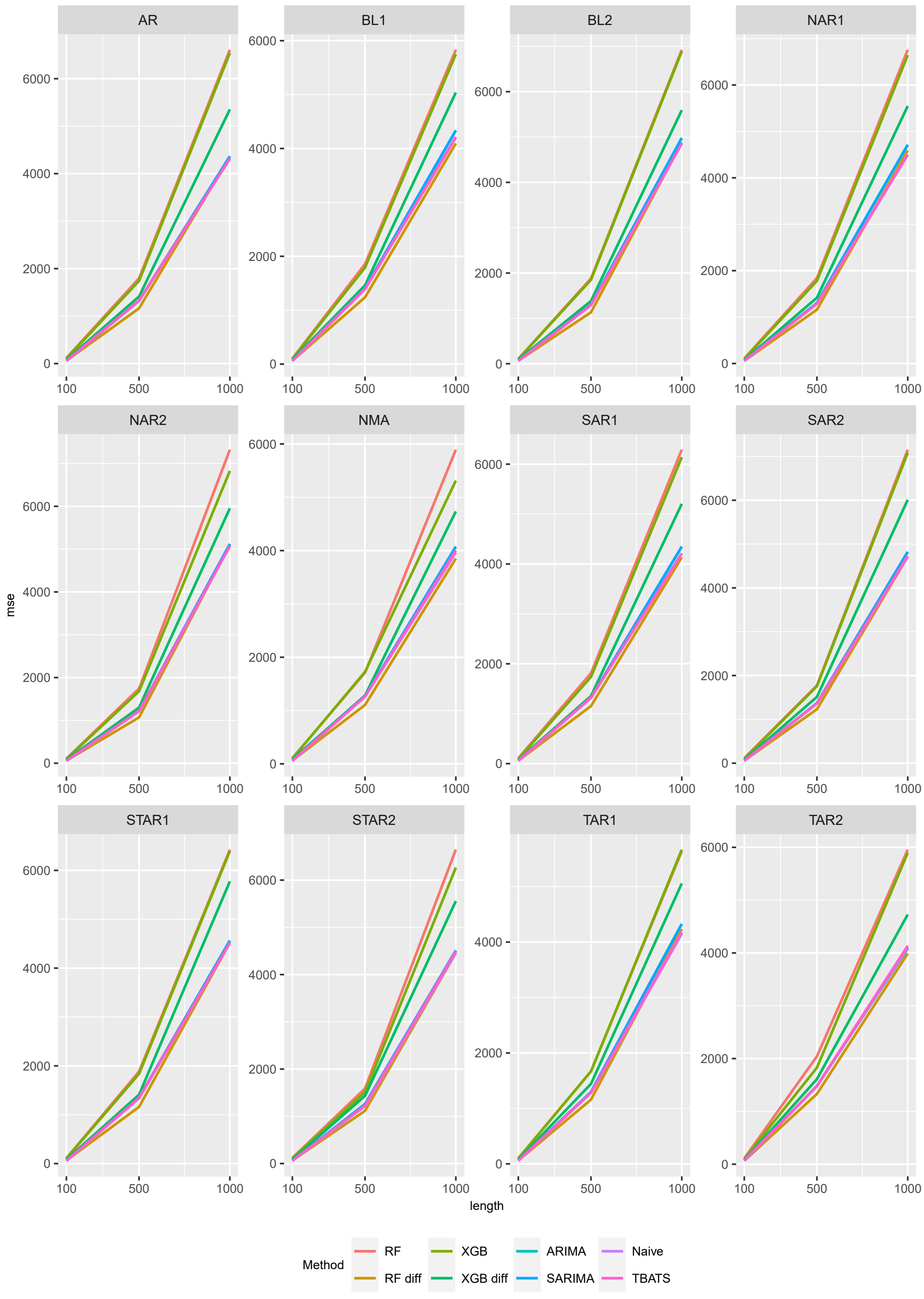


Figure A.15: MSE of all methods and settings, where the data generating processes were superposed by a random walk and compound Poisson process.

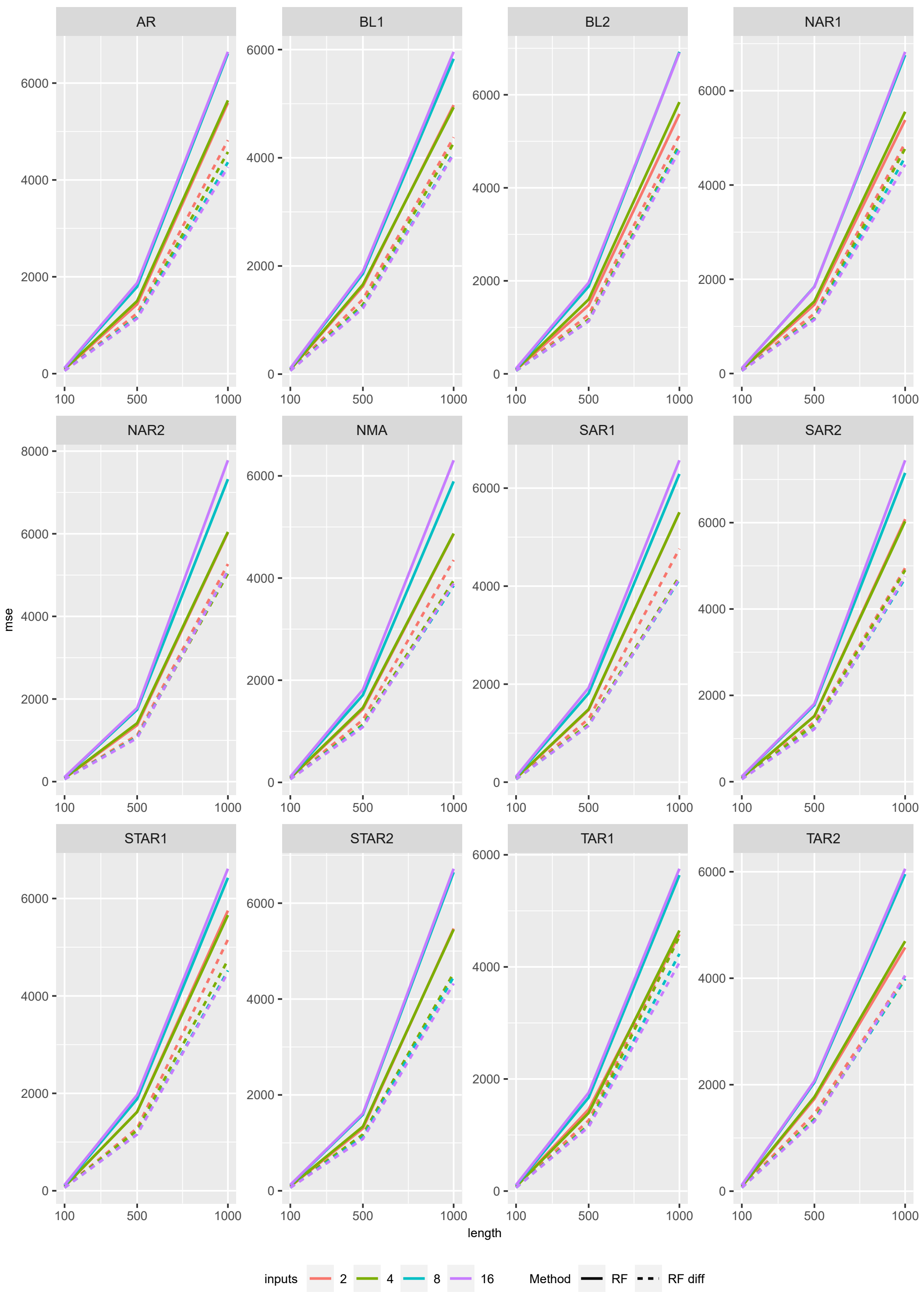


Figure A.16: MSE of all Random Forest approaches, sliding window sizes and settings, where the data generating processes were superposed by a random walk and compound Poisson process.

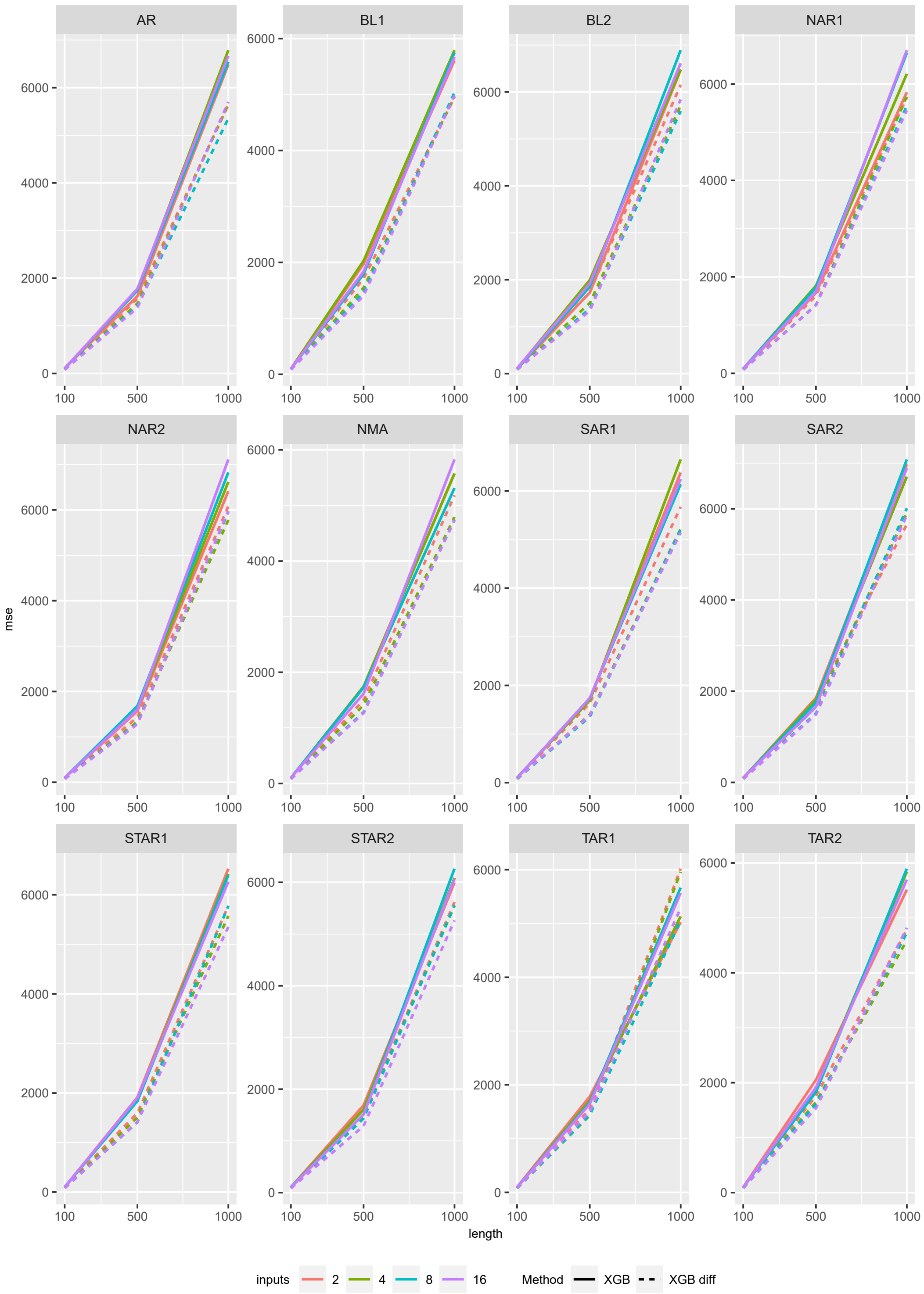


Figure A.17: MSE of all XGBoost approaches, sliding window sizes and settings, where the data generating processes were superposed by a random walk and compound Poisson process.

Appendix B. MAPE Results

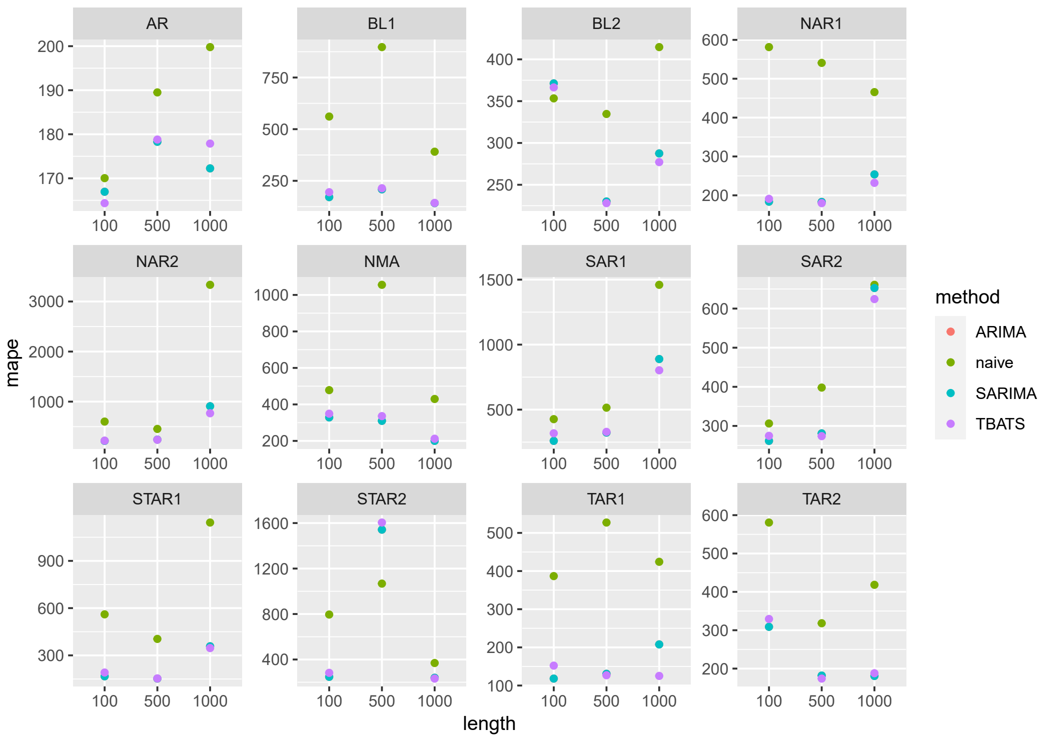


Figure B.18: MAPE of the time series approaches for the different data generating processes described in Table 1.

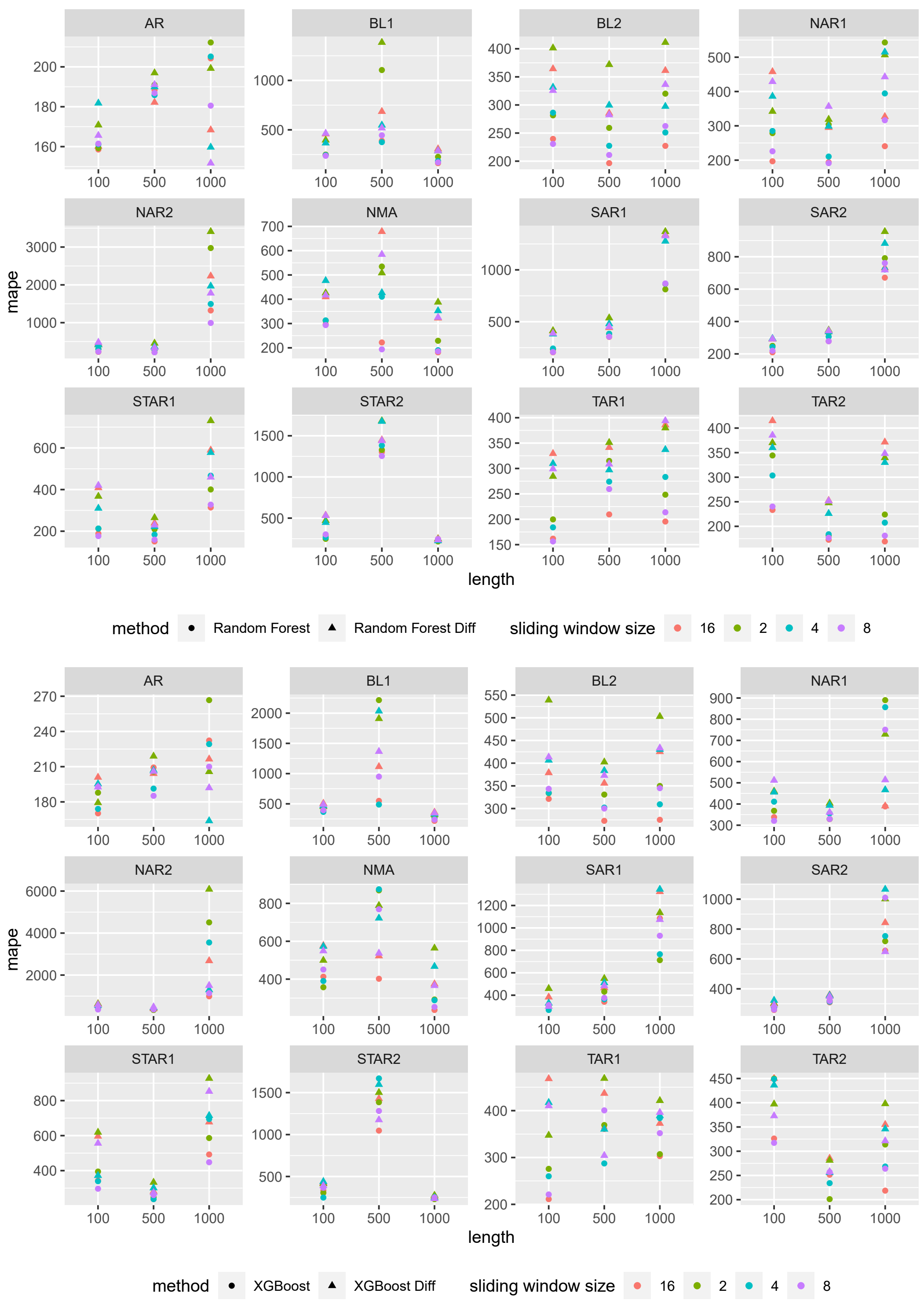
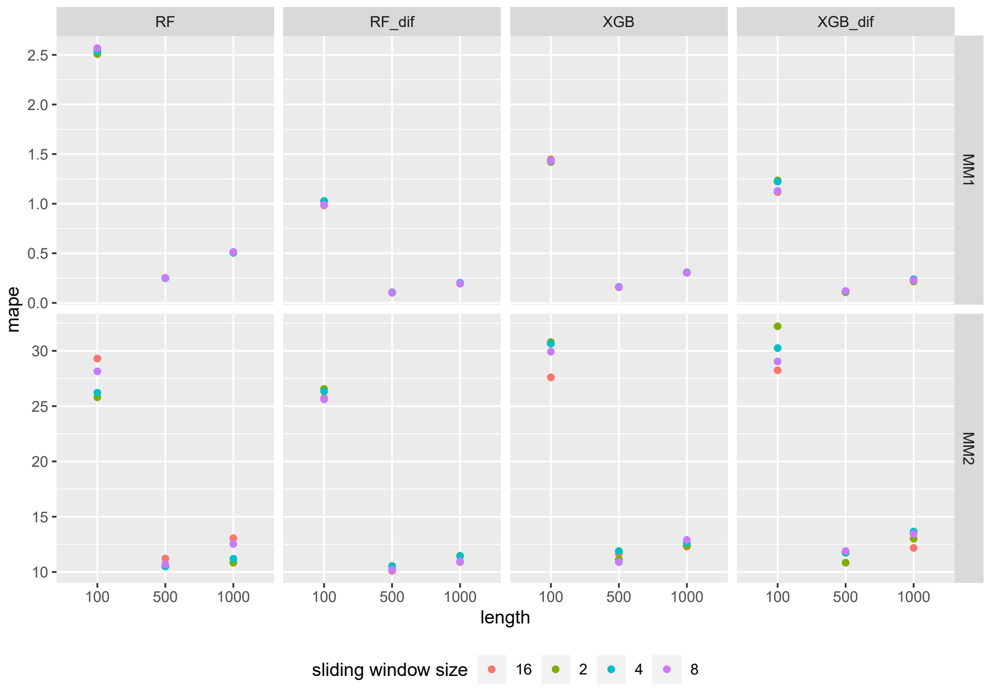


Figure B.19: MAPE of the Random Forest (above) and XGBoost (below) approaches for the different data generating processes described in Table 1 superposed by a compound Poisson process.



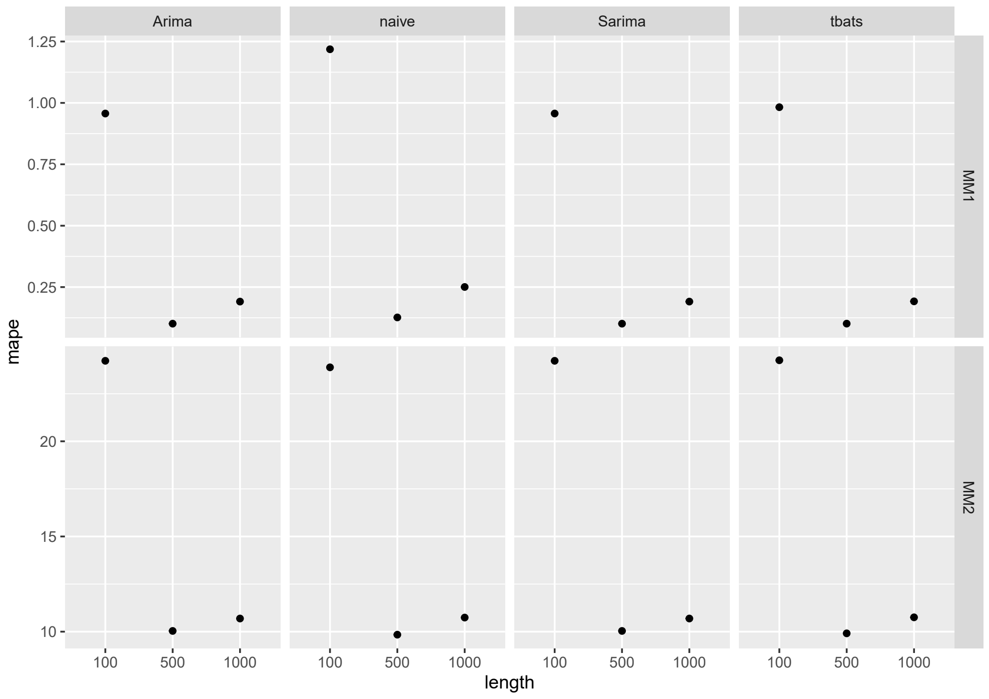
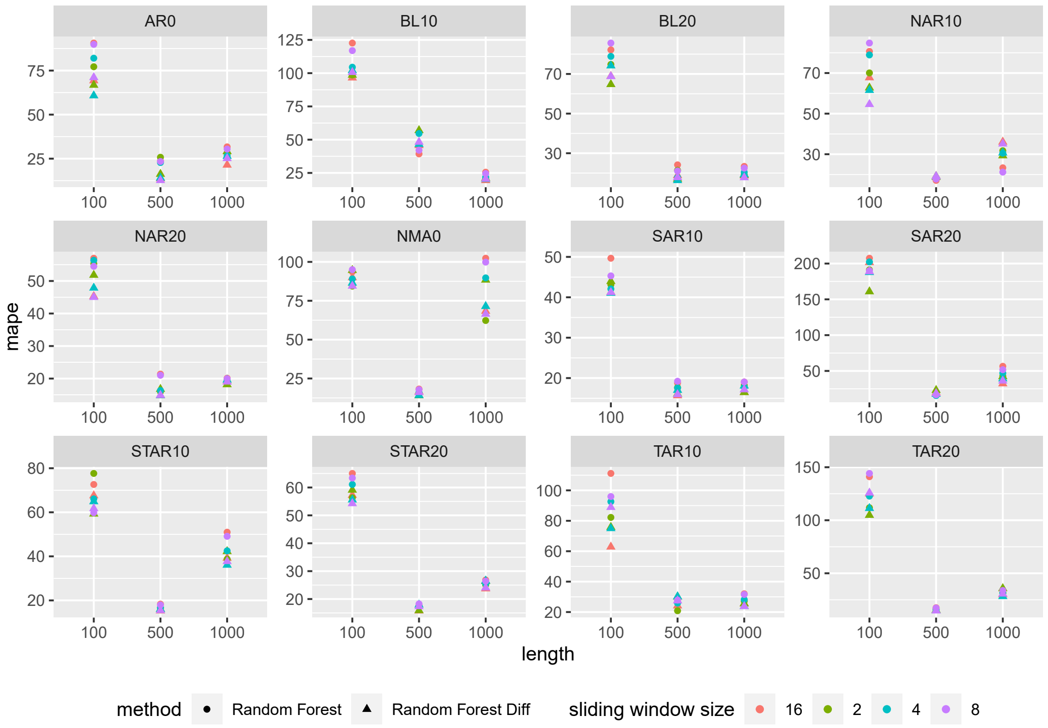


Figure B.20: MAPE of the machine learning algorithms (above) and time series approaches (below) for the M/M/1 and M/M/2 data generating process.



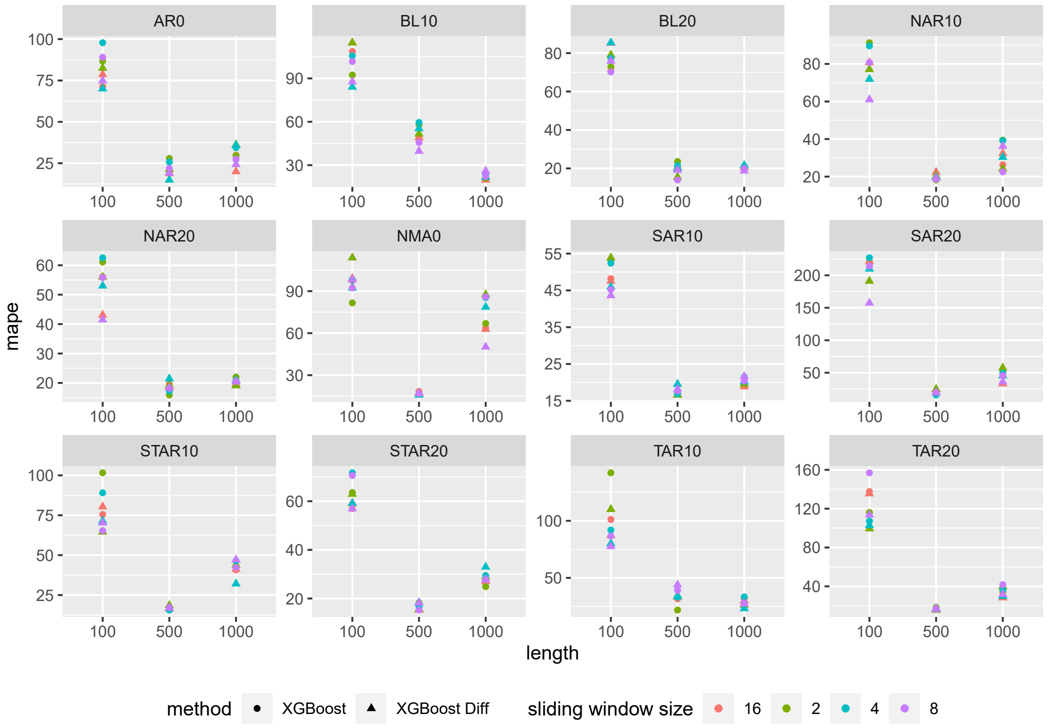
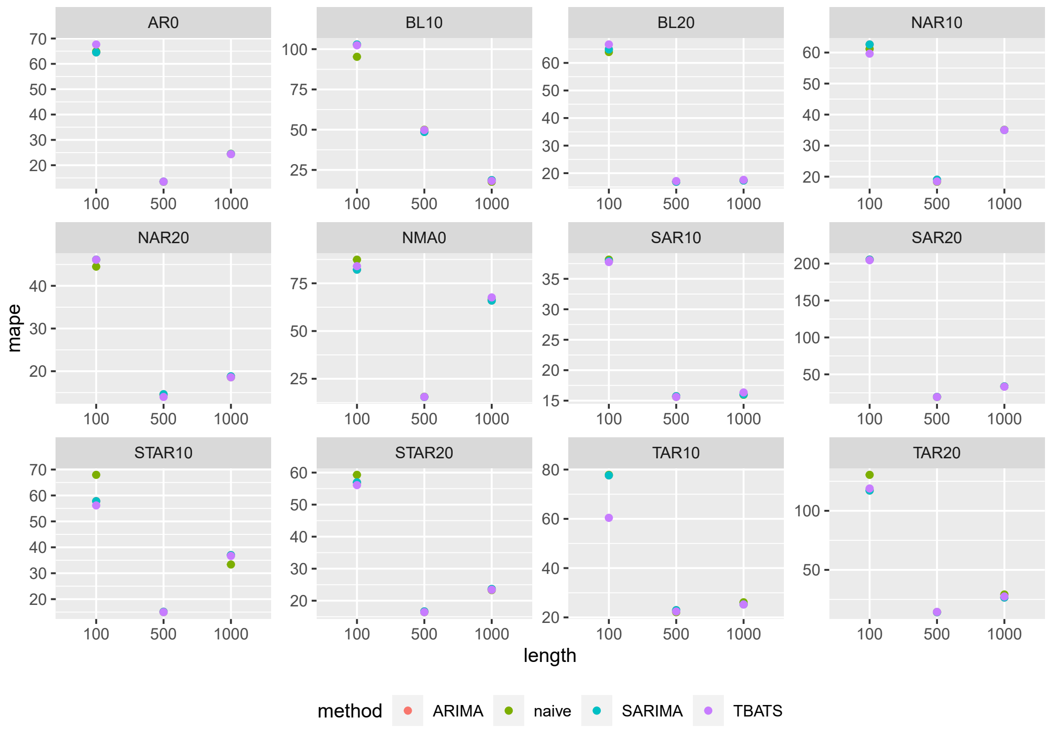


Figure B.21: MAPE of the Random Forest (above) and XGBoost (below) approaches for the different data generating processes described in Table 1 superposed by a compound Poisson process.



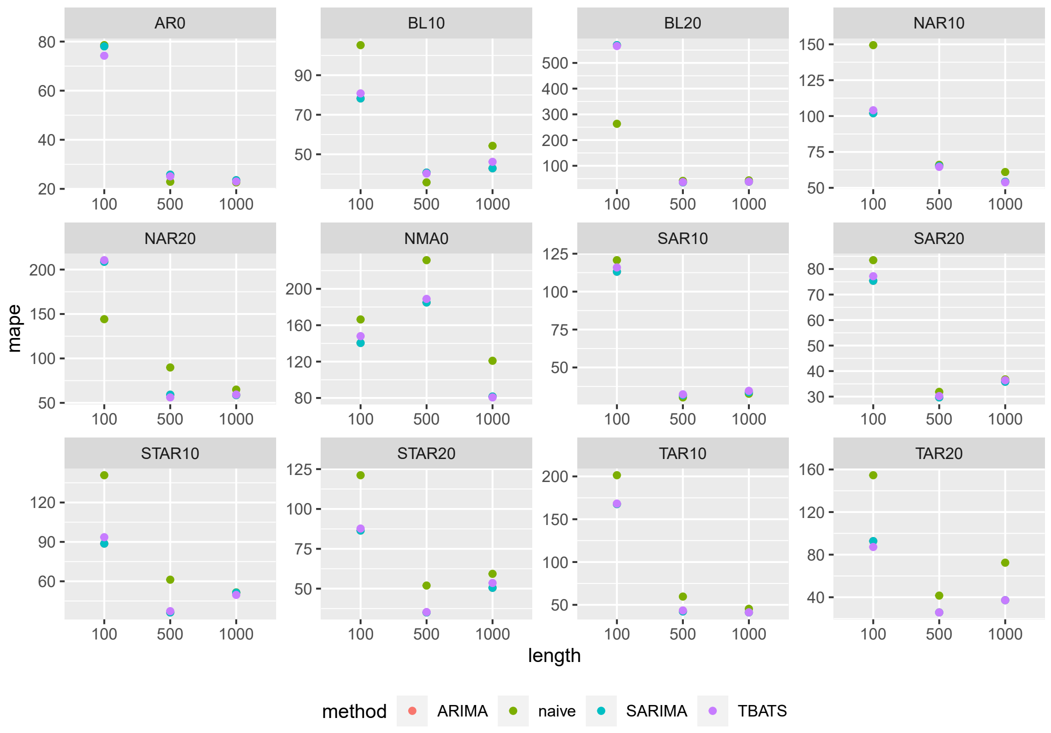
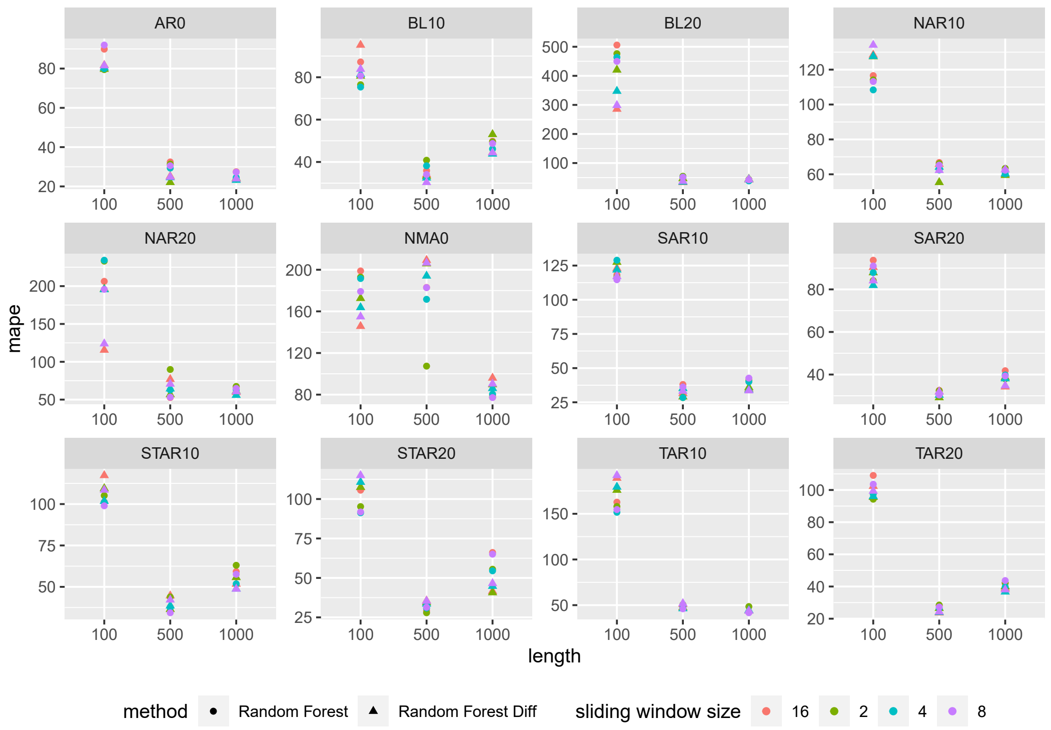


Figure B.22: MAPE of the time series approaches for the different data generating processes described in Table 1 superposed by a compound Poisson process (above) or a random walk (below).



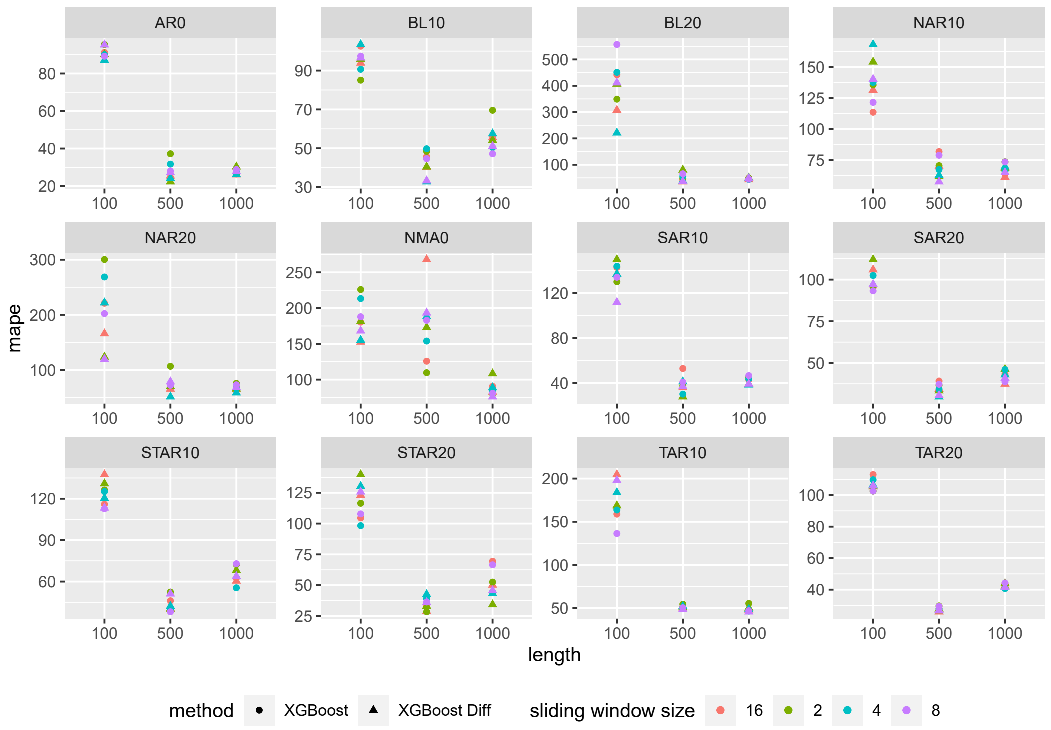
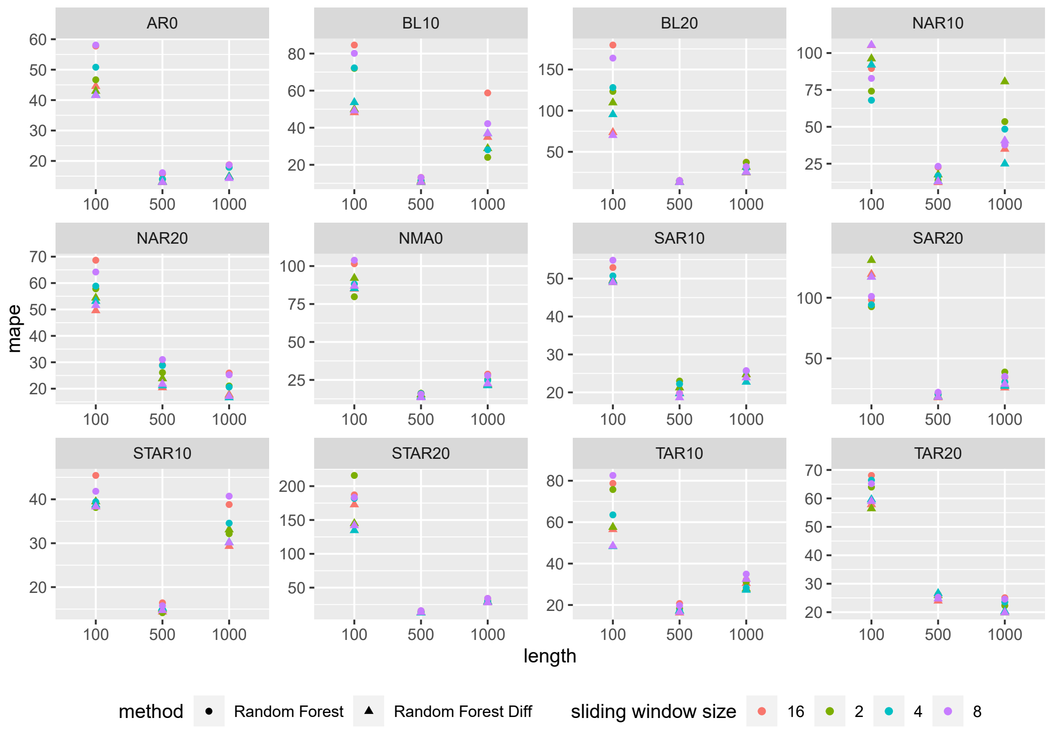


Figure B.23: MAPE of the Random Forest (above) and XGBoost (below) approaches for the different data generating processes described in Table 1 superposed by a random walk.



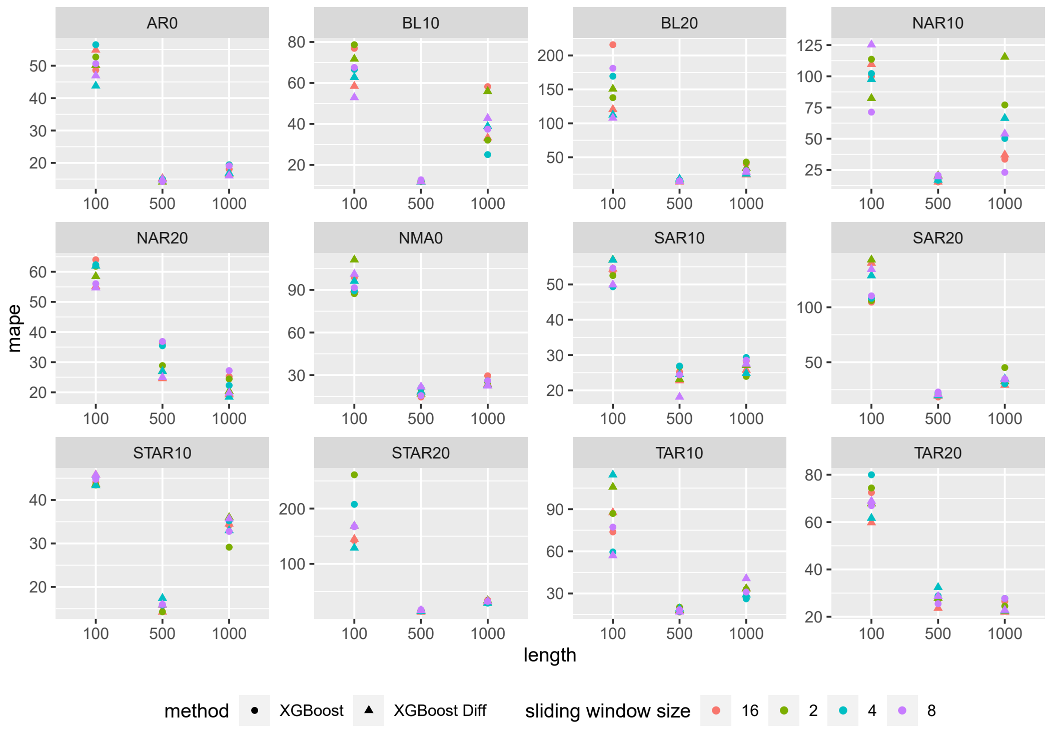


Figure B.24: MAPE of the Random Forest (above) and XGBoost (below) approaches for the different data generating processes described in Table 1 superposed by a compound Poisson process and a random walk.

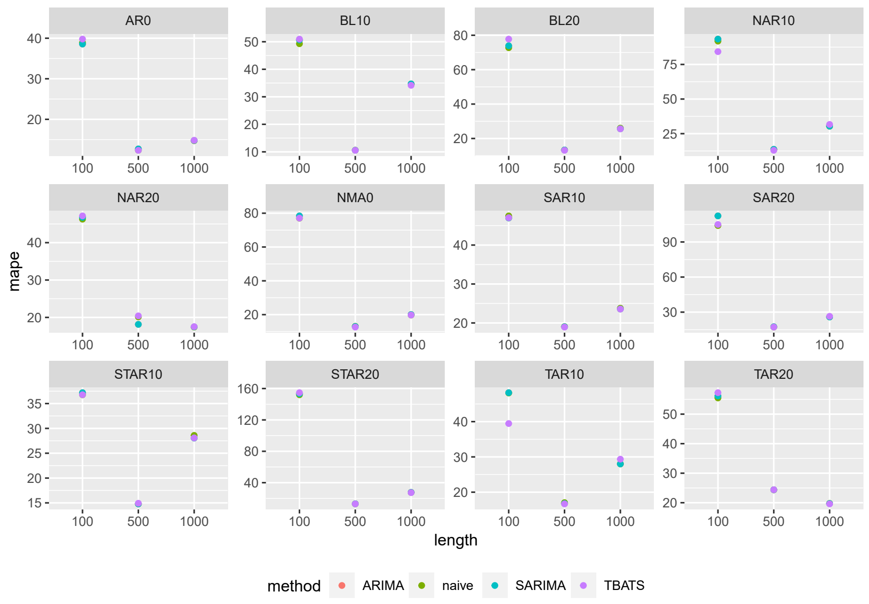


Figure B.25: MAPE of the time series approaches for the different data generating processes described in Table 1 superposed by a compound Poisson process and a random walk.

Article 3

Schmid, L., Biebert, D., Hakert, C., Chen, K.-H., Lang, M., Pauly, M. & Chen, J.-J. (2024). TREE: Tree Regularization for Efficient Execution. *arXiv preprint*, https://doi.org/10.48550/arXiv.2406.12531

TREE: TREE REGULARIZATION FOR EFFICIENT EXECUTION

Lena Schmid [1] **Daniel Biebert** [1] **Christian Hakert** [1] **Kuan-Hsun Chen** [2] **Michel Lang** [1] **Markus Pauly** [1] **Jian-Jia Chen** [1]

ABSTRACT

The rise of machine learning methods on heavily resource constrained devices requires not only the choice of a suitable model architecture for the target platform, but also the optimization of the chosen model with regard to execution time consumption for inference in order to optimally utilize the available resources. Random forests and decision trees are shown to be a suitable model for such a scenario, since they are not only heavily tunable towards the total model size, but also offer a high potential for optimizing their executions according to the underlying memory architecture.

In addition to the straightforward strategy of enforcing shorter paths through decision trees and hence reducing the execution time for inference, hardware-aware implementations can optimize the execution time in an orthogonal manner. One particular hardware-aware optimization is to layout the memory of decision trees in such a way, that higher probably paths are less likely to be evicted from system caches. This works particularly well when splits within tree nodes are uneven and have a high probability to visit one of the child nodes.

In this paper, we present a method to reduce path lengths by *rewarding* uneven probability distributions during the training of decision trees at the cost of a minimal accuracy degradation. Specifically, we regularize the impurity computation of the CART algorithm in order to favor not only low impurity, but also highly asymmetric distributions for the evaluation of split criteria and hence offer a high optimization potential for a memory architecture-aware implementation. We show that especially for binary classification data sets and data sets with many samples, this form of regularization can lead to an reduction of up to $\approx 4\times$ in the execution time with a minimal accuracy degradation.

1 INTRODUCTION

Execution time optimization of machine learning models on the edge on extremely resource constrained devices has been widely studied, especially known as TinyML scenarios. While one popular approach is to shrink the models (e.g., by reducing the number of neurons in neural networks, or the depth of decision trees in random forests) without losing much accuracy, this approach is agnostic to the actual properties of the underlying hardware. One aspect of resource limitation is often the limited availability of energy and hence time budget for the execution of inference. Shrinking models indeed can meet this requirement, but a considerable reduction of the execution time can also be achieved by an orthogonal hardware-aware implementation of the model, especially in the context of random forests (Chen et al., 2022; Tabanelli et al., 2022).

Random forests and their inner structure of decision trees are a premier candidate for cache-aware optimizations, since every inference follows one path, requiring only a small subset of nodes from the tree. This naturally fits in the design principle of caches, since these are usually small and depend on a high locality of the memory accesses to be fast and efficient. Chen et al. leverage a probabilistic model, describing the distribution of splits, to place the frequent accessed paths in a cache-friendly manner (Chen et al., 2022). Breaking the probabilistic model down to a single tree node, we observe that the approach can be beneficial only when the probability of data tuples (i.e., split) to take the left branch or the right branch differs significantly. In consequence, when optimizing the execution time of random forest models, a reduction in the tree size should account for the distribution of splits in the tree nodes and maintain or enforce the property of uneven splits. This can lead to a considerable execution time improvement by the reduced tree size and an orthogonal improvement to favor such cache-friendly implementations.

In this paper, we introduce the design of a hardware-aware regularization for decision tree training by actively *rewarding* uneven splits in single decision tree nodes. This leads to

[1]TU Dortmund University, Germany [2]University of Twente, Netherlands. Correspondence to: Lena Schmid <lena.schmid@tu-dortmund.de>.

the regularized construction of decision trees, which maintain the crucial properties for cache optimization, but with reduced total size or depth of some paths. Consequently, the *studied problem* of this paper is how to regularize random forest training with the objective of reducing the model size and to reward uneven splits, while not degrading the accuracy significantly. We *tackle the problem* by introducing a regularization term into the split method of decision trees. This regularizer rewards split decisions that lead to uneven splits to uphold asymmetric distributions. This leads to an orthogonal speed improvement to the cache optimization mentioned above, and can even assist cache optimization.

The introduction of the regularization offers a trade-off for the application. The regularization term can be controlled by a factor to take an either minor or major influence. We propose an intuitive application, where a tolerable degradation in accuracy can be defined by the user. Subsequently, possible degrees of regularization are automatically tested, and the configuration with the maximal improvement in execution time within the tolerable accuracy degradation is chosen. If users are eager for a deeper investigation and the manual choice of a trade-off between accuracy degradation and execution time improvement, we report the corresponding data for a comprehensive set of possible and meaningful regularization degrees. These results are graphically illustrated and allow an easy choice of the trade-off. It is generally a good idea to focus on Pareto optimal points with respect to accuracy degradation and execution time improvement in this data set for a first investigation. Beyond the choice of the meaningful application of the degree of regularization, the level of maximal possible meaningful regularization can reveal the information of how well the data set, which is used for training is suited for this form of regularization. With the help of this, we determine a property, which we call *regularization robust* on data sets, and identify properties, which make data sets more regularization robust.

Despite the realization of the regularization in scikit-learn (Pedregosa et al., 2011), we focus on a comprehensive experimental evaluation of the proposed regularizer in this paper. In detail, we take a set of UCI datasets (Dua & Graff, 2017) and investigate the regularization in different model configurations. Furthermore, we conduct an extensive simulation study with synthetic datasets, where the relation between dataset properties and the effectiveness of the regularization is analyzed. In short, we provide the following contributions:

- A regularization term for the provocation of uneven splits in decision tree training, including an implementation in scikit-learn.
- Evaluation of the regularization on UCI datasets.
- An extensive simulation study with synthetic datasets to reveal the relation between dataset properties and regularization effectiveness.

2 Related Work

Performance optimization of trees and random forests is a widely studied topic in the literature. When it comes to concrete hardware-close implementations, one popular example is the C++ implementation for random forests in Wright & Ziegler (2017). The prominent concept of native trees, where nodes are stored in an array and executed in a narrow loop and if-else trees, where nodes form deeply nested if-else constructs, is introduced to maintain locality in the data and instruction memory in Asadi et al. (2014). More variances of tree implementations are studied for the runtime of inference on RISC-V MCUs (Tabanelli et al., 2022).[1]

Random forests are also considered to be executed on accelerator devices, such as GPUs or FPGAs (Van Essen et al., 2012; Nakandala et al., 2020; Buschjäger & Morik, 2018) or in a vectorized manner (Kim et al., 2010). In addition to the deployment of the models to a hardware-close language and massive parallel computation devices, also the optimization of the usage of the underlying hardware is investigated. This includes optimization of the throughput in a pipeline execution (Prenger et al., 2013) and investigating the data structure and the decision tree structure itself and gain performance improvement with proper reordering (Dato et al., 2016; Lucchese et al., 2016). More specifically, the usage of floating point hardware units and their performance impact is studied (Hakert et al., 2022a;b). Chen et al. (2022) utilize a probabilistic model of the data distribution in the data set to optimize the memory layout, in order to favor frequently used paths for the cache behavior.

Although the approaches above provide various optimized implementations of random forests, they do not alter the training process in order to gain execution time performance. One relevant approach is hyperparameter tuning (Bischl et al., 2023). Hyperparameter tuning specifically for random forests is covered in Probst et al. (2019), resulting in the tool `tuneRanger` focusing on both accuracy maximization and explainability. The tool, however, does not include execution time performance as an objective. Mondrian forests (Lakshminarayanan et al., 2014) in contrast, introduce an online adaptive realization of random forests, which can improve the execution time performance while maintaining a similar accuracy.

Regularizing the training process of random forests, to the

[1]Its naming system is deviated to the terminology used by most of related works. For example, the if-else trees are named Naive kernels, and the native trees are named Loop kernels. In this work, we follow the majority and use the terminology commonly found in the literature.

best of our knowledge, has not been studied for the objective of execution time performance. Regularization, also beyond the scope of random forests, however, is a studied topic in order to provide more explainability (Wu et al., 2018) or achieve higher accuracy (Scheffer, 2000). Also, the effect of high randomness in the random forest training as a form of regularization is investigated (Mentch & Zhou, 2020). The objective of error tolerance and robustness is further shown to be addressable by regularizing the training of binarized neural networks (BNNs) (Buschjäger et al., 2021).

3 Tree Regularization

Improving the execution time of decision tree inference on real hardware opens a larger design space. One way to achieve faster inference is to decrease the size of the model itself. The obvious benefit towards execution time is, that less computation is needed to return an inference result. This approach usually introduces degradations in accuracy, as such the model cannot be shrank to an arbitrarily small size. A widely used method to decrease the model size is limiting the maximal depth a tree is allowed to grow to.

In this work, we introduce an alternative method towards reducing the overall model size. We optimize the decision tree construction to increase the existence of uneven splits to benefit shorter paths to leaf nodes. More precisely, a penalty term in the splitting criterion is introduced, which serves as a control parameter to trade-off between tree size and predictive accuracy. This control parameter effectively shrinks the model size and reduces depths of single paths by maintaining and provoking uneven split decisions.

Another effective method is utilizing the cache behavior of the CPU. Chen et al. have shown that reordering the nodes inside memory in a cache-friendly manner improves execution time (Chen et al., 2022). In their approach the split probabilities of nodes are used to determine the new order in memory. Here uneven splits are beneficial, as they result in nodes which are accessed more often. Therefore, the benefit of the cache-friendly ordering is increased. Our proposed regularization both optimizes for smaller model sizes and increases the likelihood of uneven splits.

For the sake of completeness, we first give a short overview of the decision tree construction with the CART algorithm. Afterwards, we present the introduced regularization and how it can be tuned for different scenarios iteratively. Lastly, we discuss why the persistence of uneven splits are orthogonal to the cache-aware optimizations in detail.

3.1 Decision Tree Construction

A widely used training method to construct decision trees is the CART algorithm (Breiman et al., 1984), by which the samples are repeatedly split by a chosen criterion. The result will be two sets of samples from the dataset. This is recursively repeated until a given stopping criterion is met (e.g. a certain depth is reached).

The basic working principle of all split criteria is to compute a score for all possible split values at each node, and then select the split point corresponding to the best combined criterion scores in the two resulting child nodes. More precisely, for a classification problem with k labels, p_i denotes the proportion of samples with class c_i $(i = 1, \dots, k)$ in a node. A widely used score for the impurity is the Gini impurity, which is measured as

$$\mathrm{GINI} = 1 - \sum_{i=1}^{k} p_i^2. \qquad (1)$$

Hence, when all samples belong to one class, the sum is 1 and the resulting impurity is 0. The Gini impurity results in a larger value, the more evenly the class labels are distributed in the node. One popular way to find the best split inside a node is finding the minimal mean Gini impurity of both resulting child nodes. We note that other split criteria such as Entropy and Information Gain can also be used (Breiman et al., 1984). However, as the exact criterion for splitting is not relevant for our proposed regularization, only the Gini impurity is covered here.

The split results in the samples being separated into two portions, being further used in the left and right child. This division in the samples then determines the probability of the left or right subtree to be used in an inference, Each node has a distinct access path starting from the root node and ending in the node itself. To get the absolute probability of any node, the individual probabilities of every node on the path to that node need to be multiplied. The resulting value is the probability of this node to be accessed during prediction. Intuitively, the absolute probability of the root node is 100%. The probability of any path to be taken during inference is the probability of the leaf node the path ends in. These absolute probabilities can be used to identify which paths are frequently accessed.

3.2 Regularization Factor

A possibility to improve the execution time is to reduce the total model size by controlling the training process to only keep important paths. The reduced amount of nodes leads to less memory loads during an inference. In addition, this regularization of the training can be designed such that not only important paths in terms of prediction accuracy are kept, but also the access frequency of paths is maintained kept. This consequently leads to an orthogonal optimization of the cache optimization from Chen et al., since the cache friendly handling of frequently accessed paths is kept, and cache replacements are reduced.

Since training of decision trees according to the CART algorithm (Breiman et al., 1984) consists of recursively splitting the samples into two child nodes based on a threshold value, the split decision can be modified in order to favor asymmetric probabilities. In order to allow a trade-off between the original split criterion and the size-aware split, we introduce an additive regularization factor for the split criterion, penalizing even splits. The amount of penalization can be controlled with a real-valued factor λ which is subject to tuning. Although this design is applicable to arbitrary split criteria, we here restrict ourselves to the popular Gini impurity criterion in order to analyse the effect in depth.

In order to include a size-aware splitting criterion into this process, we define a regularization term R as

$$R = 1 - \frac{|\#\text{samples}_{\text{left}} - \#\text{samples}_{\text{right}}|}{\#\text{samples}}. \qquad (2)$$

Hence, when the split distributes samples almost equally to the left and right child nodes, the value is close to 1, when the split is very asymmetric on the other hand, the value is closer to 0. Note that in contrast to the Gini impurity, the regularization term does not operate on the class labels, but instead on the number of samples. In order to form the resulting split criterion, we add the regularization term with an adjustable weight $\lambda \in \mathbb{R}^+$ to the Gini impurity:

$$\text{GINI}' = \text{GINI} + \lambda \cdot R. \qquad (3)$$

Adding the regularization term to the evaluation and optimization of the Gini impurity in every step of the CART algorithm allows accounting for cache-friendly splits during the training. It should be noted that the introduction of the regularization potentially degrades the Gini impurity and hence also the accuracy of the trained model. Consequently, the parameter λ has to be chosen effectively to provide a good trade-off between accuracy and asymmetric splits.

Our modifications are directly implemented in scikit-learn. To achieve the outlined regularization, a new split criterion based on the Gini impurity is introduced. The implementation is largely similar as for the standard Gini split criterion. However, when calculating the node impurity, the resulting value is adapted according to Equation (3) and returned. To accommodate the factor λ, an additional hyperparameter can be set while fitting the model to control the amount of regularization. The source code is publicly available under `[hiddenduetodoubleblindsubmission]`.

3.3 λ Tuning

During training, the regularization factor λ needs to be set. It should improve training towards the best performance optimization while preserving the accuracy as good as possible. An optimal regularization factor cannot be picked universally. The effectiveness and influence of the factor changes highly depending on a variety of factors (e.g., the number of classes in the dataset).

There is a limit to how much any split can be usefully regularized, as at some point all samples would go to one child node. Therefore, the impact of the regularization factor is going to approach a limit the larger the factor gets.

To find the optimal factor for a given scenario, the expected performance improvement needs to be quantified. To that end, we define the expected depth of a single tree. It is measured as

$$\sum_{l \in \text{leaf}(t)} p_l * \text{depth}(l) \qquad (4)$$

where leaf(t) are all leaves of tree t, p_l is the probability of leaf node l and depth(l) is the depth of node l. The expected depth is therefore the mean depth the inference is expected to reach during repeated inference operations. Consequently, a reduction in the expected depth results in an increase in performance, as fewer nodes have to be loaded during inference. Furthermore, once the expected depth does not change significantly, the influence of the regularization factor is less pronounced and less performance gain is to be expected. To find an optimal factor, the factor is iteratively increased until the difference in expected depth falls under a set threshold, which decides how close to the best possible performance improvement the factor is tuned. At that point, performance is unlikely to improve further, and the corresponding value for λ is chosen.

4 Experimental Evaluation

To evaluate the application of the hardware-aware regularization, we conducted experiments on real and synthetic data sets. First, we apply a default setting, where the maximal regularization is applied with a configurable, tolerable accuracy degradation. Second, we enlighten the trade-off between degree of regularization, speed improvement and accuracy drop. Lastly, we evaluate the limitations of regularization itself and report the boundaries for the meaningful application.

4.1 Evaluation Setup

For evaluating the execution time improvement, we trained random forests with different degrees of regularization (i.e. varying λ) on real and synthetic datasets. We subsequently generated a straightforward C implementation and a cache optimized implementation via Chen et al. (2022). The generated trees of both implementations are executed on a real world target machine. We use a server class system, i.e. with an Intel(R) Xeon(R) Gold 5218 @ 2.3GHz CPU with 16 cores, 1024 KiB L1 Cache, 16 MiB L2 Cache and 22 MiB L3 Cache and 180GB RAM. We utilized Scikit-learn to train random forests with varying number of trees and maximal tree depths (1, 5, 15, 20) for each of these datasets. After a given threshold, the number of trees was only increased for more experiments if it improves the accuracy enough. This was done to reduce the amount of redundant experiments. To provide a better intuition for the impact of the regularization, we always compare the regularized implementation to the comparable not regularized counterpart. In greater detail, the not cache optimized regularized implementation for a specific number of trees and maximal tree depth is compared to the not regularized version of the not cache optimized implementation. This is similarly done for the cache optimized implementations.

In addition, we measured the balanced accuracy of the trained model based on the test dataset. Balanced accuracy evaluates a model's classification performance by considering both sensitivity (true positive rate) and specificity (true negative rate), making it particularly useful in scenarios with imbalanced datasets. Since the methods from Chen et al. (2022) only optimize the memory layout and do not change the model structure, the balanced accuracy is the same for all implementations. For the measurement of the execution time, we executed 50 repetitions of the inference of the test dataset and average the time consumption under realistic execution conditions. To compare the balanced accuracy and mean relative execution time, we used a training-test split ratio of 3:1 and repeated it 8 times. The scitkit-learn hyperparameter $max\_features$ was varied across a set of recommended default values $\{\frac{\lfloor\sqrt{p}\rfloor}{2}, \lfloor\sqrt{p}\rfloor, 2\lfloor\sqrt{p}\rfloor, p\}$, where p denotes the number of features (Wright & Ziegler, 2017; Hastie et al., 2009; Liaw et al., 2002).

4.2 UCI Datasets

In the following experiments, the influence of the regularization is compared on eleven datasets from the UCI repository, which was also adopted in Chen et al. (2022). Table 1 lists the dataset name, source, number of samples (n), number of features (p), and number of classes (cl). For the ease of

Table 1. Name, source, number of samples (n), number of features (p), number of classes (cl) of each used dataset.

Dataset	Source	n	p	cl
Adult	(Kohavi, 1996)	48,842	64	2
Bank Marketing	(Moro et al., 2014)	45,211	59	2
Covertype	(Blackard & Dean, 1999)	581,012	54	7
Letter	(Frey & Slate, 1991)	20,000	16	26
Magic	(Dua & Graff, 2017)	19,020	10	2
MNIST	(Dua & Graff, 2017)	45,000	784	10
Satlog	(Dua & Graff, 2017)	6,435	36	6
Spambase	(Dua & Graff, 2017)	4,601	57	2
Sensorless Drive	(Dua & Graff, 2017)	58,509	48	11
Wearable Computing	(Ugulino et al., 2012)	165,632	17	5
Wine Quality	(Cortez et al., 2009)	6,497	11	7

presentation, we aggregated the multiple simulation settings with regard to the 8 replications and focused on results obtained with $max\_features$ set to $\lfloor\sqrt{p}\rfloor$. Detailed results for all settings are available in the Appendix.

Intuitive Application

To illustrate the most intuitive use case of the regularization, we limit the allowed degradations in accuracy to 5%. We then pick the best regularization factor λ, which achieves the maximal execution time improvement, while not degrading the accuracy beyond the specified level. Figure 1 reports the corresponding results, where the x-axis separates the different data sets from the UCI repository. The y-axis shows the relative speed improvement with regularization in comparison to the same configuration without regularization. Each box includes random forests with different numbers of trees. The different colors indicate different maximal depths of the trained decision trees and configurations without and with cache optimization.

From the presented results in Figure 1, several observations can be made. First, it can be seen that for trees with a small maximal depth, the improvement in terms of execution time is not reliably observable. Some configurations degrade the speed, some configurations only slightly increase the speed. Considering that a limited maximal depth of 1 only allows for 3 tree nodes, these results are not surprising. Further, it can be observed that the speed improvement grows, the deeper the trees become. A general tendency can be observed, that the deepest trees also benefit most from regularization in terms of execution time improvement. For the data sets, which achieve a significant execution time improvement, a similar scale of improvement for not cache optimized and cache optimized implementations can be observed. This supports the design principle of a regularization, improving both not cache-optimized and cache-optimized implementations in an orthogonal manner. It should be noted that this plot shows the relative execution time in comparison to the not regularized version, i.e., when the cache optimization improves the execution time upon the not cache optimized implementation, this improvement is orthogonal to the regularization. The maximal improvement in terms of execution time can be observed to be more than 75%, i.e., more than $4\times$ faster than without regularization. The data sets, which profit most from the regularization in terms of execution time improvement are adult, bank and magic. Spambase and satlog also show a higher timing improvement than most of the other data sets. Comparing this finding to Table 1 suggests the conclusion that data sets with binary classification can benefit most from the regularization in terms of execution time improvement.

Regularization Trade-Off

Tolerating only accuracy degradation until a configurable threshold is a simplified form of application, which does not allow to make a trade-off. It could still happen, that a higher

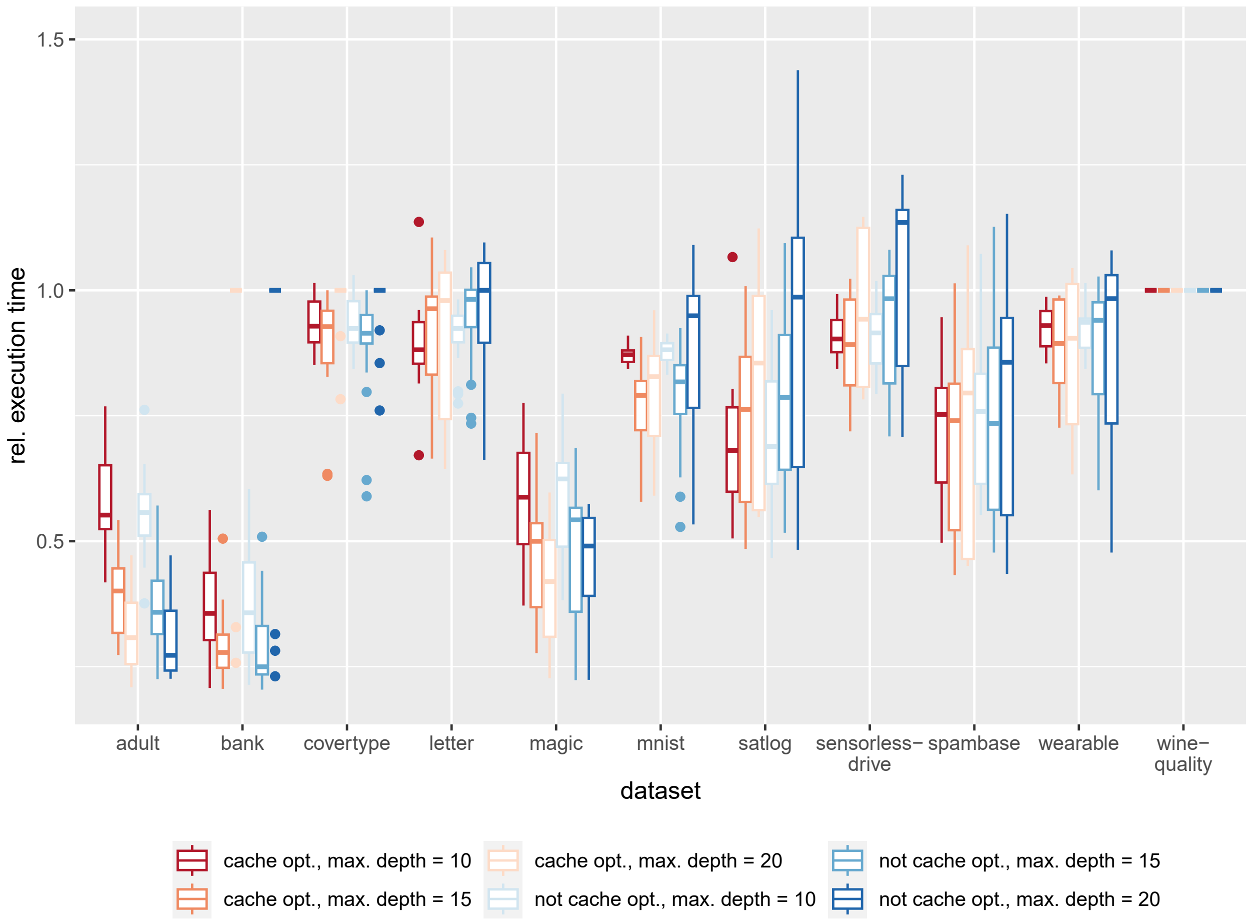


Figure 1. Impact of regularization on execution time across datasets

degree of regularization degrades the accuracy slightly beyond this threshold, but achieves significant faster speed. Such scenarios are evaluated by analyzing the relation between the accuracy drop and the runtime improvement for different degrees of regularization. We illustrate the results of corresponding experiments in Figure 2. The data sets are separated in different subplots. Each configuration, including different amounts of trees and different degrees of regularization. forms one point, which is denoted by the relative execution time improvement to the corresponding not regularized counterpart on the x-axis and the accuracy drop on the y-axis respectively. Cache optimized and not cache optimized implementations are separated by squares and pluses. We further denote the limit of 5% accuracy drop, as used for the previous intuition, by a dotted gray line.

From the results, two different major behaviors can be identified: For certain data sets, namely adult, bank, magic, mnist, spambase and wearable, the configurations with the maximal speed improvement are either Pareto optimal or only have a slight larger degradation in the accuracy than the configurations with the lowest accuracy degradation. This trend can be observed to exist across different maximal depths of trees. For the other configurations, it can be observed that a higher execution time improvement also comes with higher accuracy degradation, especially for deeper trees. It can be as well observed, that cache optimized and not cache optimized implementations form close results, which supports again the design principle of an orthogonal optimization. This suggests the conclusion that, data sets with either binary classification or large sample sizes are better suited for execution time improvement due to regularization without high accuracy impact than other data sets. We call these data sets *regularization robust*.

Limits of Regularization

The previous experiments and result discussions suggest the conclusion, that several data sets have a property, namely *regularization robust*. This property refers to data sets, which can benefit strongly from high degrees of regularization in terms of their improved runtime, while not sacrificing too much accuracy. Previous experiments suggest that the data sets adult, bank, magic, mnist, spambase and wearable

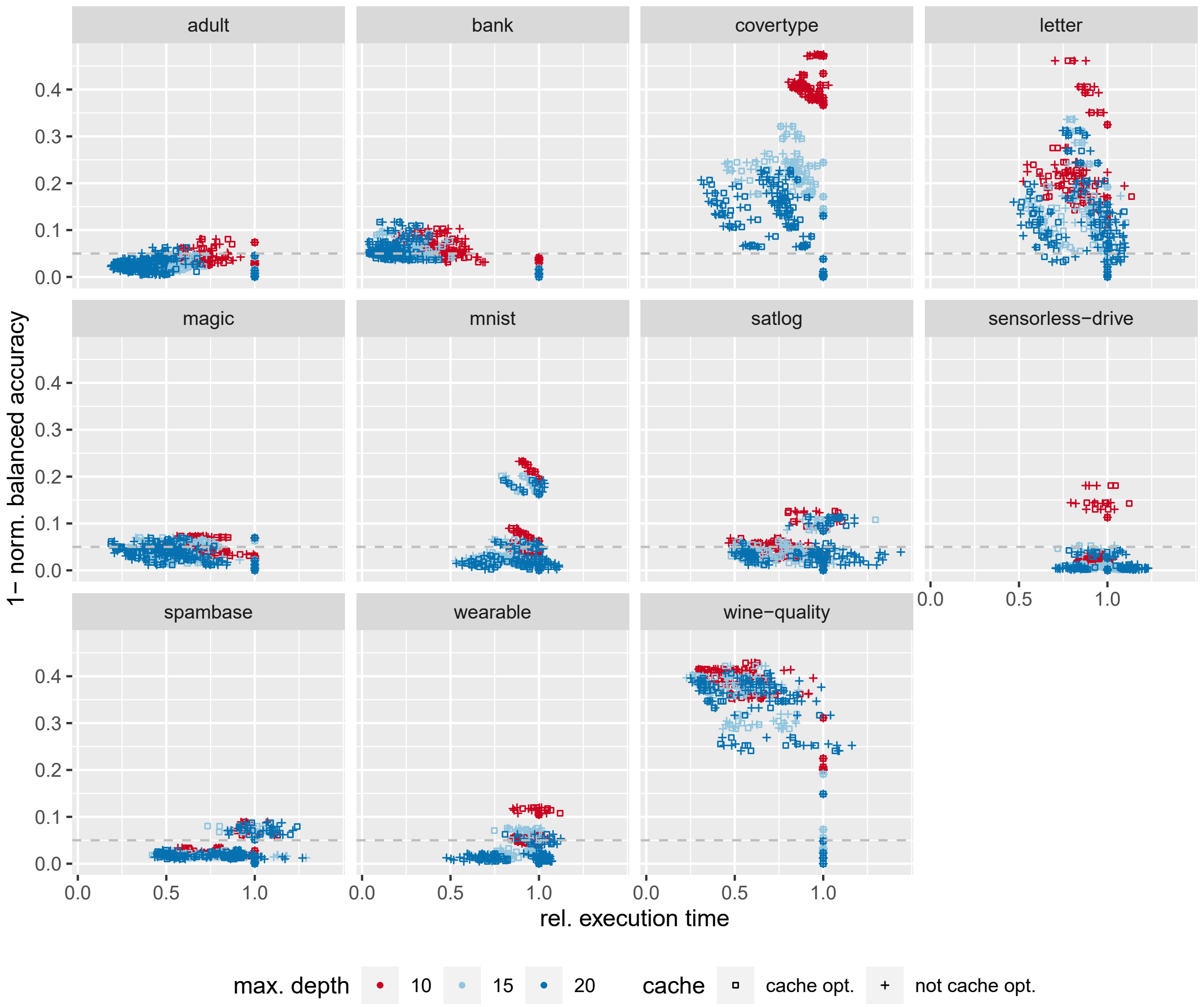


Figure 2. Normalized balanced accuracy values (with respect to the maximum value within each data set) and relative execution times for all data sets with different tree depths and separated by cache optimization or not.

have this property to a certain degree. In order to investigate this property even further, we use regularization tuning as explained in Section 3.3 to stop increasing the regularization factor once the expected increase in speed falls under a change threshold of 5%. In other words, if no further execution time improvement is achieved, the regularization factor is not further increased.

We illustrate the amount of configurations of a data set (i.e. different number of trees in an ensemble and different implementation strategies) with their maximal regularization factor in Figure 3. To get a general picture of the influence of the regularization factor λ, experiments with $\lambda \in [0, 40]$ are run regardless of any metric. Next, regularization tuning is used to stop once the factor is expected to not make a significant difference to execution speed. It can be observed that for certain data sets, a reasonable amount of configuration can profit from high regularization factors. These data sets are adult, bank, magic, spambase and wearable. Except the mnist data set, which has an exceptional high number of classes, this is exactly the list of data sets, which are encountered as regularization robust before. Hence, by only investigating the data set properties upfront, an assertion can be made whether the data set is regularization robust and thus may profit from strong degrees of regularization. We have seen that this often holds for binary classification problems or very large data sets.

4.3 Simulation with Synthetic Data

In order to better understand the effects of tree regularization on binary classification datasets, particularly with respect to *regularization robustness*, we conducted experiments with synthetic data. We first describe how the data is generated

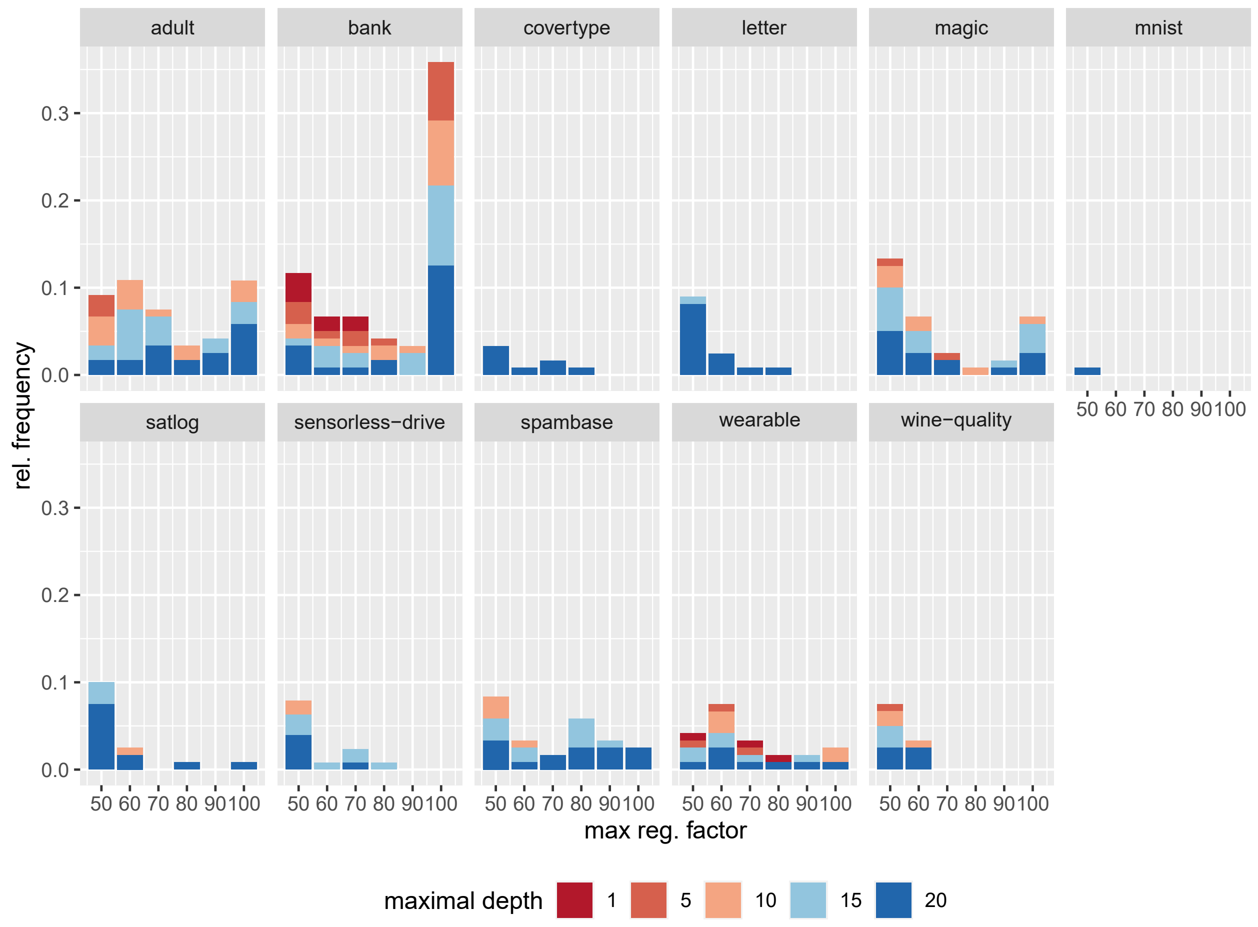


Figure 3. Relative frequency of maximum regularization factors in each data set, focusing on the range of values above 40.

and subsequently present the measured results.

Simulation Setup

We consider a binary classification problem $Y \in \{0, 1\}$ with ten real-valued features $X_1, \ldots, X_{10}$ for which we specify different distributions, dependencies and underlying models, described in the following. We model the last five features $X_6, \ldots, X_{10}$ as independent and uniformly distributed random variables from $[0, 10]$, independent of the first five features $X_1, \ldots, X_5$. For the first five features we consider three different dependence structures as summarized in Table 2. In the first setting (Independent), we consider completely independent features, where each feature follows a distinct mixed normal distribution $Z_{\ell,k} = lW_1 + (1 - \ell)\mathcal{N}(\Delta_\mu + k, 1)$ with $\Delta_\mu \in \{1, 3, 5, 8\}$ and $W_1 \sim \mathcal{N}(1, 1)$. The concrete choices for ℓ and k are given in Table 2. In the other two settings, we model a weak dependence between X_2, X_3 and X_4 (second last column) and a strong dependence between X_1 and X_5 (last column), respectively. Note that for the first feature, $k = b$ is not held constant, but is systematically varied from 0.2 to 0.9 to provide some degree of adjustment to modulate the class balance. Having fixed the dependencies among the features,

Table 2. Distributions of $X_1, \ldots, X_5$ used in the simulation studies with $p \in \{0.2, 0.5, 0.7, 0.9\}$.

Features	Independent	Weakly dependent	Strongly dependent
X_1	$Z_{p,1}$	$Z_{p,1}$	$Z_{p,1}$
X_2	$Z_{0.1,-5}$	$Z_{0.1,-5}$	$Z_{0.1,-5}$
X_3	$Z_{0.5,2}$	$Z_{0.5,2}$	$Z_{0.5,2}$
X_4	$Z_{0.3,3}$	$Z_{0.1,-5} + Z_{0.5,2}$	$Z_{0.1,-5} + Z_{0.5,2}$
X_5	$Z_{0.8,-2}$	$Z_{0.8,-2}$	$Z_{0.5,2} + 0.5Z_{p,1}$

we now model the dependencies with the outcome Y. In this study, we investigate three different relationships between Y and $X_1, \ldots, X_{10}$ by incorporating different dependencies and correlation structures through logical rules. The settings range from a simple dependence of the outcome solely on the first feature X_1 (S1), whereby Y equals 1 if the realization derived from X_1 originates from W_1 of the normal mixed distribution. The more complex dependencies involve the first three or five features. The concrete details are illustrated in Table 3.

Table 3. Dependent models between the output and some of the features. Here O_i refers to the event that the realization of the feature X_i originates from the first part of its mixed normal distribution.

Setting	$Y = 1$
S1	O_1
S3	(O_1 and O_4) or $\neg O_2$
S5	(O_1 and $\neg O_3$) or ($\neg O_5$ and $\neg O_4$) or O_2

For each setting, we generated samples of size num from the respective model with $num \in \{100, 200, 500\}$. The regularization strength λ and the number of trees are varied as described in Section 4.1. The same applies to the hyperparameter $max\_features$.

Results

For ease of presentation, we focus on the most important results and general trends. Studying the simulation study results for all configurations, we observed that changes in the dependency structure of the feature, the relationship between features and outcome and the size of the inner bootstrap sample ($max\_features$) of the random forest had no large effect on the behavior of trees under the regularization. In comparison, the balance of the prediction classes, regulated by the balance parameter b and Δ_μ, and the sample size n were the driving forces for changes in the influence of the regularization.

Examining the effects of sample size, we find results consistent with those of the previous section. As the sample size increases, a greater improvement in execution time is observed along with a decrease in accuracy. Details can be found in the appendix

Results for different combinations of the balance parameters are shown in Figure 4. The results shown in Figure 4 are for $max\_features = 6, n = 100, num = 100$, independent characteristics and the S3 model for the outcome. We present results for three combinations of b and Δ_μ: $b = 0.9, \Delta_\mu = 8$ (red), $b = 0.7, \Delta_\mu = 3$ (green) and $b = 0.5, \Delta_\mu = 1$ (blue). These combinations were selected because of their different strengths of balance. Red is the most unbalanced and blue is the most balanced. The x-axis for each of the first two plots shows an increasing regularization factor, the y-axis shows the balanced accuracy in Figure 4 (left), and the relative execution time in Figure 4 (center). The x-axis for the figure on the right shows the 1-balanced accuracy, and the y-axis shows the relative execution time. The different shapes indicates wether a cache-optimized version is used or not. For all three settings, there is a clear trend towards faster relative execution times as the regularization factor increases. However, the improvement diminishes as soon as the regularization factor reachs 50. When examining the effect of regularization on balanced accuracy in these settings (plot on the left), it is noticeable that the blue and green settings show a more significant decrease in accuracy than the red settings. This suggests that the balance of classes influences the effect of regularization, with unbalanced classes showing greater sensitivity to regularization. By looking at the Pareto front (plot on the right), we can see that the red setting dominates the others for most configurations.

4.4 Discussion

The previously presented results indicate that the introduction of regularization offers a trade-off between a degradation of accuracy and the improvement of execution time. While for shallow decision trees the regularization generally cannot offer a large spectrum for the trade-off and quickly degrades to extreme cases, a wider spectrum for the trade-off is offered for deeper tree models in general. It is worth noticing that the degradation of the accuracy is usually less by one order of magnitude than the gained speed improvement, when a moderate amount of regularization is chosen.

Investigating the dataset properties itself, the comparison between synthetic and real data sets shows that a major influence on the effectiveness of the regularization is put by the balance of the prediction classes. Table 4 shows the Chi-Squared values for the UCI datasets, where a high values indicate a potential high imbalance of the distribution in the prediction classes. It should be noted that these values can only be interpreted for binary classification datasets (indicated in bold), since the structure of the imbalance becomes too complex for multi label classifications. It can be seen, that for the adult and bank dataset, which are observed to provide high speed improvements on minimal accuracy degradation, a relatively high Chi-Square value can be observed. This aligns with the observations from the synthetic datasets, where highly imbalanced distributions also allow high speed improvements on minimal accuracy

Table 4. Chi-Square Values of UCI Datasets

Dataset	Chi-square
Adult	**6,556.066**
Bank Marketing	**18,552.56**
Covertype	71,3450.9
Letter	19.3273
Magic	**1271.086**
MNIST	139.705
Satlog	504.379
Spambase	**153.515**
Sensorless Drive	2.2707
Wearable Computing	33,934.62
Wine Quality	6,383.254

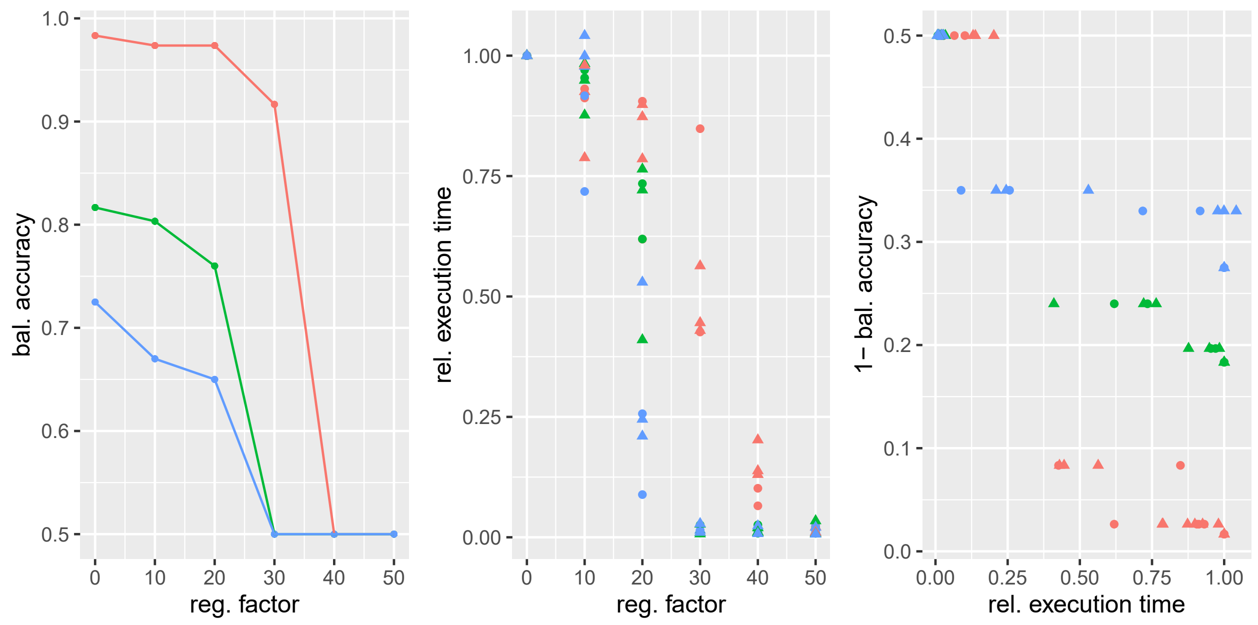


Figure 4. Simulation results of the regularization for varying b and Δ_μ: $b = 0.9, \Delta_\mu = 8$ (red), $b = 0.7, \Delta_\mu = 3$ (green) and $b = 0.5, \Delta_\mu = 1$ (blue). The different shapes of the points indicate whether a cache-optimized version is used (circle) or not (triangle).

degradations.

From the perspective of an user, regularization should be considered for deeper tree models, since the effectiveness for small trees is highly limited. When a dataset is used, which by default is imbalanced, regularization can be generally turned up further and gain more speed improvement while not degrading the accuracy much.

5 Conclusion and Outlook

Deploying machine learning models efficiently on resource-constrained devices requires a carefully tuned model shape in terms of model size and a hardware-close implementations, of which the state-of-the-art cache-aware optimizations are prominent for random forests and decision trees. In this work, we present a method to regularize the impurity computation and reward highly asymmetric distributions in the training process of decision trees, which provokes uneven probability distributions (i.e., uneven splits) for offering high optimization potential.

To examine the effectiveness of our method, we conduct extensive experimental evaluation on synthetic datasets and on UCI datasets. The evaluation results show that a large execution time reduction of up to $\approx 4\times$ can be gained in many cases while degrading the target accuracy by a few percent. The user can either specify an acceptable threshold of sacrificable accuracy degradation and derive the optimal regularized result or can make an own trade-off by choosing between pareto optimal points in the scope of accuracy degradation and execution time improvement. We can further categorize data sets as regularization robust, when they are either binary classification data sets or have a high amount of samples. Such data sets may benefit strongly from regularization.Spending a deeper focus on the property of being regularization robust, we see a dependency to the sample size in the synthetically generated data sets. We further observe a strong dependency with the imbalance of the synthesized data sets and the effectiveness of regularization, supporting the initial design principle. An implementation in scitkit-learn is openly available.

For future work, the application of regularization across the random forest structure should be studied, instead of considering single trees in separation. For instance, the dataset can be split into subsets with strong dependencies for the training of different trees, making the regularization more effective. Furthermore, it can be considered to have heterogeneous degrees of regularization for other tree-based ensembles.

References

Asadi, N., Lin, J., and de Vries, A. P. Runtime optimizations for tree-based machine learning models. *IEEE Transactions on Knowledge and Data Engineering*, 26 (9):2281–2292, Sept 2014. ISSN 1041-4347. doi: 10.1109/TKDE.2013.73.

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A.-L., Deng, D., and Lindauer, M. Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *WIREs Data Mining and Knowledge Discovery*, 13(2), January 2023. doi: 10.1002/widm.1484.

Blackard, J. A. and Dean, D. J. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151, 1999.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. CART. *Classification and regression trees*, 1984. doi: 10.1201/9781315139470.

Buschjäger, S., Chen, J.-J., Chen, K.-H., Günzel, M., Hakert, C., Morik, K., Novkin, R., Pfahler, L., and Yayla, M. Margin-maximization in binarized neural networks for optimizing bit error tolerance. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 673–678, 2021. doi: 10.23919/DATE51398.2021.9473918.

Buschjäger, S. and Morik, K. Decision tree and random forest implementations for fast filtering of sensor data. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(1):209–222, 2018.

Chen, K.-H., Su, C., Hakert, C., Buschjäger, S., Lee, C.-L., Lee, J.-K., Morik, K., and Chen, J.-J. Efficient realization of decision trees for real-time inference. *ACM transactions on embedded computing systems*, 21(6):1–26, 2022.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., and Reis, J. Modeling wine preferences by data mining from physicochemical properties. *Decision support systems*, 47(4):547–553, 2009.

Dato, D., Lucchese, C., Nardini, F. M., Orlando, S., Perego, R., Tonellotto, N., and Venturini, R. Fast ranking with additive ensembles of oblivious and non-oblivious regression trees. *ACM Transactions on Information Systems*, 2016. ISSN 15582868. doi: 10.1145/2987380.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Frey, P. W. and Slate, D. J. Letter recognition using holland-style adaptive classifiers. *Machine learning*, 6:161–182, 1991.

Hakert, C., Chen, K.-H., and Chen, J.-J. Immediate split trees: Immediate encoding of floating point split values in randomforests. *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2022a.

Hakert, C., Chen, K.-H., and Chen, J.-J. Flint: Exploiting floating point enabled integer arithmetic for efficient random forest inference. *arXiv preprint arXiv:2209.04181*, 2022b.

Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Kim, C., Chhugani, J., Satish, N., Sedlar, E., Nguyen, A., Kaldewey, T., Lee, V., Brandt, S., and Dubey, P. FAST: Fast architecture sensitive tree search on modern CPUs and GPUs. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pp. 339–350. ACM, 2010.

Kohavi, R. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In *Second International Conference on Knowledge Discovery and Data Mining*, 1996.

Lakshminarayanan, B., Roy, D. M., and Teh, Y. W. Mondrian forests: Efficient online random forests. *Advances in neural information processing systems*, 27, 2014.

Liaw, A., Wiener, M., et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

Lucchese, C., Perego, R., Nardini, F. M., Tonellotto, N., Orlando, S., and Venturini, R. Exploiting CPU SIMD extensions to speed-up document scoring with tree ensembles. In *SIGIR 2016 - Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2016. ISBN 9781450342902. doi: 10.1145/2911451.2914758.

Mentch, L. and Zhou, S. Randomization as regularization: A degrees of freedom explanation for random forest success. *The Journal of Machine Learning Research*, 21(1):6918–6953, 2020.

Moro, S., Cortez, P., and Rita, P. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

Nakandala, S., Saur, K., Yu, G.-I., Karanasos, K., Curino, C., Weimer, M., and Interlandi, M. A tensor compiler for unified machine learning prediction serving. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pp. 899–917, 2020.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

Prenger, R., Chen, B., Marlatt, T., and Merl, D. Fast map search for compact additive tree ensembles (cate). Technical report, Tech. rep., Lawrence Livermore National Laboratory (LLNL), Livermore, CA, 2013.

Probst, P., Wright, M. N., and Boulesteix, A.-L. Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: data mining and knowledge discovery*, 9(3):e1301, 2019.

Scheffer, T. Nonparametric regularization of decision trees. In *Machine Learning: ECML 2000: 11th European Conference on Machine Learning Barcelona, Catalonia, Spain, May 31–June 2, 2000 Proceedings 11*, pp. 344–356. Springer, 2000.

Tabanelli, E., Tagliavini, G., and Benini, L. Optimizing random forest-based inference on risc-v mcus at the extreme edge. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(11):4516–4526, 2022. doi: 10.1109/TCAD.2022.3199903.

Ugulino, W., Cardador, D., Vega, K., Velloso, E., Milidiú, R., and Fuks, H. Wearable computing: Accelerometers' data classification of body postures and movements. In *Advances in Artificial Intelligence-SBIA 2012: 21th Brazilian Symposium on Artificial Intelligence, Curitiba, Brazil, October 20-25, 2012. Proceedings*, pp. 52–61. Springer, 2012.

Van Essen, B., Macaraeg, C., Gokhale, M., and Prenger, R. Accelerating a random forest classifier: Multi-core, gp-gpu, or fpga? In *Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on*, pp. 232–239. IEEE, 2012.

Wright, M. N. and Ziegler, A. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17, 2017. doi: 10.18637/jss.v077.i01.

Wu, M., Hughes, M., Parbhoo, S., Zazzi, M., Roth, V., and Doshi-Velez, F. Beyond sparsity: Tree regularization of deep models for interpretability. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

APPENDIX

We present the complete results regarding $max\_features$ and the different execution types. The results are presented in graphical form.

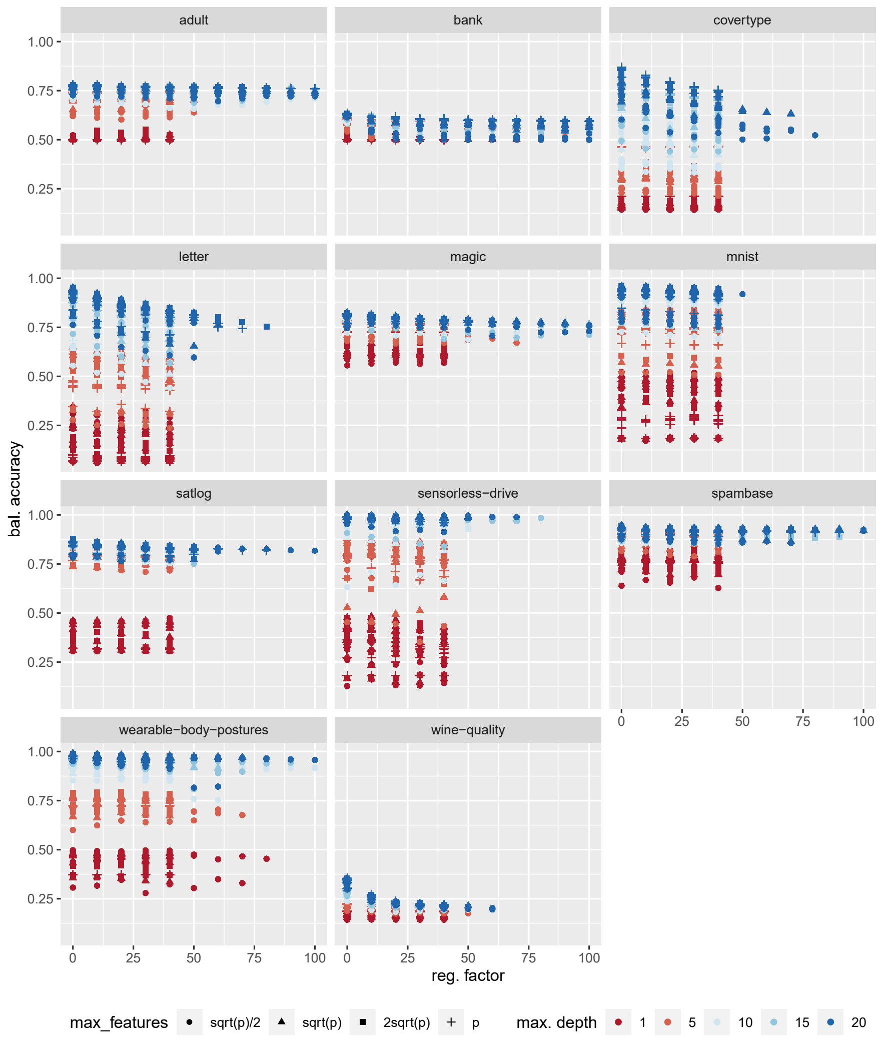


Figure 5. Evaluation of the balanced accuracy for the UCI datasets

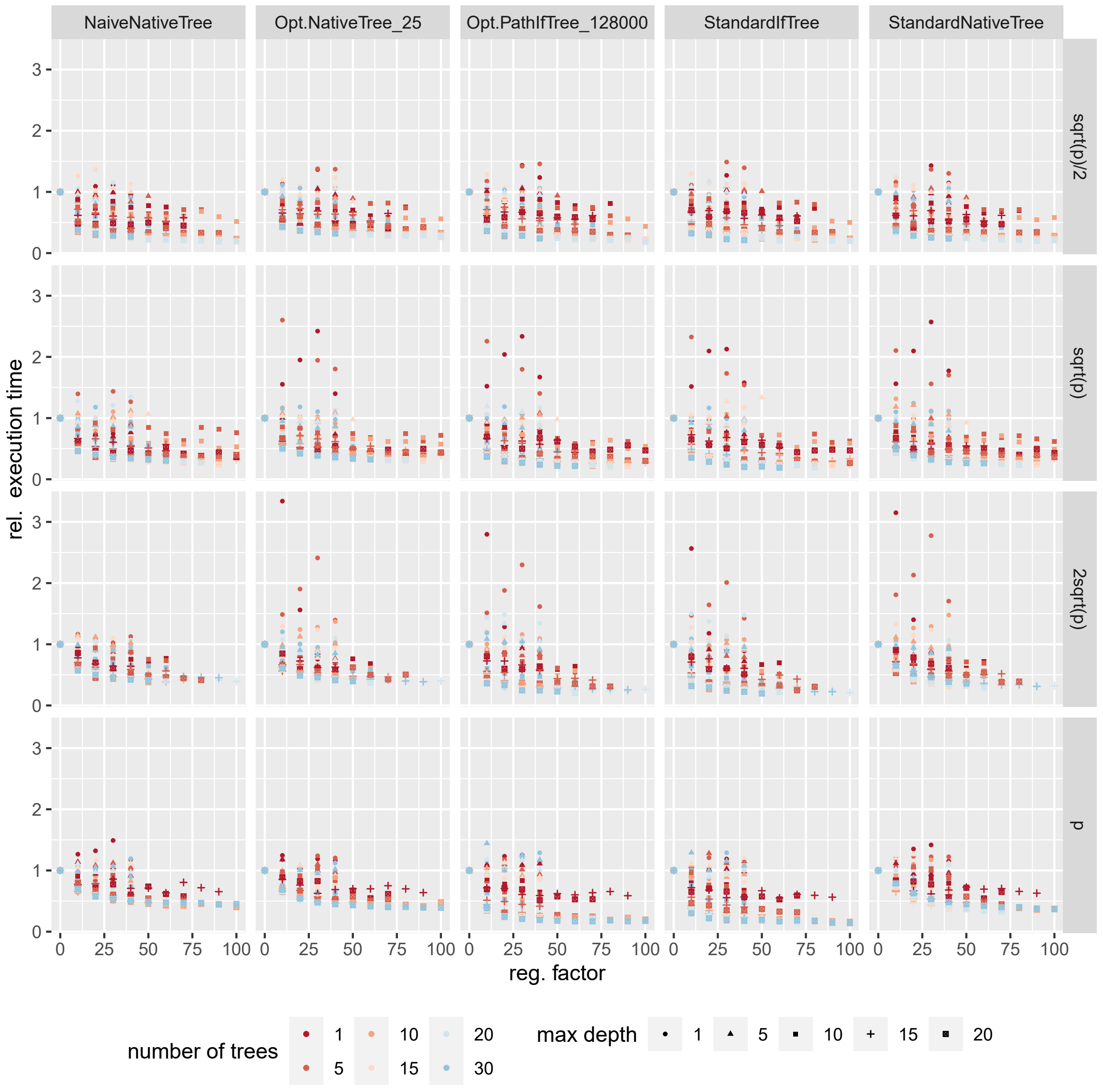


Figure 6. Evaluation of the relative execution time for the adult dataset separated by the execution type $max\_features$, the maximum depth (shape of the points) and the number of trees (color).

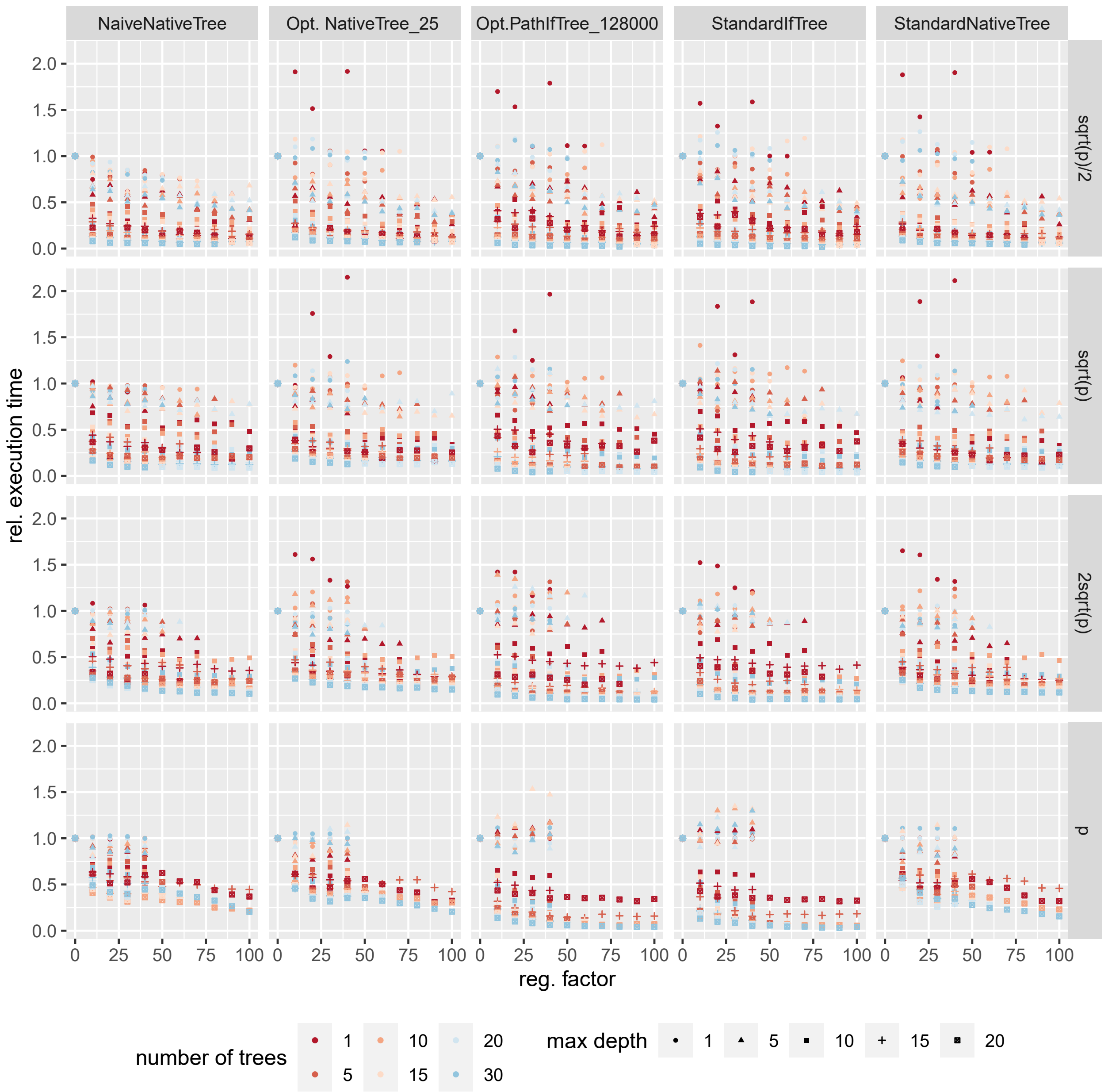


Figure 7. Evaluation of the relative execution time for the bank dataset separated by the execution type $max\_features$, the maximum depth (shape of the points) and the number of trees (color).

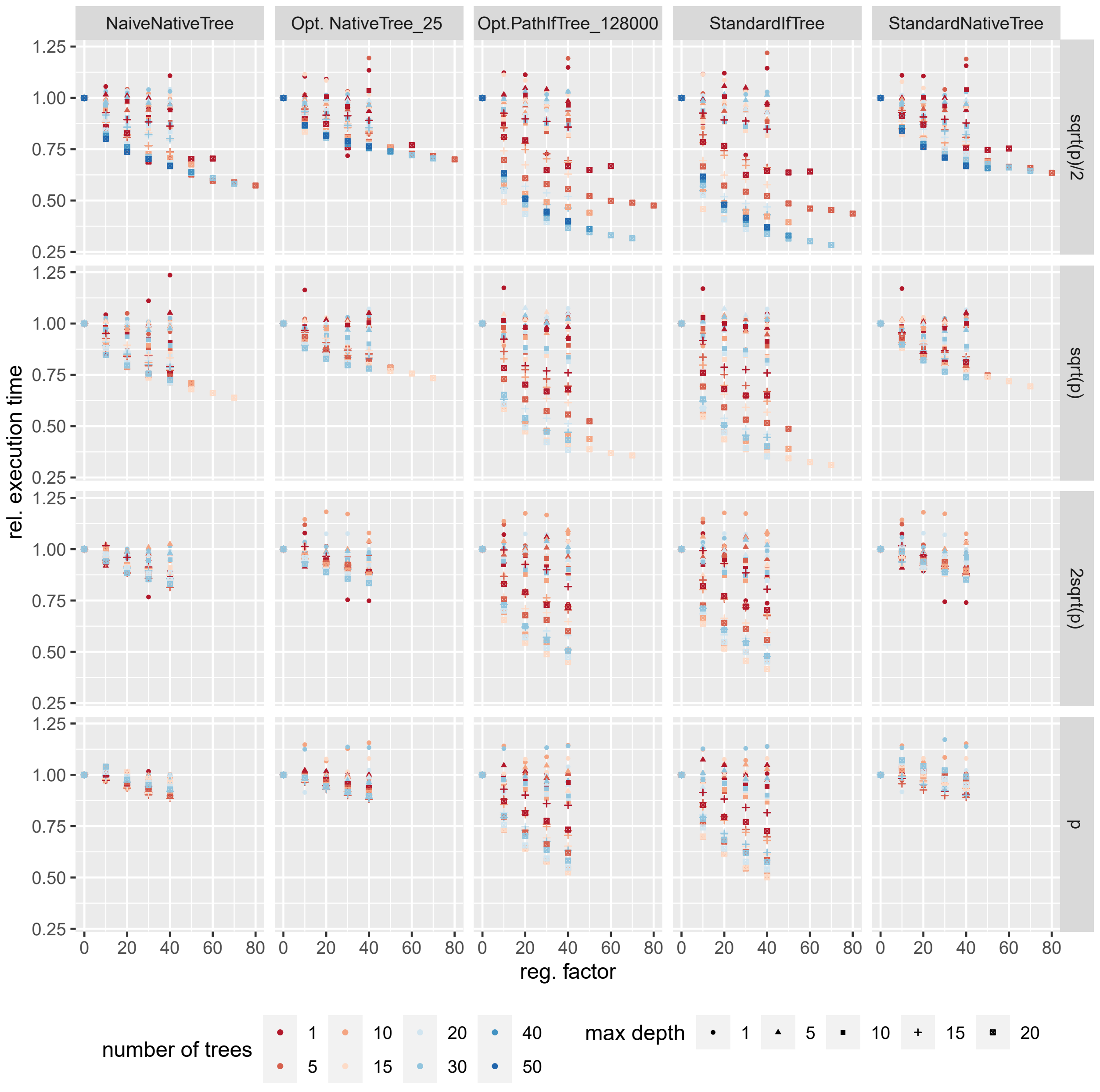


Figure 8. Evaluation of the relative execution time for the covertype dataset separated by the execution type $max\_features$, the maximum depth (shape of the points) and the number of trees (color).

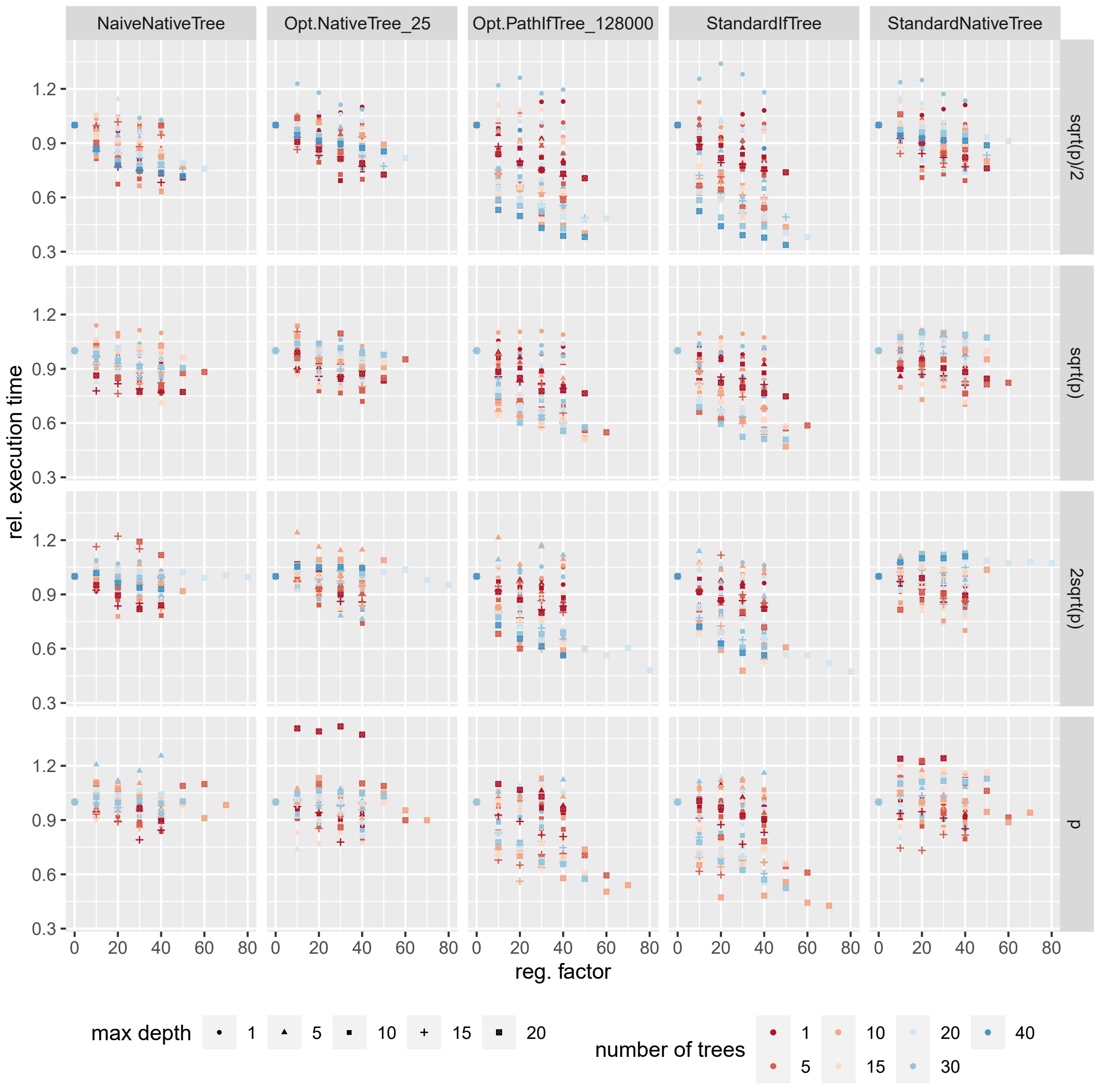


Figure 9. Evaluation of the relative execution time for the letter dataset separated by the execution type $max\_features$, the maximum depth (shape of the points) and the number of trees (color).

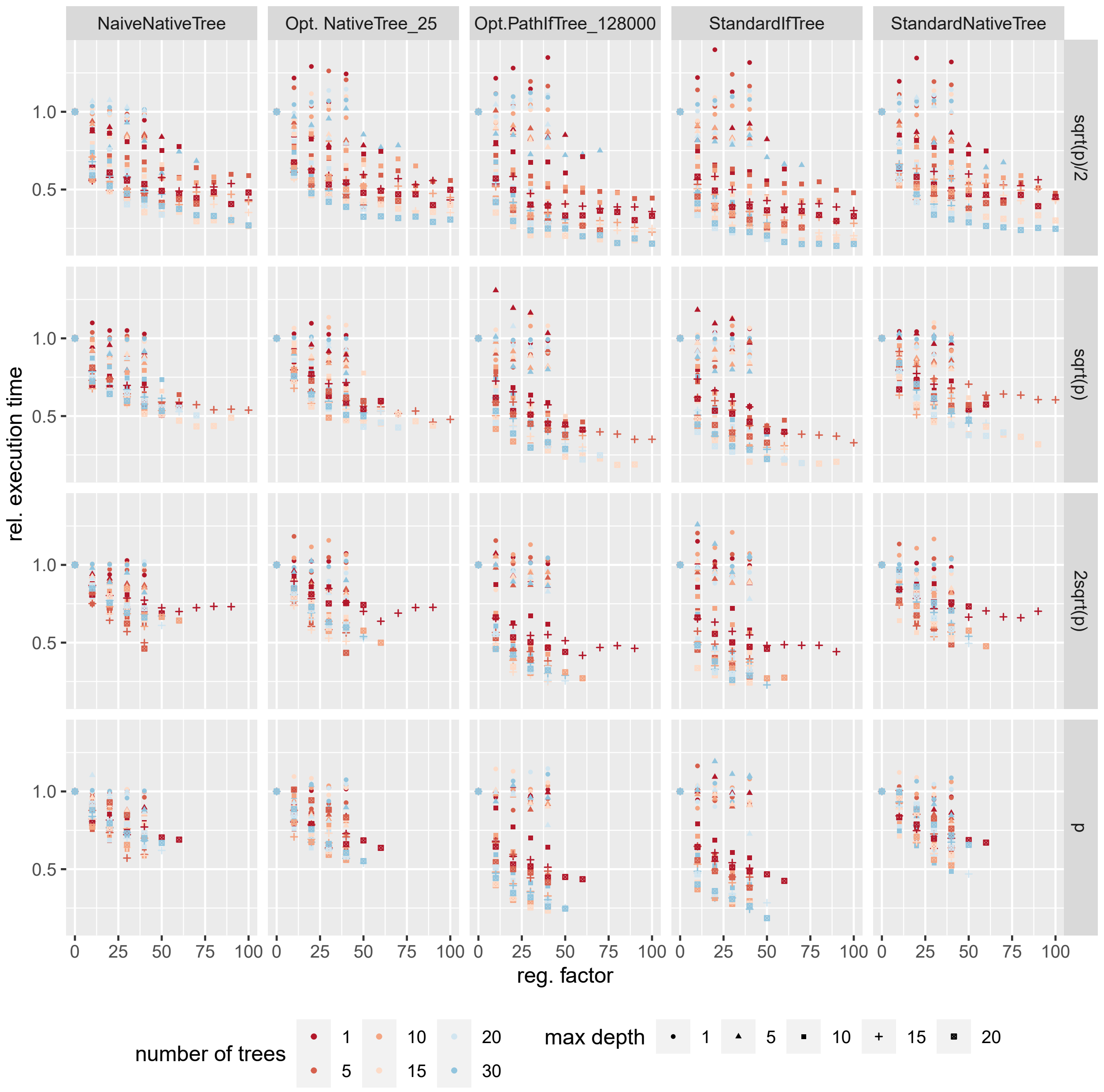


Figure 10. Evaluation of the relative execution time for the magic dataset separated by the execution type $max\_features$, the maximum depth (shape of the points) and the number of trees (color).

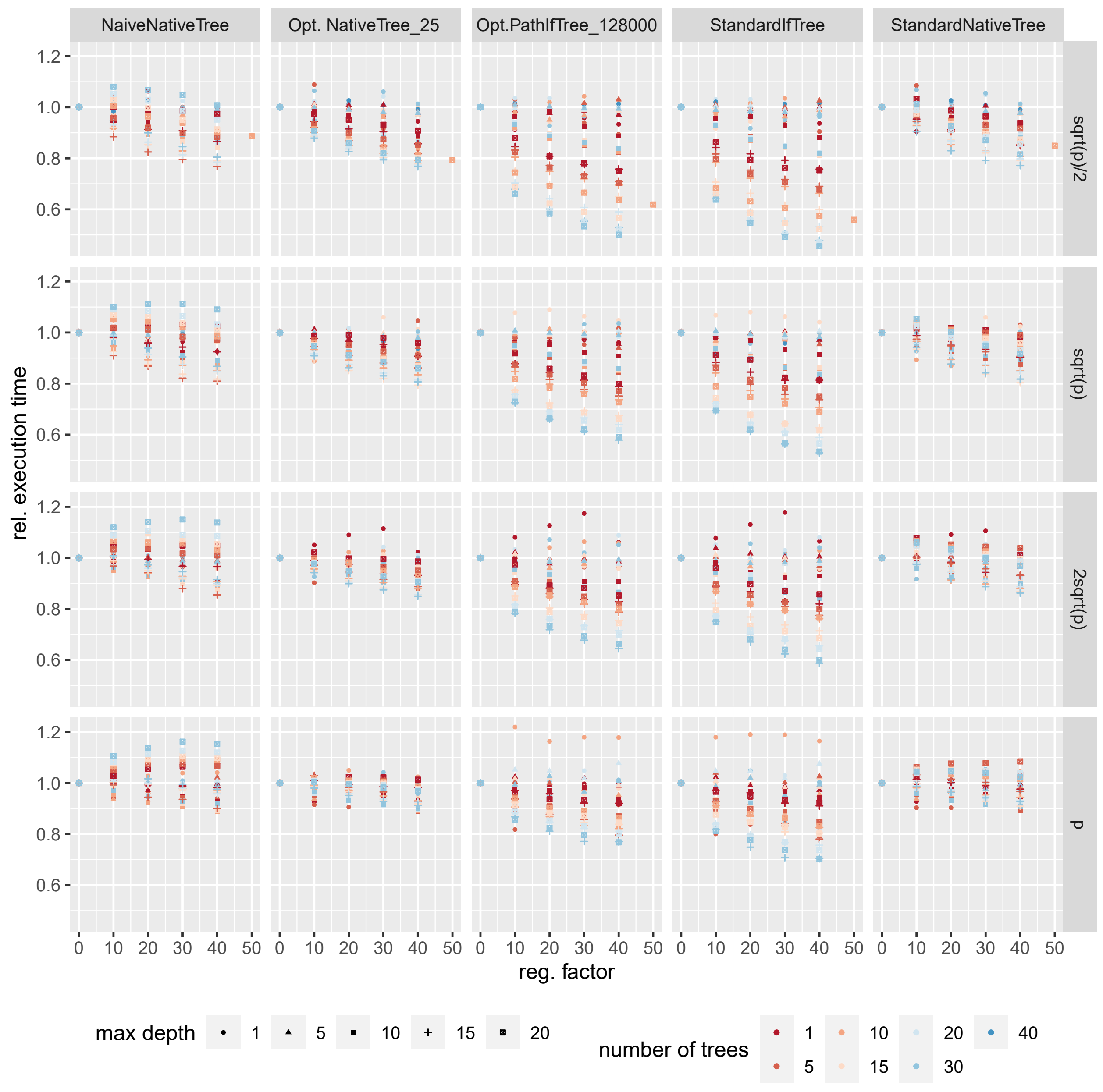


Figure 11. Evaluation of the relative execution time for the mnist dataset separated by the execution type $max\_features$, the maximum depth (shape of the points) and the number of trees (color).

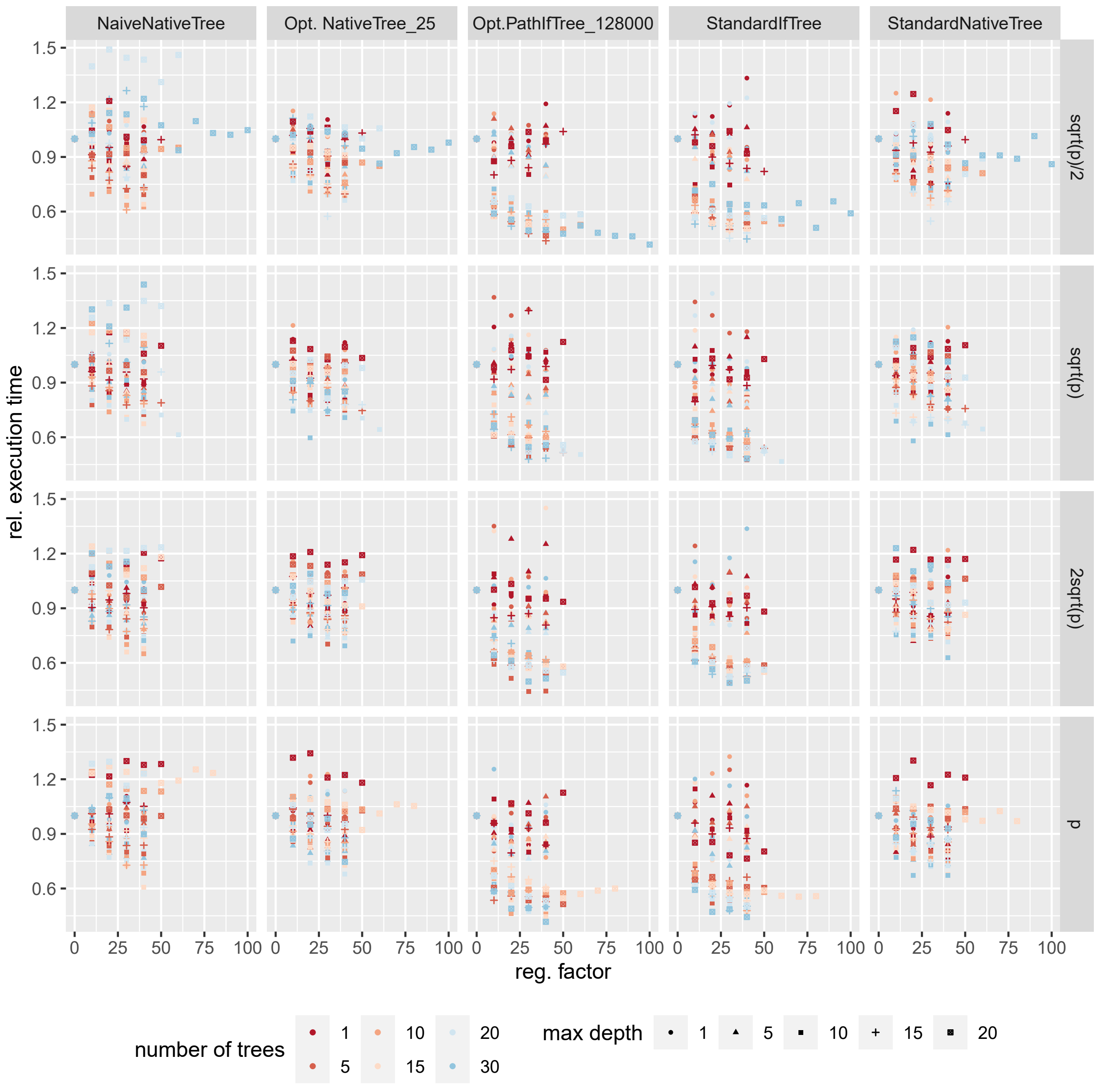


Figure 12. Evaluation of the relative execution time for the satlog dataset separated by the execution type $max\_features$, the maximum depth (shape of the points) and the number of trees (color).

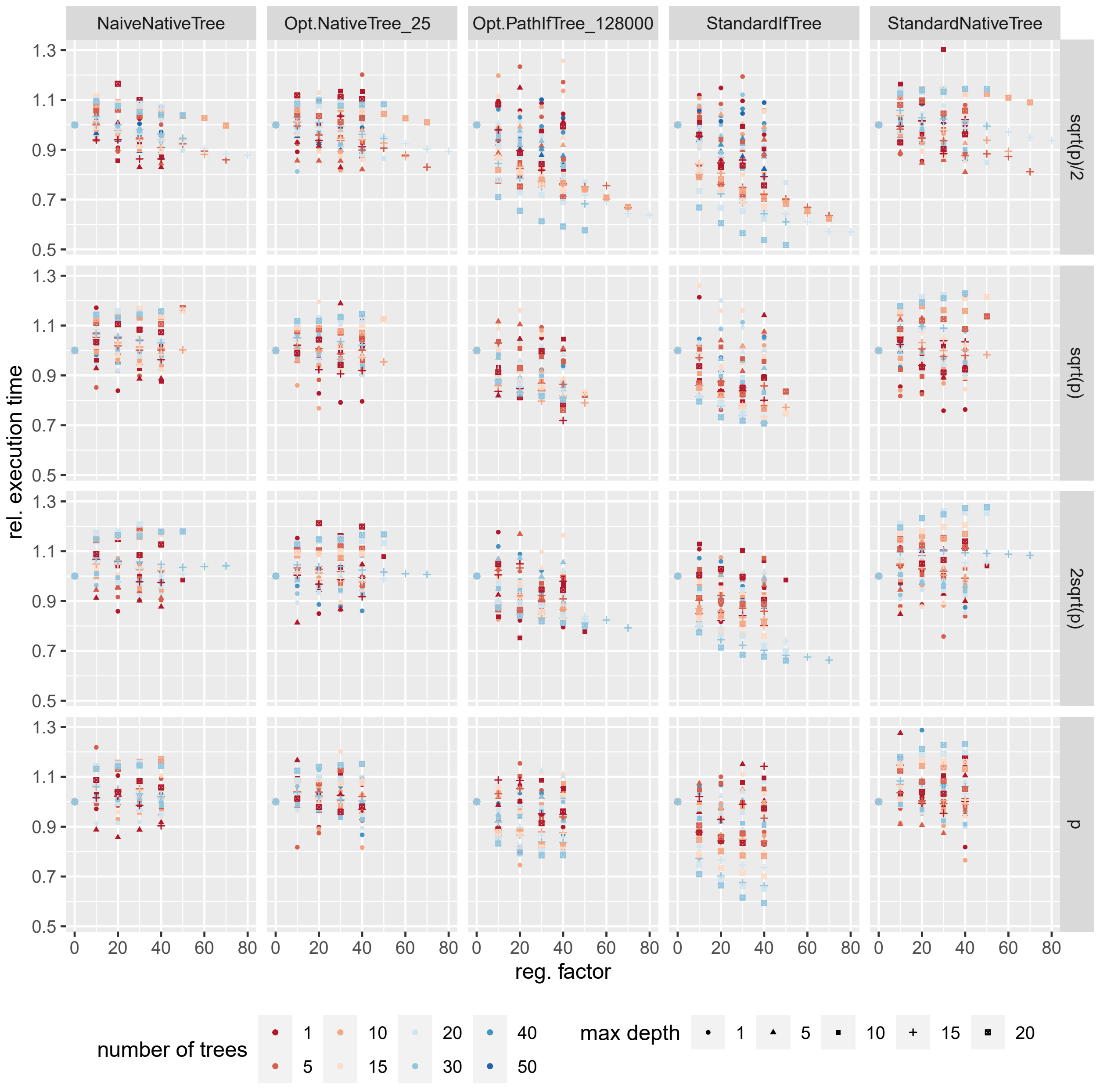


Figure 13. Evaluation of the relative execution time for the sensorless dataset separated by the execution type $max\_features$, the maximum depth (shape of the points) and the number of trees (color).

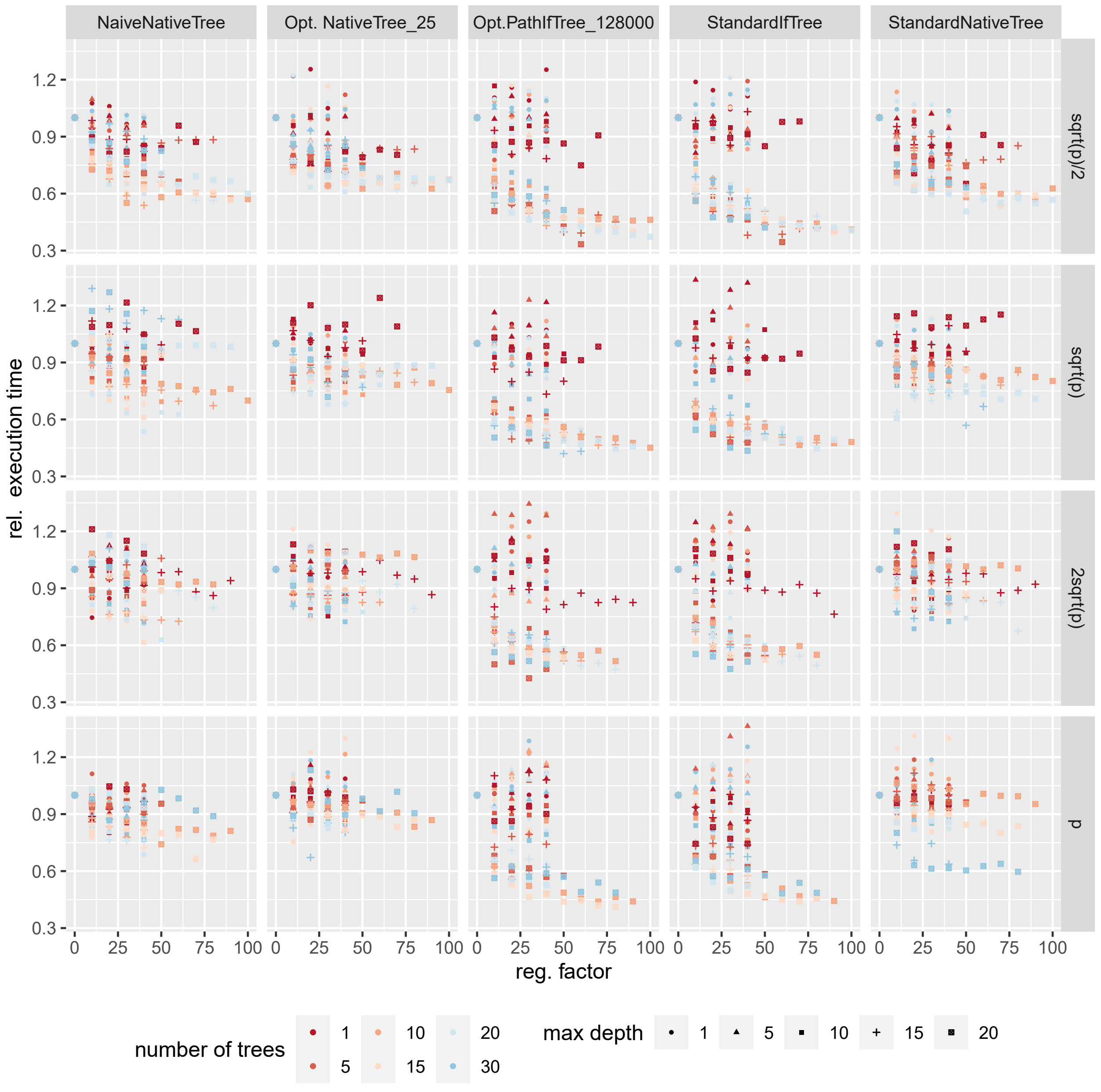


Figure 14. Evaluation of the relative execution time for the spambase dataset separated by the execution type $max\_features$, the maximum depth (shape of the points) and the number of trees (color).

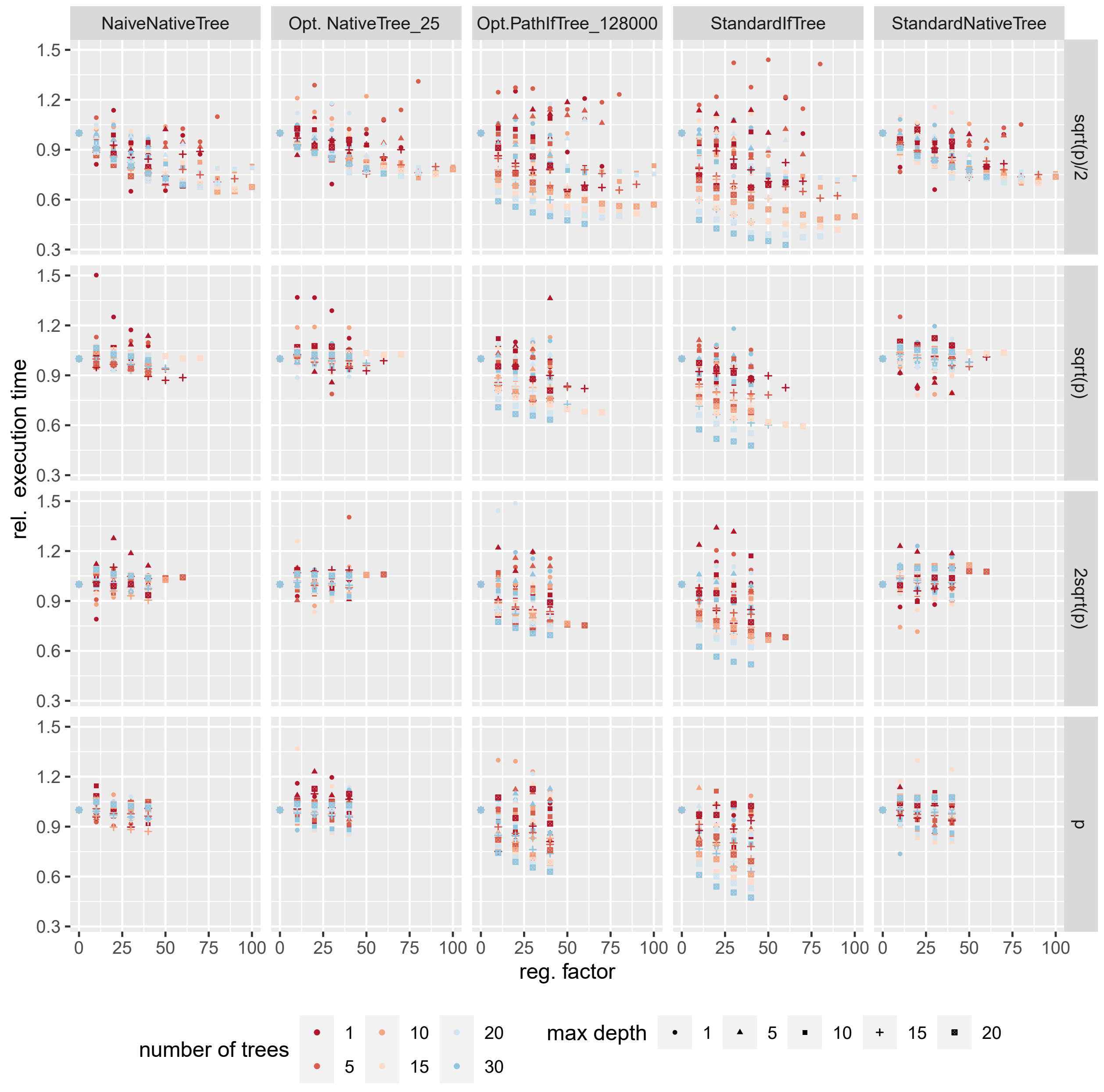


Figure 15. Evaluation of the relative execution time for the wearable dataset separated by the execution type $max\_features$, the maximum depth (shape of the points) and the number of trees (color).

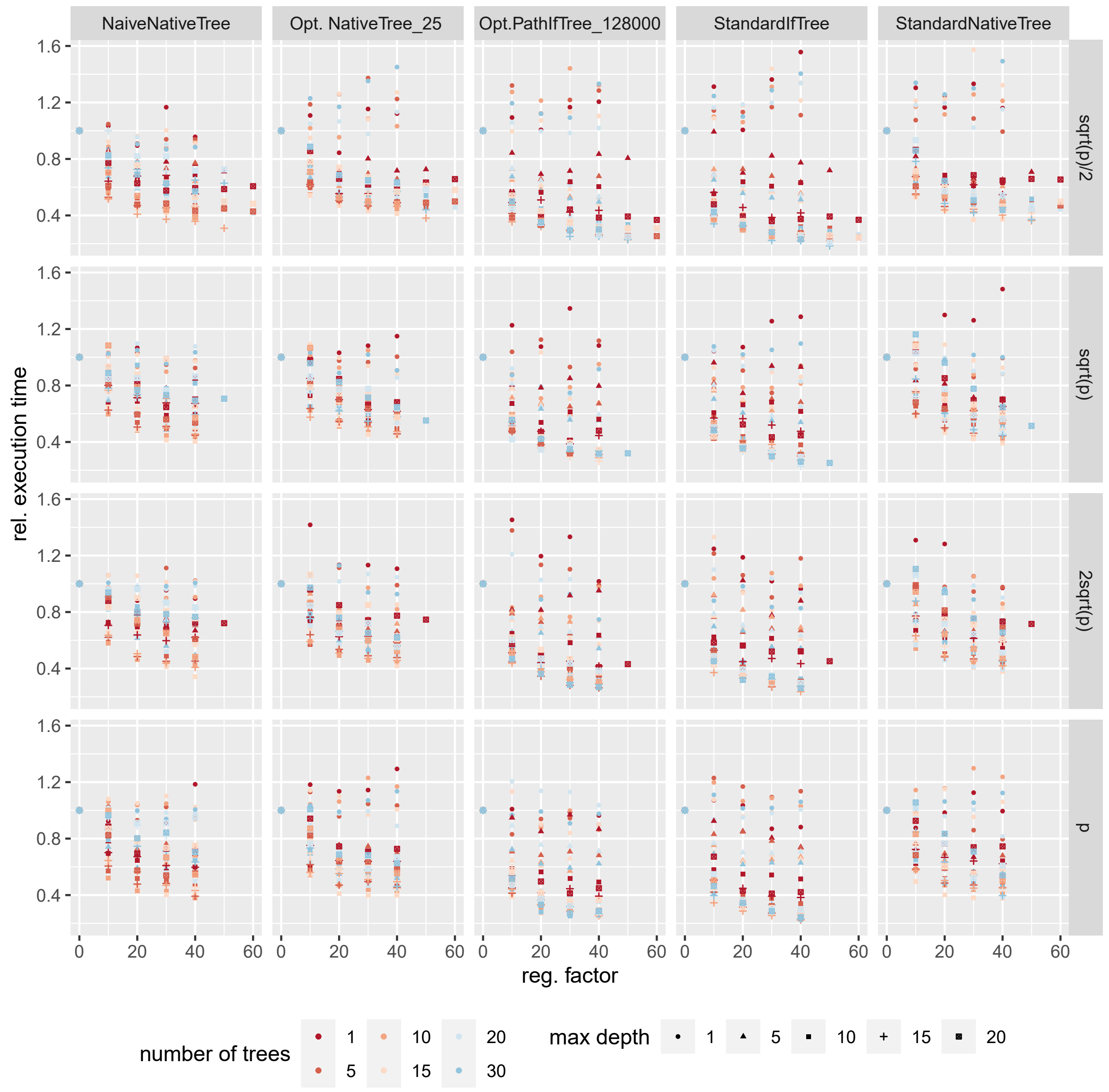


Figure 16. Evaluation of the relative execution time for the wine-quality dataset separated by the execution type $max\_features$, the maximum depth (shape of the points) and the number of trees (color).

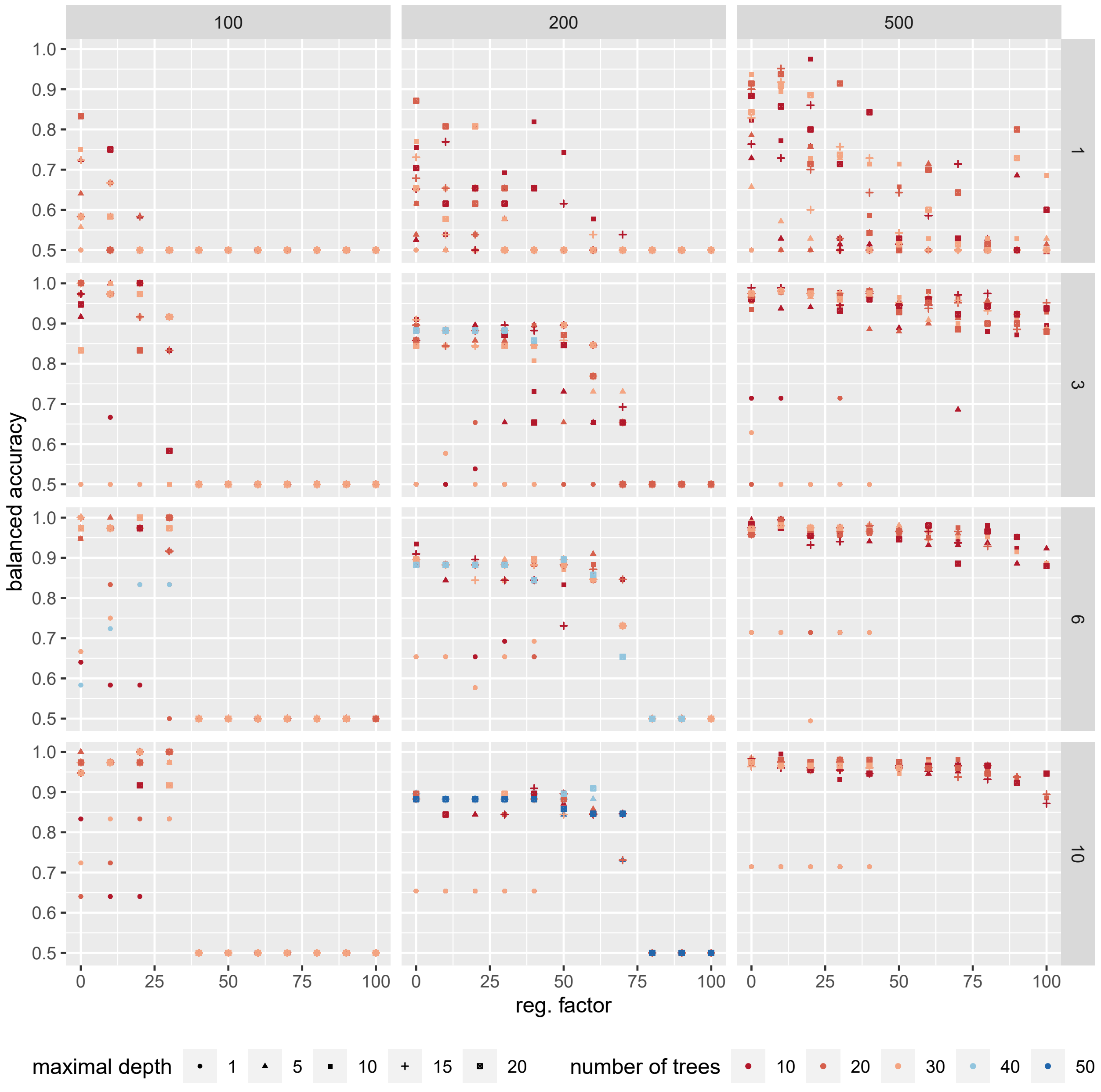


Figure 17. Balanced accuracy evaluation for synthetic data (red setting) with varying sample size, $max\_features$, maximum depth (point shape) and number of trees (color).

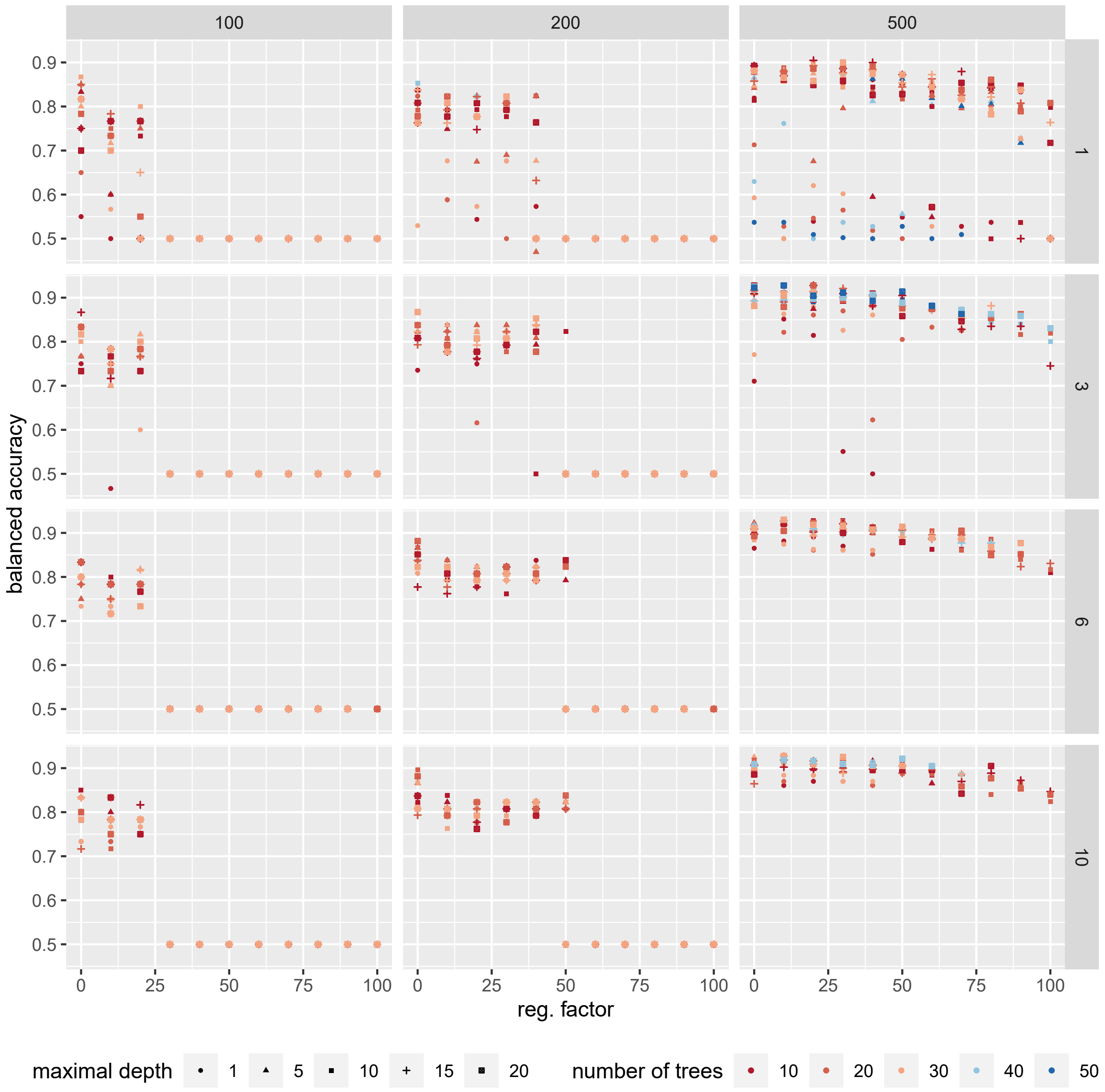


Figure 18. Balanced accuracy evaluation for synthetic data (green setting) with varying sample size, $max\_features$, maximum depth (point shape) and number of trees (color).

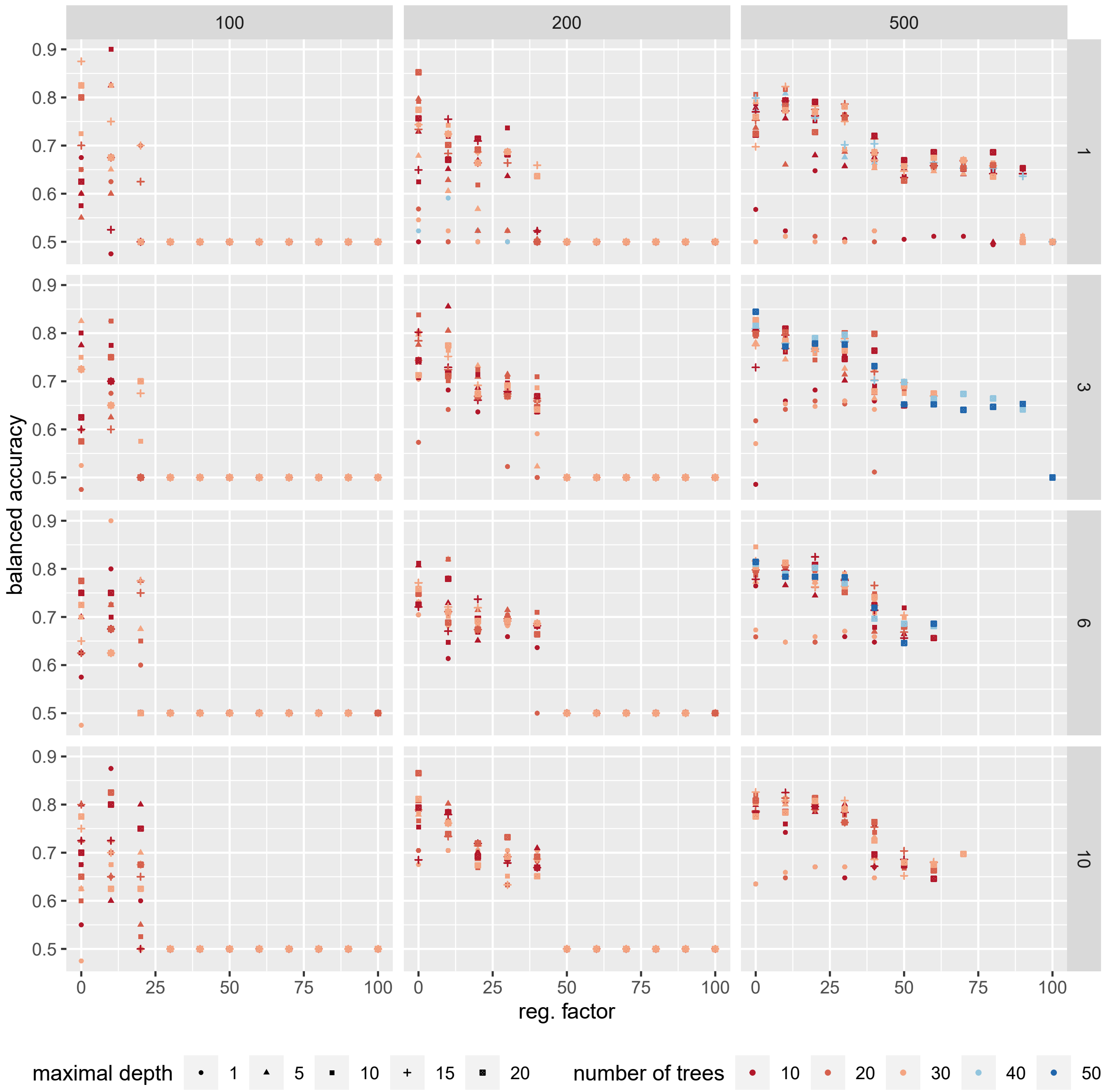


Figure 19. Balanced accuracy evaluation for synthetic data (blue setting) with varying sample size, $max\_features$, maximum depth (point shape) and number of trees (color).

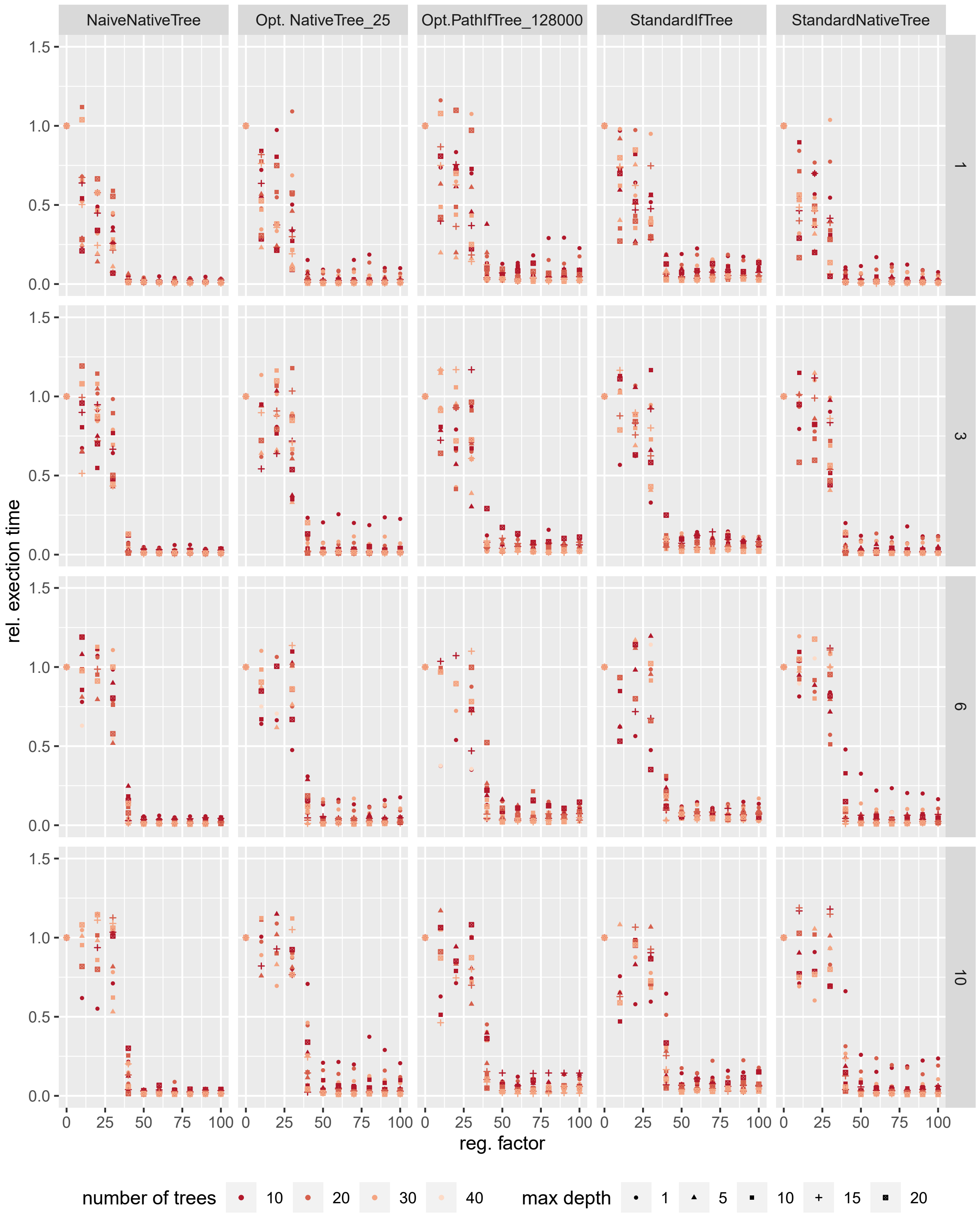


Figure 20. Relative execution time evaluation for synthetic data (red setting) and sample size=100 with varying sample size, $max\_features$, maximum depth (point shape) and number of trees (color).

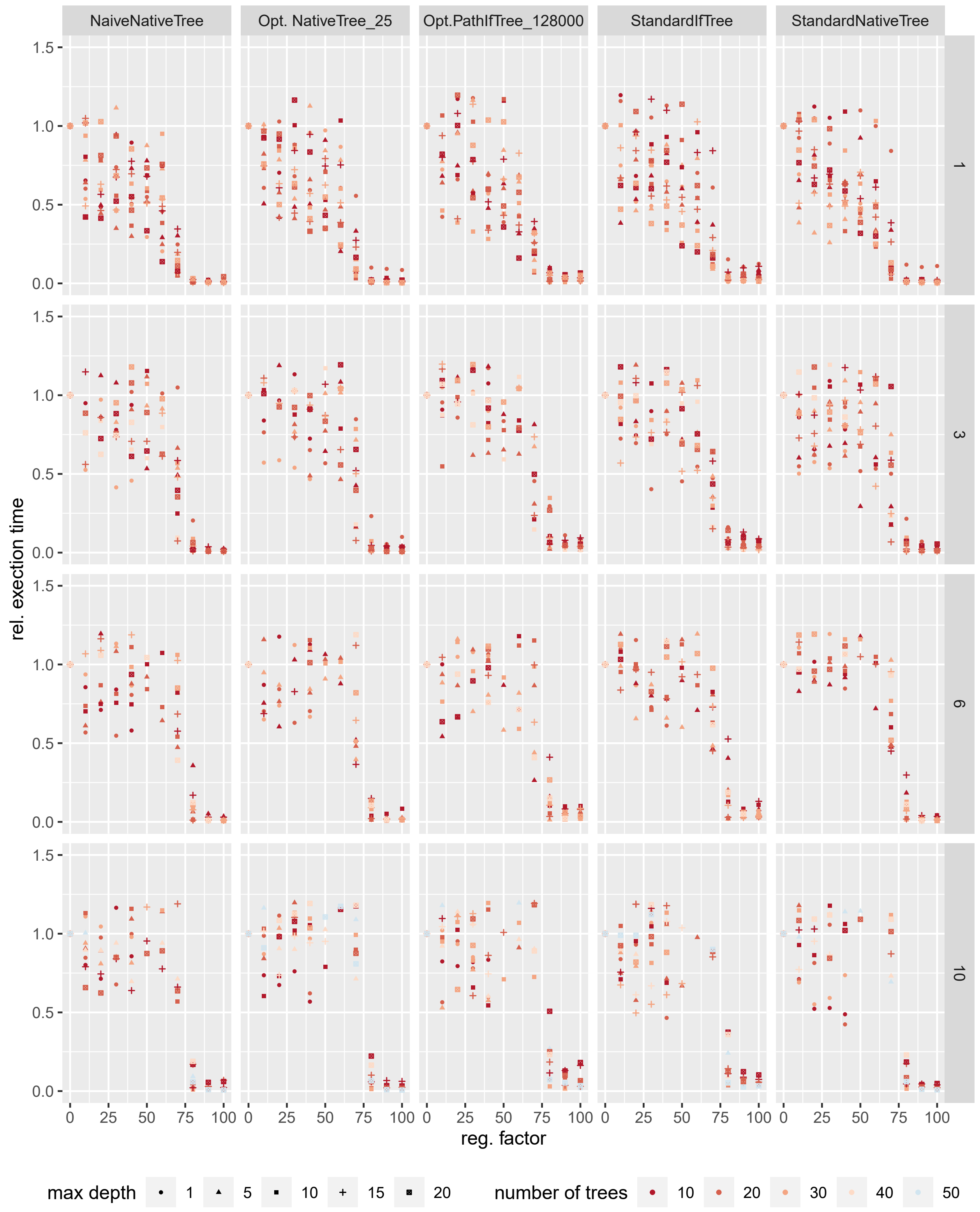


Figure 21. Relative execution time evaluation for synthetic data (red setting) and sample size=200 with varying sample size, $max\_features$, maximum depth (point shape) and number of trees (color).

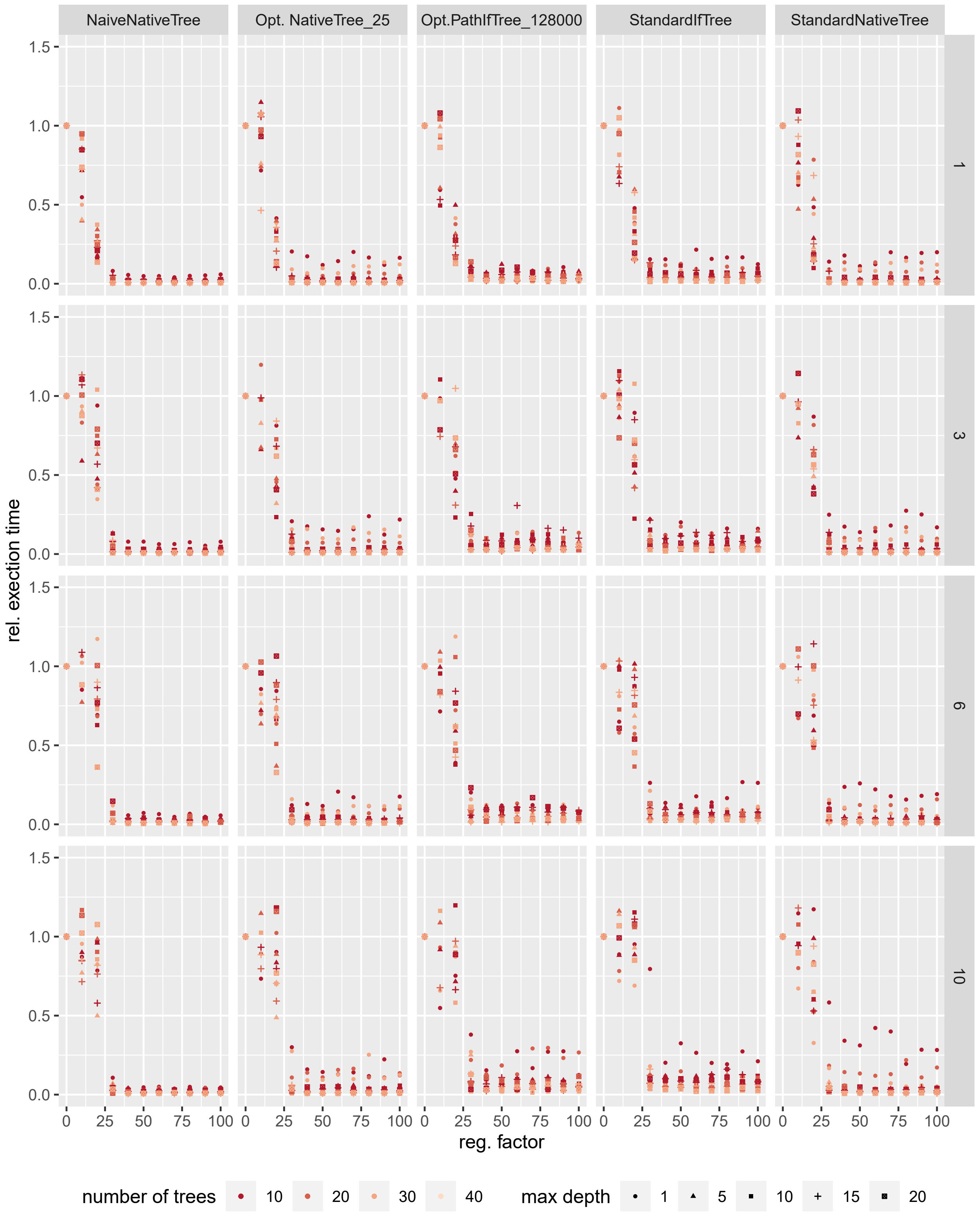


Figure 22. Relative execution time evaluation for synthetic data (green setting) and sample size=100 with varying sample size, $max\_features$, maximum depth (point shape) and number of trees (color).

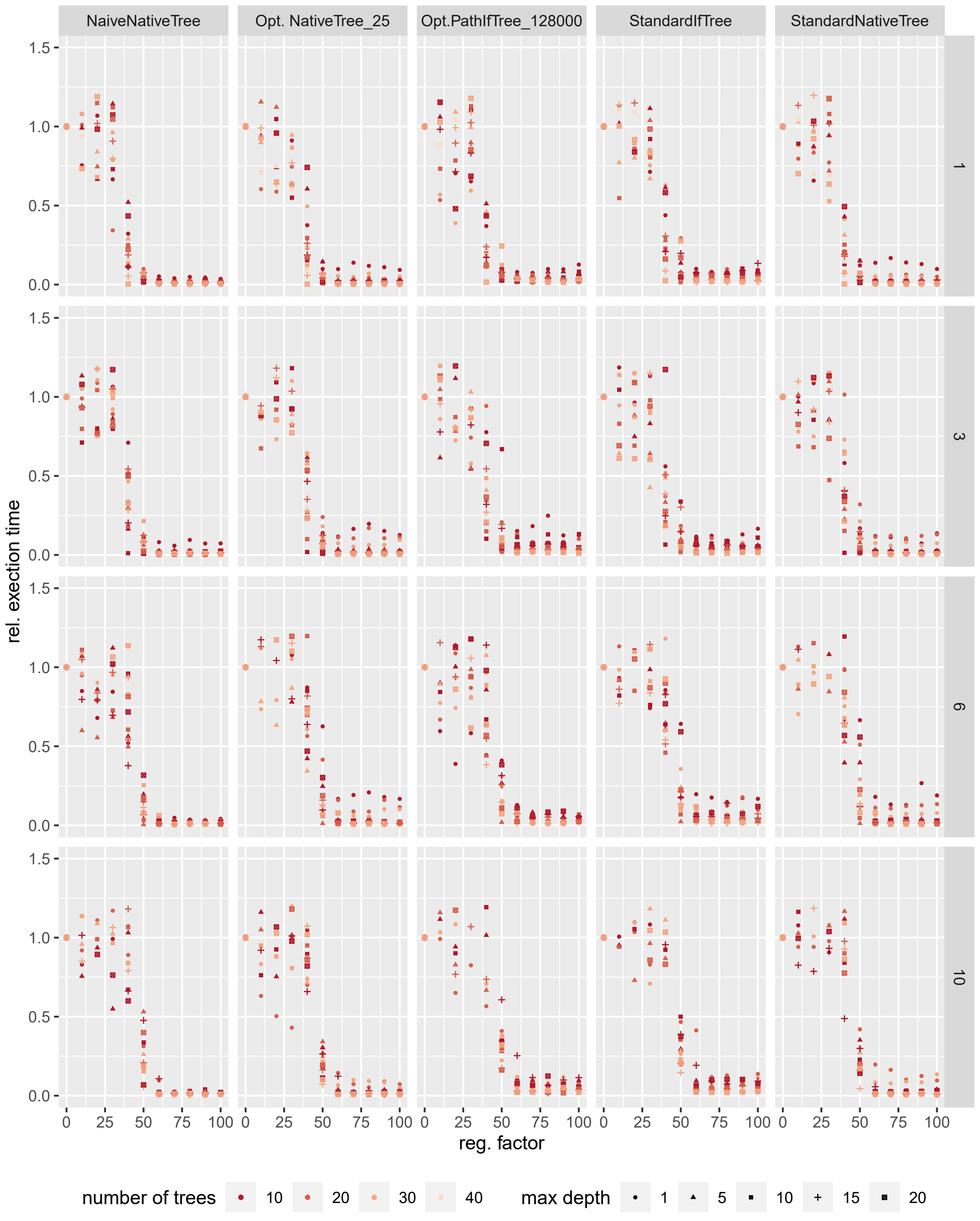


Figure 23. Relative execution time evaluation for synthetic data (green setting) and sample size=200 with varying sample size, $max\_features$, maximum depth (point shape) and number of trees (color).

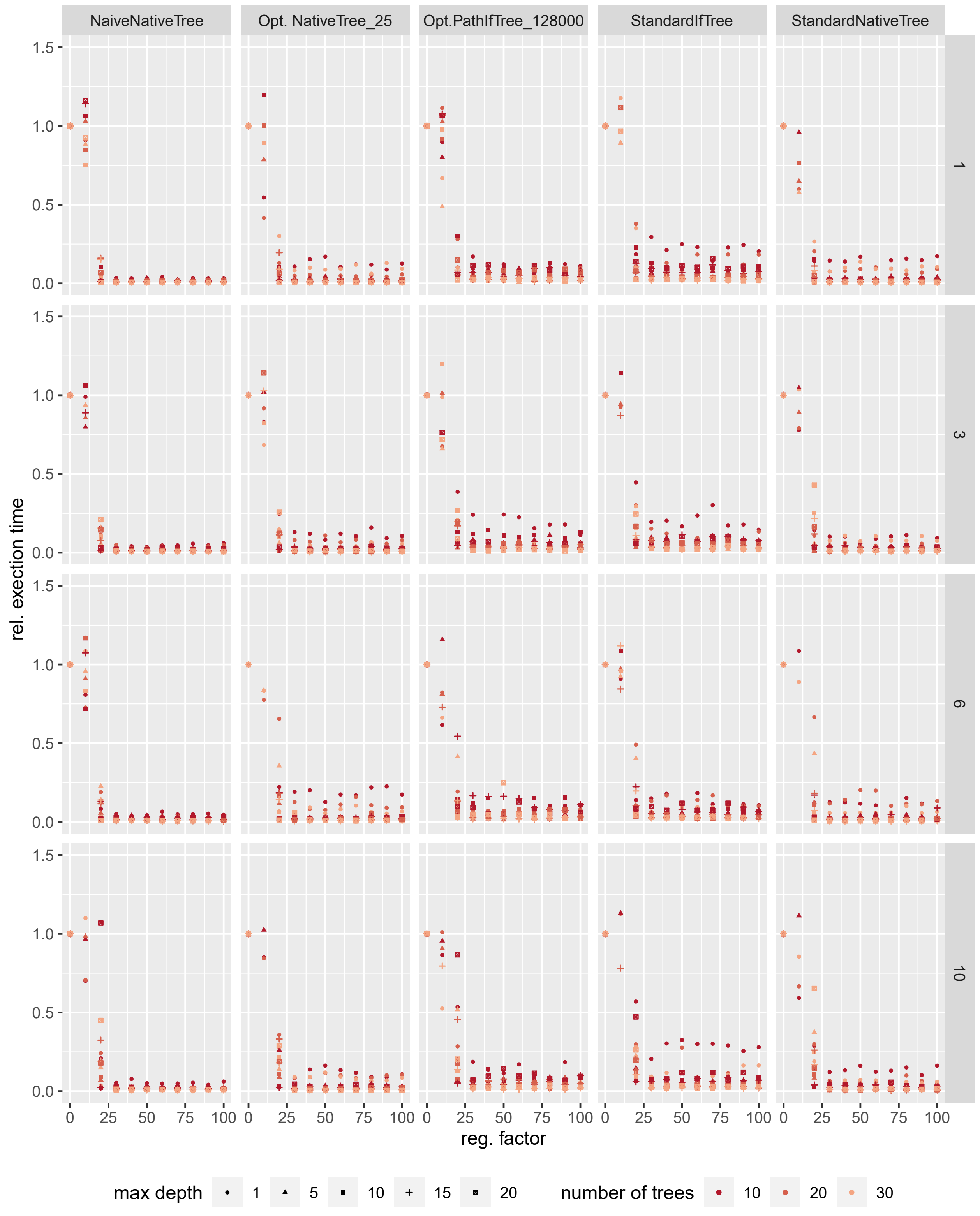


Figure 24. Relative execution time evaluation for synthetic data (blue setting) and sample size=100 with varying sample size, $max\_features$, maximum depth (point shape) and number of trees (color).

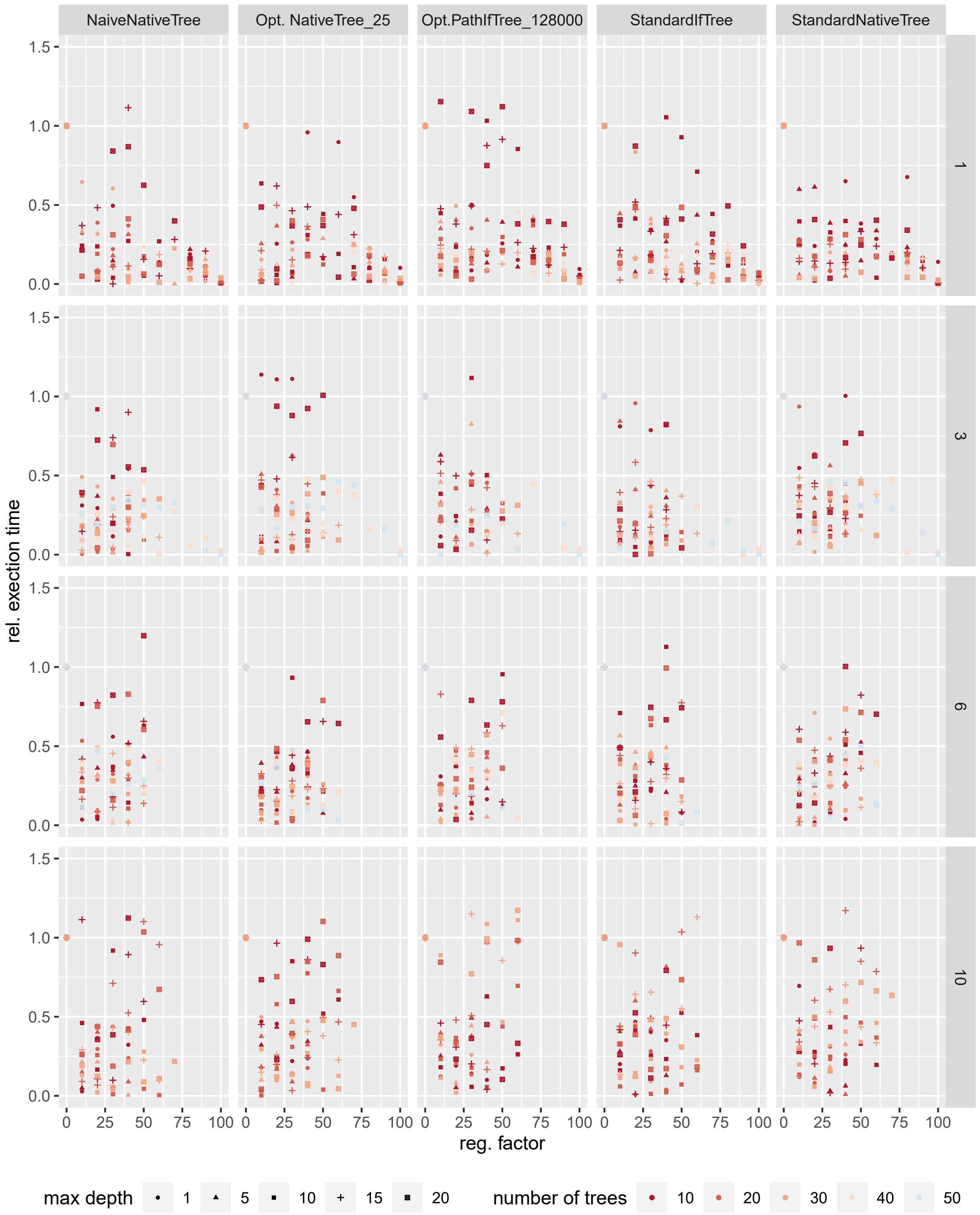


Figure 25. Relative execution time evaluation for synthetic data (blue setting) and sample size=200 with varying sample size, $max\_features$, maximum depth (point shape) and number of trees (color).

Name: Lena Schmid

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Dissertation mit dem Titel

"Statistical Analyses of Tree-Based Ensembles"

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe und die Satzung der Technischen Universität Dortmund zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet habe. Ich versichere außerdem, dass ich die beigefügte Dissertation nur in diesem und keinem anderen Promotionsverfahren eingereicht habe und dass diesem Promotionsverfahren keine endgültig gescheiterten Promotionsverfahren vorausgegangen sind. Ferner erkläre ich, dass keine Aberkennung eines bereits erworbenen Doktorgrades vorliegt. Ich versichere an Eides statt, dass ich nach bestem Wissen die reine Wahrheit erklärt und nichts verschwiegen habe.

Dortmund, den ....

Lena Schmid