Received September 23, 2020, accepted October 5, 2020, date of publication October 19, 2020, date of current version October 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3032095

Generalizing Gain Penalization for Feature Selection in Tree-Based Models

BRUNA WUNDERVALD, ANDREW C. PARNELL, AND KATARINA DOMIJAN
Hamilton Institute, National University of Ireland Maynooth, Maynooth, W23 F2H8 Ireland

Corresponding author: Bruna Wundervald (bruna.wundervald@mu.ie)

This work was supported by a Science Foundation Ireland Career Development Award grant number: 17/CDA/4695.

ABSTRACT We develop a new approach for feature selection via gain penalization in tree-based models. First, we show that previous methods do not perform sufficient regularization and often exhibit sub-optimal out-of-sample performance, especially when correlated features are present. Instead, we develop a new gain penalization idea that exhibits a general local-global regularization for tree-based models. The new method allows for full flexibility in the choice of feature-specific importance weights, while also applying a global penalization. We validate our method on both simulated and real data, exploring how the hyperparameters interact and we provide the implementation as an extension of the popular `R` package `ranger`.

INDEX TERMS Dimensionality reduction, feature selection, gain penalization, tree-models.

I. INTRODUCTION

In many Machine Learning problems, features can be hard or economically expensive to obtain, and some may be irrelevant or poorly linked to the target. For these reasons, reducing the number of features is an important task when building a model, and benefits the data visualization and model performance, whilst reducing storage and training time requirements [1]. However, for tree-based methods, there is no standard procedure for feature selection or regularization in the literature, as one would find for Linear Regression and the LASSO [2] for example. Performing feature selection in trees can be difficult, as they struggle to detect highly correlated features and their feature importance measures are not fully trustworthy [3]. Several methods to tackle this problem have been recently proposed, including [4], [5], and [6].

In [5], the authors treat trees as parametric models and use procedures analogous to LASSO-type shrinkage methods, by penalizing the coefficients of the base learners and reducing the redundancy in each path from the root node to a leaf node. However, their selected features can still be redundant, since the focus is on reducing the number of rules instead of the number of features.

On a different approach we have [4], which focuses on gene selection specifically for classification methods. The authors propose an iterative tool that eliminates the least important features (in fractions of the number of features, p) and updates the algorithm at each iteration. The complication is that the method will always be either computationally expensive, if p is low, or will eliminate too many features at once, which can exclude useful or interaction features. Besides, the method does not generalize to other dataset contexts or tasks, such as regression.

In the contrasting approach of [6] and [7], the authors regularize Random Forests by gain penalization. Their method consists of letting the features only be picked by a Random Forest if their penalized (weighted) gain is still high. They make recommendations on how to set the penalization coefficients and present their implementation in the `RRF` package for `R` [8]. However, the authors give no further guidelines on how to generalize their method for other tree-based models and penalization types and do not explore the influence of hyperparameters on the algorithm.

Given that, in this work, we develop a gain penalization approach that is fully generalizable and widely applicable, in opposition to those mentioned above. In particular, our main contributions are:

- We provide a general gain penalization procedure for tree-based models, which allows for a combination of local and global regularization parameters.
- We allow for the bespoke local regularization functions to be domain-specific, which introduces a prior-like component to feature selection in tree-based models.
- We propose different techniques for setting the regularization parameters and discuss how they affect the final results, with real and simulated examples.

The associate editor coordinating the review of this manuscript and approving it for publication was Sotirios Goudos.

- We generalize gain penalization to multiple tree-based methods (CART, Bagging, Random Forests), for both regression and classification.
- We make available a faster implementation of the gain penalization method, included in the very widely used `ranger` package.

The format of this article is structured as follows. Section 2 explains the problem setup, followed by the generalization of gain penalization in Section 3. In Section 4, we present the results for simulated and real data. Section 5 explains the implementation details, and Section 6 has the conclusions and future work.

II. PROBLEM SETUP

Consider a set of training target-feature pairs $(Y_i, \mathbf{x}_i) \in \mathbb{R} \times \mathbb{R}^p$, with $i = 1, \ldots, N$ indexing the observations with p being the total number of features. In general, we can estimate an $\hat{f}$ that describes how the features $\mathbf{x}_i$ relate to Y_i and use it for prediction or inference. However, not all features need to be involved in $\hat{f}$. Especially for tree-based models, the occurrence of noisy or correlated features can badly influence the results [9]. Given that, our interest here relies on estimating $\hat{f}$ such that it will only use the matrix $\mathbf{x}_A$, composed by the sub-vectors of $\mathbf{x} \in \mathbb{R}^p$ indexed by A, $A \subset \{1, \ldots p\}$, which should contain the optimal set of features (it produces similar or equal prediction errors as the full set of features), that is potentially of a much smaller dimension.

A. TREES

Non-linear models have been extensively used for regression and classification problems. Trees are a particular case of such models, that recursively partition the feature space, resulting in a local model for each estimated region [10]. They learn the features directly from the training data, creating an adaptive basis function model (ABM) [11] of the form

$$f(\mathbf{x}) = \mathbb{E}[y \mid \mathbf{x}] = \sum_{r=1}^{R} w_r \mathbb{I}(\mathbf{x} \in R_r) = \sum_{r=1}^{R} w_r \phi_r(\mathbf{x}; \mathbf{v}_r), \quad (1)$$

where R_r is the r-th region, w_r is the prediction given to this region and $\mathbf{v}_r$ represents the splitting feature chosen and the corresponding splitting value. These algorithms are fitted using a greedy procedure, that computes a locally optimal maximum likelihood estimator by finding the splits that lead to the minimization of a cost function. For regression, the cost function of a decision $\mathbb{D}$ is frequently defined as $cost(\mathbb{D}) = \sum_{i \in \mathbb{D}} (y_i - \bar{y})^2$, where $\bar{y} = (\sum_{i \in \mathbb{D}} y_i)|\mathbb{D}|^{-1}$ is the mean of the observations in the specified region, while for classification this function is replaced by the misclassification rate, or $cost(\mathbb{D}) = |\mathbb{D}|^{-1} \sum_{i \in \mathbb{D}} \mathbb{I}(y_i \neq \hat{y})$.

Since trees are in general considered non-probabilistic algorithms, one way of measuring the importance of each feature is to calculate and aggregate their split gains. The gain of a new split is a normalized measure of the cost reduction, given by

$$\Delta(i, t) = cost(\mathbb{D}) - \Big(\frac{|\mathbb{D}_{LN_{(i,t)}}|}{|\mathbb{D}|} cost(\mathbb{D}_{LN_{(i,t)}}) + \frac{|\mathbb{D}_{RN_{(i,t)}}|}{|\mathbb{D}|} cost(\mathbb{D}_{RN_{(i,t)}}) \Big), \quad (2)$$

for feature i at splitting point t, while $\mathbb{D}$ is related to the previous estimated split, LN = (left candidate node) and RN = (right candidate node). The global importance value is given by accumulating the gain over a feature, $\Delta(i) = \sum_{t \in \mathbb{S}_i} \Delta(i, t)$, where $\mathbb{S}_i$ now represents all the splitting points used in a tree for the $i-$th feature. This measure will be a key component in our developments below.

B. TREE ENSEMBLES

Trees are known to be high variance estimators: small changes in the data can lead to the estimation of a completely different tree [11]. One way to increase stability is to use the property that an average of many estimates has a smaller variance than one estimate, and grow many trees from re-samples of the data. Averaging such results give us a bagged ensemble [12] of the form

$$\hat{f}(\mathbf{x}) = \sum_{n=1}^{N_{tree}} \frac{1}{N_{tree}} \hat{f}_n(\mathbf{x}), \quad (3)$$

where $\hat{f}_n$ corresponds to the n-th tree. The Random Forest [13] algorithm, for example, is defined by allowing only a random subset m of the features to be the candidates in each split. As for the importance values, in tree ensembles the feature importances get averaged over all the trees, or

$$Imp_i = \frac{1}{N_{tree}} \sum_{n=1}^{N_{tree}} \Delta(i)_n, \quad (4)$$

where i represents the feature index. Moreover, the prediction performance of the trees in ensembles such as the Random Forest relies on the number of features tried at each split, called `mtry` here, as when `mtry` $\to 1$, the larger the variance of each tree, but the more effective will be the averaging process, and vice versa [3]. As we will see later, the gain penalization procedure also depends on `mtry`, because when the wrong value is set for this hyperparameter, the penalization can lead to models that are very far from the truth.

C. REGULARIZATION BY GAIN PENALIZATION

In [6], the authors first discuss the regularization of Random Forests by gain penalization. The *Regularized Random Forest* (*RRF*) proposes weighting the gains of the splits during the greedy procedure, guiding the feature choosing of the model. The regularized gain is defined as

$$\text{Gain}_R(\mathbf{X}_i, t) = \begin{cases} \lambda\Delta(i, t), & i \notin \mathbb{U} \text{ and} \\ \Delta(i, t), & i \in \mathbb{U}, \end{cases} \quad (5)$$

where $\mathbb{U}$ is the set of indices of the features previously used, $\mathbf{X}_i$ is the candidate feature, and t the candidate splitting point. The $\lambda \in (0, 1]$ hyperparameter is the penalty coefficient that controls the amount of regularization each feature receives. A feature is penalized if it is new to the whole ensemble, as the method has a memory of which features were already used. Naturally, λ can be a constant value for all the features but ideally, there should be a regularization parameter for each feature that best represents the information they carry about the target. In [7], the authors modify this idea by introducing the *Guided RRF*. It consists of first running a Standard Random Forest (`mtry` $\approx \sqrt{p}$, number of trees = 500) and producing an importance measure for each feature and scaling this measure, in order to find $Imp'_i = \frac{Imp_i}{max_{j=1}^{P} Imp_j}$ where Imp_i is the importance measure calculated for the i-th feature in the Random Forest. The Imp'_i becomes a component of the penalization factor, in the form $\lambda_i = (1-\gamma) + \gamma Imp'_i$. However, this method is explicitly developed for Random Forests, as the gain penalization itself depends on the results of a previously run Random Forest. No other methodologies or extensions are explored, and the influence of the Random Forests hyperparameters are not studied by the authors.

III. GENERALIZING GAIN PENALIZATION

One of our goals with this work is to propose a generalization of gain penalization in tree-based models for feature selection. This generalization is made in two senses: for the penalization methodology and in the algorithm type. For the algorithm, this means that the regularization method can be applied to any tree-based algorithm, be it either single trees such as CART, or elaborated ensembles like Bagging and Random Forests. All these methods are available in our implementation. As for the penalization method, we generalize it by proposing a gain penalization based on a λ_i parameter, that is written as

$$\lambda_i = (1-\gamma)\lambda_0 + \gamma g(\mathbf{x}_i), \tag{6}$$

where $\lambda_0 \in [0, 1)$ is interpreted as the baseline regularization, $g(\mathbf{x}_i)$ is a function of the i-th feature, and $\gamma \in [0, 1)$ is their mixture parameter, with $\lambda_i \in [0, 1)$. In this fashion, we propose a local-global form of penalization, that is applied to all the features used in the model. The equation balances how much all features should be jointly, or globally, penalized and how much will it be due to a local $g(\mathbf{x}_i)$, that is manually set. When $\gamma = 0$, only a global penalization is performed, while when $\gamma = 1$, the regularization is fully controlled by $g(\mathbf{x}_i)$.

The $g(\mathbf{x}_i)$ should represent relevant information about the features, based on some characteristic of interest. It can include, for example, external information about the relationship between $\mathbf{x}_i$ and $\mathbf{y}$, or information only about $\mathbf{x}_i$ and its utility for the model. This formulation has inspiration on the use of priors made in Bayesian methods since we intend to introduce prior knowledge regarding the importance of each feature into the model. In the same way, the data will tell us how strong our assumptions about the penalization are, since even if we try to penalize a truly important feature, its gain will be high enough to overcome the penalization and the feature will get selected by the algorithm.

A. CHOOSING $g(\mathbf{x}_i)$

In the following, we list a few $g(\mathbf{x}_i)$ options, taking into consideration the possible natures of the features.

1) CORRELATION

A familiar option for continuous features is just to use $g(\mathbf{x}_i)$ as the absolute value of the marginal correlation between $\mathbf{x}_i$ and $\mathbf{y}$, when we assume a continuous target problem. It could be either Pearson's, Kendall's, Spearman's, or any other correlation coefficient of preference (the first is more suitable for ordinary numeric inputs, while the others will be more convenient for ordered inputs [14]). We drop the *sign* because when two features are correlated, the magnitude of the coefficient is enough to define its importance in terms of significance, and one can use simply

$$g(\mathbf{x}_i) = |corr(\mathbf{y}, \mathbf{x}_i)|. \tag{7}$$

2) ENTROPY AND MUTUAL INFORMATION

A different situation is when the features are discrete. In information theory, Shannon's entropy [15] is a measure of the uncertainty of a (discrete) random feature. In a short description, if a discrete feature X has K states, its entropy will be calculated as

$$\mathbb{H}(X) = -\sum_{k=1}^{K} p(X=k)\log_2 p(X=k), \tag{8}$$

where $\mathbb{H}(X) \in [0, \infty]$. Higher entropy will mean more uncertainty, so it can be reasonable to give more weight to features with lower uncertainties. One can use a normalized version of the entropy calculated for each $\mathbf{x}_i$, or

$$g(\mathbf{x}_i) = 1 - \frac{\mathbb{H}(\mathbf{x}_i)}{max_{j=1}^{P}\mathbb{H}(\mathbf{x}_j)}, \tag{9}$$

compelling the features with lower entropy to have larger penalization coefficients. Under the same framework, a more general approach to quantify how much knowing about one feature tells us about the other is the Mutual Information function. In this case, the similarity between a joint distribution $p(X, Y)$ and a factored distribution $p(X)p(Y)$ is calculated with

$$\text{MutInf}(X; Y) = \sum_{x}\sum_{y} p(x, y)\log\frac{p(x, y)}{p(x)p(y)} \tag{10}$$

for two features X and Y, which is basically the Kullback-Leibler divergence between the two distributions [11]. Recalling Equation 8, it is easy to see that the Mutual Information value is the reduction in the uncertainty about Y when we observe X, so it can be straightforwardly used as

$$g(\mathbf{x}_i) = \frac{\text{MutInf}(\mathbf{x}_i, \mathbf{y})}{max_{j=1}^{P}\text{MutInf}(\mathbf{x}_j, \mathbf{y})}. \tag{11}$$

3) BOOSTED

If there is no interest in differentiating continuous or discrete features, one can use what we call a *Boosted* $g(\mathbf{x}_i)$. Such functions depend on previously run machine learning models that provide an importance value for the features. The term *Boosted* is to introduce some familiarity with what the algorithm consists of since we can arguably see it as a heterogenous Boosting [16] applied to the features instead of the observations.

More generally, many Machine Learning algorithms can be used whenever they allow for the calculation of an importance value. Some examples include: Generalized Linear Models [17], where e.g. the normalized absolute parameter coefficients can be interpreted as importance values, and Support Vector Machines [18] that produce importance values via sensitivity analysis ([19], [20]). Each family of algorithms will have its specific characteristics and preferences towards the features, so it is advisable to be aware of those details when using it in a *Boosted* $g(\mathbf{x}_i)$.

4) COMBINATION

Another possibility is combining two or more $g(\mathbf{x}_i)$. Objectively speaking, some functions will be more appropriate to one type of feature than others. As an example, one could combine a *Boosted* method with the marginal correlations between the target and each feature. This formulation can, for example, be written as

$$g(\mathbf{x}_i) = \begin{cases} |corr(X_i, y)|, & if\,|corr(X_i, y)| > \epsilon \\ (Imp'_i, & if\,|corr(X_i, y)| \le \epsilon \end{cases}, \tag{12}$$

where we let the absolute values of the correlations compose $g(\mathbf{x}_i)$ if the correlation is over a certain threshold ϵ, and use Imp'_i from a previously run algorithm. Analogously, we might want to explore the features that are not so correlated to the target (e.g. when their relationship is very non-linear) by using a high value for ϵ.

B. DEPTH PARAMETER

Sometimes, growing very bushy trees with new features is not desirable when we want to use the smallest set of features possible. Following [21], where the authors use prior distributions for whether a new feature should be picked in a Bayesian Regression Tree, we introduce the idea of increasing a penalization given the current depth of the tree. Their priors take into account the current depth of a tree, so when a tree is already deep the priors get less concentrated in high probability regions, resulting in lesser bushier trees. In our framework, a similar idea is applied by setting

$$Gain_R(\mathbf{X}_i, t, \mathbb{T}) = \begin{cases} \lambda_i^{d_{\mathbb{T}}} \Delta(i, t), & i \notin \mathbb{U} \text{ and} \\ \Delta(i, t), & i \in \mathbb{U}, \end{cases} \tag{13}$$

where $d_{\mathbb{T}}$ is the current depth of the $\mathbb{T}$ tree, $\mathbb{T} = (1, \ldots, \texttt{ntree})$, for the i-th feature. The aim here is to reduce the gains of the features if they are to be picked in a deep node, preventing new features to appear at the bottom of trees unless their gains are exceptionally high. The benefit of this comes from the fact that deep nodes contain fewer observations than their parents, so a deep split will likely lead to a smaller gain if any at all. In a scenario where we want to keep only the variables that have a high importance to the model, this is undesirable and can be prevented by using our method added with the depth penalization.

C. DETAILS AND ADVANTAGES

1) FEATURE MASKING EFFECT

Tree-based models often suffer from feature masking effects [3]. For example, in a tree, some feature X_j might never occur in the algorithm if it leads to splits slightly worse than some other feature X_i. So if X_i is removed, X_j can prominently occur and have a high importance value. In theory, this problem is overcome by ensembles like Random Forests, as selecting only m features to pick from decorrelates the trees, but if we regularize Random Forests, the problem remains. This happens because if weak features end up being picked (randomly) by the trees, their gains will have an unfair advantage against the other features (possibly very important features), that will be penalized. Luckily this situation is can be fixed with hyperparameter tuning for `mtry`.

2) CORRELATED FEATURES

A second issue to have in mind when working with tree ensembles is their bias towards giving high importance to correlated features [9]. As an example, suppose we have a subset $\mathbf{C} \subseteq \mathbf{X}$ of features which are correlated. Ideally, we would expect to have only one or just a few of these features being selected, because if one of the correlated features is truly important for prediction, using this one feature is enough, but the ensembles are not able to detect and eliminate correlated features. The naive approach to tackle this problem is to calculate a full correlation matrix between all the features and filter by only the least correlated one, but when p grows this might not be computationally feasible, and it implies more manual work when building the model. Our gain penalization method automatically deals with the correlated features, since when one of the features in $\mathbf{C}$ gets picked, the algorithm is less likely to pick the other correlated features as well, given that a new feature needs to reduce the prediction error more drastically to be selected.

IV. EXPERIMENTS

This section shows the results of our experiments, that evaluated the effects of different regularization types in simulated and real datasets using the Random Forest algorithm.

A. SIMULATED DATA

Consider now a set $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{205})$ of features, all sampled from a Uniform[0, 1] distribution, $n = 1000$.

We generated a target of interest $\mathbf{Y} \in \mathbb{R}$ as

$$\mathbf{y} = 0.8\, sin(\mathbf{x}_1\mathbf{x}_2) + 2(\mathbf{x}_3 - 0.5)^2 + 1\mathbf{x}_4 + 0.7\mathbf{x}_5 + \sum_{j=1}^{200} 0.9^{(j/3)}\mathbf{x}_{j+5} + \sum_{j=1}^{45} 0.9^{j}\mathbf{x}_5 + \epsilon,\ \epsilon \sim N(0,1), \quad (14)$$

inspired by the simulation equation proposed in [22], totaling 250 features. This framework produces interesting relationships between the target and the features: non-linearities ($i = (1, 2, 3)$), decreasing importances ($i = (6, \dots, 205)$) and correlations ($i = (5, 206, \dots, 250)$), inducing a more complicated scenario. We created 10 datasets, all randomly split into train and test set (80%/20%). For all the algorithms we fixed the number of trees at 100, varied `mtry` $= (15, 45, 75, 105, 135, 165, 195, 225, 250)$ and our accuracy measure is the RMSE calculated in the test set. We used a standardized version of $\mathbf{y}$ and, in the following, the term *selected feature* represents any feature with importance $\Delta(i) > 0$ in the final estimated model.

B. STANDARD RANDOM FOREST, RRF AND GRRF

As a benchmark, we run a Standard Random Forest, the *RRF* and *GRRF* models for each of the the 10 simulated datasets and all the different values of `mtry`, and the results are compared to our method. The first `mtry` is what would be the default in a Standard RF, since $\sqrt{250} \approx 15$, and the last is the total of features available.

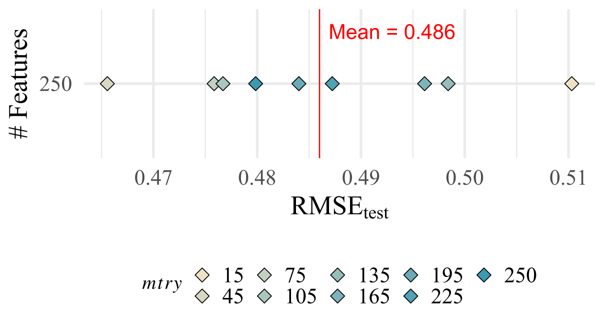

FIGURE 1. Averages of the number of features used and *RMSE$_{test}$* values for a Standard Random Forest. The models use all 250 features in every run. The lowest RMSE occurs with `mtry` = 45 and the highest RMSE with `mtry` = 15.

For the Random Forest model, the resulting number of features used for all the models is always the maximum available (see Figure 1). If we consider the correlated features issue, this means that too many features are being picked, once we know that they become irrelevant in their joint presence. The RMSE$_{\text{test}}$ changes when `mtry` changes: when `mtry` $= 45$ is when we have the best results, meaning that the default value (`mtry` $= \sqrt{p} \approx 15$) is not the best option. As for the *RRF* and *GRRF*, we used the hyperparameters $\lambda = (0.05, 0.12, 0.18, 0.25, 0.32, 0.39, 0.45, 0.52, 0.59, 0.65, 0.72, 0.79, 0.86, 0.92, 0.99)$, $\gamma = (0.05, 0.12, 0.18, 0.25, 0.32, 0.39, 0.45, 0.52, 0.59, 0.65, 0.72, 0.79, 0.86, 0.92, 0.99)$ (which represents the weighting parameter of Imp'_i in the *GRRF*) and tried all combinations between λ (*RRF*), γ (*GRRF*) and `mtry` (both). The models were run using the `RRF` [1] [23] package for `R` [8].

Figure 2 shows the results of the average RMSE$_{\text{test}}$ (left) and average number of selected features (right) in the 10 datasets for the two types of models. We can see a continuous transition in the number of features picked by the two models, but they present an inverse pattern regarding the `mtry` and penalization parameters. For the *RRF*, the region with the lowest RMSE$_{\text{test}}$ is predominantly the one with the most features, meaning that the penalization is happening but the models using the least features do not satisfactorily predict for the test set. As for the *GRRF*, `mtry` = 45 seems to be optimal, similarly to the Random Forest. However, the lowest RMSE$_{\text{test}}$ region happen when γ is high, which represents and improvement but still seemingly leads to the number of selected features to not be as low as it can be.

C. GENERALIZED GAIN PENALIZATION IN RANDOM FORESTS

Now we present the experiment results using the Generalized Gain Penalization in Random Forests. For this subsection, we vary $\lambda_0 = (0.1, 0.5, 0.9)$ and $\gamma = (0.001, 0.25, 0.5, 0.75, 0.99)$, use all combinations of the hyperparameters ($\gamma \times \lambda_0$), first with $g(\mathbf{x}_i) = |corr(\mathbf{y}, \mathbf{x}_i)|$ and later using two *Boosted* methods, with a Standard Random Forest and with a Support Vector Machine. In Figure 3 we can see that RMSE$_{\text{test}}$ values are mostly close or below the 0.5 line. In comparison to Figure 2, our algorithm is doing better, as we can spot many cases where the RMSE$_{\text{test}}$ is low while using very few features (<25), especially when $g(\mathbf{x}_i) = Boosted_{SVM}$. Even when the RMSE$_{\text{test}}$ is a little higher, 0.5 for example, the number of features used is frequently much smaller (<10) than previously seem in the other algorithms. When γ is low the regularization is primarily controlled by λ_0, and we spot a heavier influence of `mtry` on the number of selected features, which tends to decrease as λ_0 increases. When γ is high the penalization values depend more on $g(\mathbf{x}_i)$, and the results vary less regarding the values for λ_0 and `mtry`.

A more in-depth analysis of the selected features can be seen in Table 1. We define the most important features in the simulation as $\mathcal{V} = (\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}, \mathbf{x_5}, \underset{i \in [6,205] \cap [0.9^{(i-5)/3} > 0.01]}{\mathbf{x_i}})$. We do not include the last 45 features which are correlated and we ideally want to avoid them. We then calculate the percentage of important features that were selected by each algorithm, from the total of features, and for the correlated ones, which percentage of those was selected by the algorithm. So, for example, if an algorithm picked 10 features, 3 of them being important, 5 being from the correlated group

[1] There is a difference on the splitting criteria of the `RRF` package, that calculates $\Delta(i,t) = \left(\frac{cost(\mathbb{D}_{LN(i,t)})}{|\mathbb{D}_{LN(i,t)}|} + \frac{cost(\mathbb{D}_{RN(i,t)})}{|\mathbb{D}_{RN(i,t)}|}\right) - \frac{cost(\mathbb{D})}{|\mathbb{D}|}$, when finding the best feature and threshold to split on. At each time the cost reduces in $\frac{cost(\mathbb{D})}{|\mathbb{D}|}$, making the cost measure have a smaller magnitude than the one used by us in the `ranger` package.

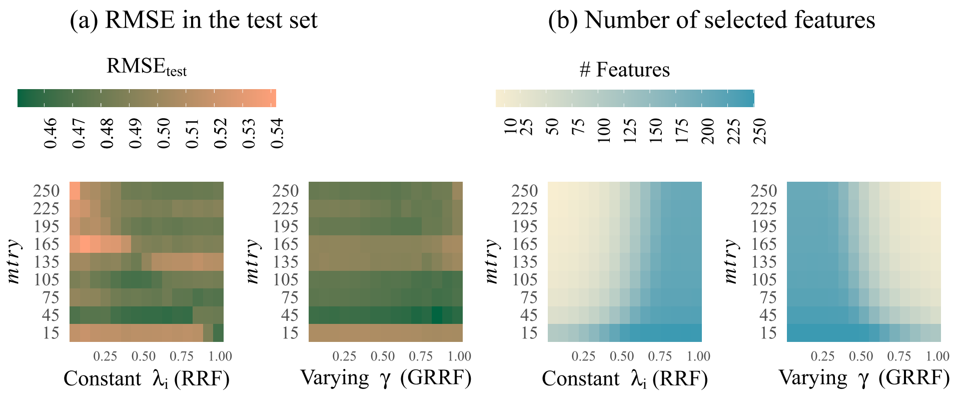

FIGURE 2. (a) Tile plot for the average of resulting $RMSE_{test}$ and (b) number of selected features in a *Regularized Random Forest* and a *Guided Regularized Random Forest* varying `mtry`, λ and γ. The number of selected features has a clear effect in the two models. For the *RRF*, the region with the lowest $RMSE_{test}$ is predominantly the one with the most features, while for the *GRRF* this situation improves.

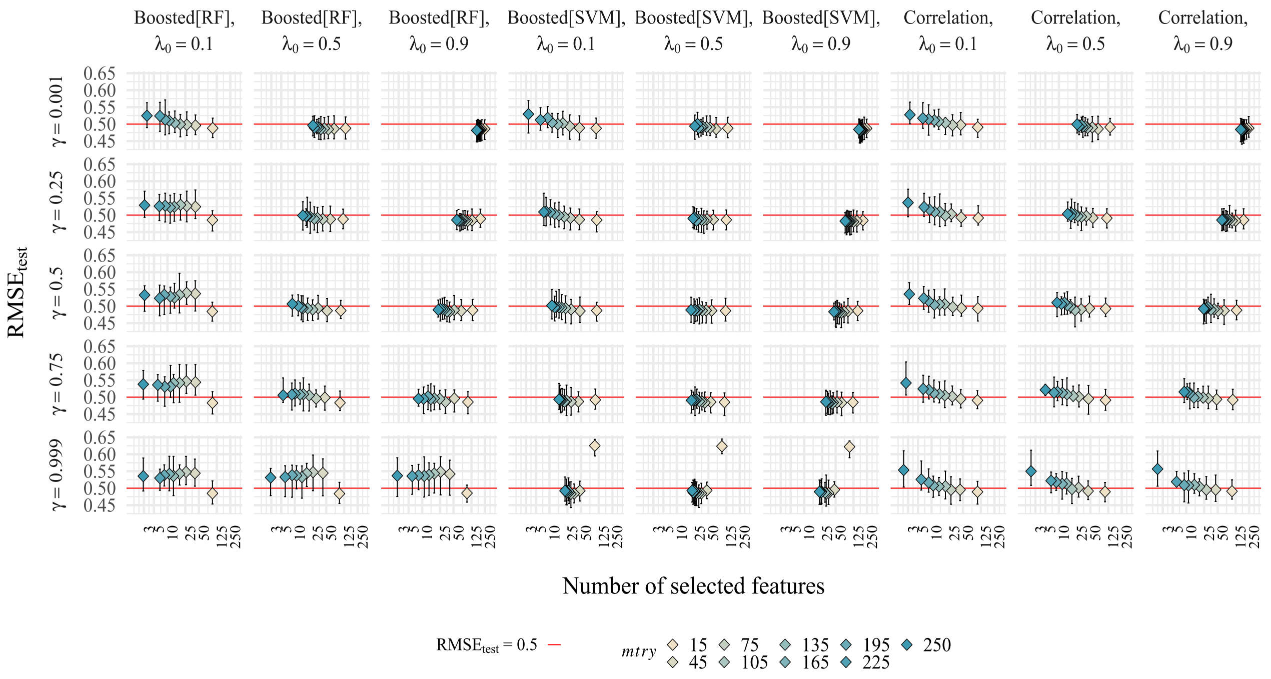

FIGURE 3. Averages of *$RMSE_{test}$* (with maximum and minimum intervals) and of the log number of features using the mixture of a λ_0 and a $g(\mathbf{x}_i)$, for $g(\mathbf{x}_i) = (|corr(\mathbf{y}, \mathbf{x}_i)|,\ Boosted_{RF},\ Boosted_{SVM})$. The x-axis shows the original scale, but the values are transformed to log. The models are mainly using fewer features than the *GRRF* or Standard RF, with λ_0 and `mtry` visibly affecting the results.

and 2 being ''non-important'', we calculate the proportion of important features as 3/5 and the proportion of correlated as 5/45.

With Table 1 we see that the proportion of important features is considerably higher for our approach with $g(\mathbf{x}_i) = Boosted_{RF}$ and $g(\mathbf{x}_i) = Boosted_{SVM}$. When we use $g(\mathbf{x}_i) = |corr(\mathbf{y}, \mathbf{x}_i)|$ the algorithm picks less of the correlated features. We also notice that the best results happened when $\lambda_0 \leq 0.5$ and $g(\mathbf{x}_i)$ has a higher influence in the penalization coefficients, so the introduction of prior information in the gain penalization is really helping the feature selection. We also show the same results for the *GRRF*, which picks many more of the correlated features and, on one occasion, more of the important ones. When looking

TABLE 1. Percentages of the most important and of correlated features selected and $RMSE_{test}$, averaged by `mtry` and γ. When using $g(\mathbf{x}_i)$ = $Boosted_{SVM}$, we pick more of the important features, less of the correlated and have lower $RMSE_{test}$. The *GRRF* tends to pick many of the correlated features, leading to non-optimal feature subsets.

$g(\mathbf{x}_i)$	λ_0	% Imp.	% Corr.	$RMSE_{test}$
Correlation	0.10	65.2%	18.8%	0.51
Correlation	0.50	64.6%	19.0%	0.50
Correlation	0.90	64.6%	20.0%	0.49
$Boosted_{RF}$	0.10	69%	33.2%	0.52
$Boosted_{RF}$	0.50	71.2%	28.2%	0.50
$Boosted_{RF}$	0.90	67.6%	28.0%	0.50
$Boosted_{SVM}$	0.10	71%	21.6%	0.50
$Boosted_{SVM}$	0.50	69.8%	20.8%	0.49
$Boosted_{SVM}$	0.90	67.6%	22.4%	0.48

GRRF

γ	% Imp.	% Corr.	$RMSE_{test}$
≈ 0.10	63.8%	33.1%	0.48
≈ 0.50	67.6%	32.6%	0.48
≈ 0.90	82.8%	32.0%	0.48

TABLE 2. Real classification datasets and its specifications. Problematic $p > n$ situation in all cases.

Dataset	Ref.	Obs.	Features	Classes
adenocarcinoma	[24]	76	9869	2
brain	[25]	42	5598	5
breast 2	[26]	77	4870	2
breast 3	[26]	95	4870	3
colon	[27]	62	2001	2
leukemia	[28]	38	3052	2
lymphoma	[29]	62	4027	3
nci 60	[30]	61	5245	8
prostate	[31]	102	6034	2
srbct	[32]	63	2309	4

at this result, we need to take into account that this model also tends to select more features in general, so that is not satisfactory if too many variables were selected.

D. REAL DATA CLASSIFICATION

This part of our work now discusses the results for real gene classification micro-array datasets, specified in detail in Table 2. With an average of 4787 features, 67 observations, and 3 classes, those are classical examples of "large p, small n" datasets. Though the focus here is in gene datasets, our method generalizes to data from any contexts or sizes and the datasets were chosen following previous literature ([4], [6]).

As the goal here is to find the best features to predict the gene classes, the experiment conducted for this section is different. We run the penalized models and extract their selected features, that are later used in a Standard Random Forest, with which the misclassification rates are calculated. This is to mimic how such an approach can be used in practice, where first a discovery experiment is run to identify important features, then a subsequent algorithm is run on a new dataset using the selected features to better assess their prediction power. We set $\gamma = \lambda_0 = 0.5$, attributing the same weight to the baseline regularization and to $g(\mathbf{x}_i)$. We vary `mtry` $= (\sqrt{p}, 0.15p, 0.40p, 0.75p, 0.95p)$ and $g(\mathbf{x}_i) = \left(\text{Boosted}_{RF}, \frac{\text{MutInf}(\mathbf{y},\mathbf{x}_i)}{max_{j=1}^{P}\text{MutInf}(\mathbf{y},\mathbf{x}_j)}\right)$. We also run a Standard Random Forest, a *GRRF*, the LASSO [2], and varSelRF [4] algorithms for each dataset, which are namely the biggest competitors of our method and are quite different feature selection techniques, bringing variety to our comparisons. All datasets were randomly separated into 50 different train (2/3) and test sets (1/3). We first find the average misclassification rates (MR) and the number of features used for each of the 50 resamples, eliminating at this step the `mtry` column. Out of that, we filter by the resample with the smallest misclassification rate.

According to Table 3, the Standard RF uses more features, but does not always have the lowest prediction errors. As for the generalized gain penalization, using $g(\mathbf{x}_i) = \left(\frac{\text{MutInf}(\mathbf{y},\mathbf{x}_i)}{max_{j=1}^{P}\text{MutInf}(\mathbf{y})}\right)$ is better for [brain] and [prostate], while when $g(\mathbf{x}_i) = \text{Boosted}_{RF}$, the results are good for the [adenocarcinoma], [breast 2], [breast 3] and [nci 60]. The *GRRF* is strictly better for the [colon] and [srbct] datasets considering the MR, though it uses many more features in comparison to the other algorithms. The LASSO often presents a low percentage of features, but its misclassification rates are much above the ones for the other models shown in the table. If we compare that to the generalized gain penalization method with $g(\mathbf{x}_i) = \left(\frac{\text{MutInf}(\mathbf{y},\mathbf{x}_i)}{max_{j=1}^{P}\text{MutInf}(\mathbf{y})}\right)$, our approach would be much more preferable since it uses less variables than the LASSO while mostly keeping a lower misclassification rate in the test set, The MRs are all the same for the [leukemia] and [lymphona] in the Standard RF, *GRRF* and generalized gain penalization models, but the percentage of features is often the lowest for our method. When this happens and such algorithms also have a low or very similar MR to a Standard RF one, we reach an optimal situation, which happened for almost all the datasets. Looking at the varSelRF results, we notice that this model produces the highest prediction errors and does not beat our models in terms of the percentage of features used, even though this algorithm is designed to work in well in this specific context.

V. IMPLEMENTATION

The implementation used here is included as an extension to the `ranger` package [33] for `R` [8]. This choice was made given that the `ranger`, originally written in `C++`, is the fastest Random Forest implementation available for `R`, so it serves us well for a general tree-based approach and the models can scale to high-dimensional settings. Furthermore, the package has a wide variety of other Random Forests extensions, is actively maintained and interfaces with `python`. The speed and scalability discussion presented in ranger and its comparison to the `randomForest` package [34] is analogous to the one about our regularization implemented in the `ranger` and the one in the `RRF` package

TABLE 3. Average percentage of features used and average misclassification rates with standard deviation for all the models. The gain penalized models used far fewer features than a Standard Random Forest, and it frequently uses fewer variables than the other feature selection techniques. Our approach also frequently has the lowest prediction errors, showing how it can use far fewer features whilst maintaining competitive misclassification performance.

Percentage of features used

Dataset	Std.RF	GRRF	Boosted$_{RF}$	Mut.Inf.	LASSO	varSelRF
adenocarcinoma	9.38	0.83	0.86	0.07	**0.02**	0.05
brain	25.06	1.44	1.46	**0.14**	0.39	0.73
breast 2	24.44	1.76	1.79	**0.14**	0.21	0.34
breast 3	38.58	1.95	2.00	**0.28**	0.67	0.28
colon	34.44	2.53	2.60	**0.44**	0.46	0.94
leukemia	8.07	1.26	1.25	**0.05**	0.28	0.09
lymphoma	13.88	1.07	1.14	**0.08**	0.34	0.72
nci	44.77	1.81	1.88	**0.16**	1.11	0.97
prostate	18.22	1.46	1.40	0.09	0.14	**0.07**
srbct	29.69	2.26	2.25	**0.3**	0.68	0.99

Misclassification rate in the test set

Dataset	Std.RF	GRRF	Boosted$_{RF}$	Mut.Inf.	LASSO	varSelRF
adenocarcinoma	3.45 (0.0)	11.03 (12)	**3.4** (0.0)	10.77 (1.7)	13.83 (6.4)	19.6 (7.7)
brain	8.00 (5.6)	15.38 (14.4)	15.00 (7)	**13.33 (10.5)**	27.6 (11.7)	29 (16.2)
breast 2	25.6 (5.4)	20.7 (2)	**17.8 (7.2)**	20.7 (7.7)	31.56 (5.19)	36.7 (9.1)
breast 3	27.6 (6.4)	30.6 (4)	**28.1 (3.8)**	29.3 (1.6)	29.7 (4.85)	33.9 (8.6)
colon	4.76 (0.0)	**5.71 (2.1)**	6.67 (0.0)	7.78 (3)	16.7 (8.6)	23.06 (8.3)
leukemia	0.0 (0.0)	**0.0 (0.0)**	**0.0 (0.0)**	**0.0 (0.0)**	10.07 (9.4)	13.2 (12)
lymphoma	0.0 (0.0)	**0.0 (0.0)**	**0.0 (0.0)**	**0.0 (0.0)**	1.48 (4.7)	5.86 (4.8)
nci	30.77 (9.4)	38.95 (8)	**37.5 (7.6)**	43.75 (0.0)	42.2 (7.5)	44.7 (12.8)
prostate	0.54 (1.2)	**0.54 (1.2)**	1.08 (1.5)	**0.54 (1.2)**	6.00 (2.8)	8.87 (1.9)
srbct	0.0 (0.0)	**0.0 (0.0)**	0.91 (2)	1.74 (3.9)	1.33 (2.9)	4.5 (3.6)

(which is based on the original `randomForest` code) so we will not repeat the same experiments.[2]

VI. CONCLUSION AND NEXT STEPS

Feature selection and regularization for tree-based methods is not an easy task and is a topic of active research. In this work, we have demonstrated that one efficient and general way of accomplishing such a task is via a generalization of feature gain penalization for tree-based methods. Our method combines previous information about the features with a baseline penalization λ_0, in a fully flexible local-global form of gain penalization. In general, the technique produces good results in terms of the number of features used and prediction error trade-off, outperforming more traditional methods. The performance and other characteristics of the method have been demonstrated with both simulated and real data in extensive experiments. Along with the methodology, we make the implementation available in the `ranger` package for `R`, which can also be used in `python`.

The downside of our approach is the addition of new hyperparameters, and how to choose them well. Future work involves finding theoretical properties of certain gain penalization approaches, parameter optimization (using e.g. [35]), and comparing our approach to other methods with a similar context ([36]–[38], for example). We would also like to explore extensions of this work to other popular and powerful tree-based models, such as gradient boosting algorithms ([39], [40]).

[2] *All the code and data used are available for reproducing the experiments.*

REFERENCES

[1] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Jan. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944968

[2] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Stat. Soc., B (Methodol.)*, vol. 58, no. 1, pp. 267–288, Jan. 1996.

[3] G. Louppe, "Understanding random forests: From theory to practice," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Fac. Appl. Sci., Univ. Liège, Liège, Belgium, Oct. 2014.

[4] R. Diaz-Uriarte, "GeneSrF and varSelRF: A web-based tool and R package for gene selection and classification using random forest," *BMC Bioinf.*, vol. 8, p. 328, Sep. 2007. [Online]. Available: http://www.biomedcentral.com/1471-2105/8/328

[5] J. H. Friedman and B. E. Popescu, "Predictive learning via rule ensembles," *Ann. Appl. Statist.*, vol. 2, no. 3, pp. 916–954, Sep. 2008.

[6] H. Deng and G. Runger, "Feature selection via regularized trees," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jun. 2012, pp. 1–8.

[7] H. Deng and G. Runger, "Gene selection with guided regularized random forest," *Pattern Recognit.*, vol. 46, no. 12, pp. 3483–3489, Dec. 2013.

[8] R. C. Team, *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing, 2018. [Online]. Available: https://www.R-project.org/

[9] C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis, "Conditional variable importance for random forests," *BMC Bioinf.*, vol. 9, no. 1, p. 307, Dec. 2008.

[10] L. Breiman, J. Friedman, C. Stone, and R. Olshen, *Classification and Regression Trees* (The Wadsworth and Brooks-Cole Statistics-Probability Series). Milton Park, U.K.: Taylor & Francis, 1984. [Online]. Available: https://books.google.ie/books?id=JwQx-WOmSyQC

[11] K. P. Murphy, "Machine learning: A probabilistic perspective," in *Adaptive Computation and Machine Learning Series*. Cambridge, MA, USA: MIT Press, 2012.

[12] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996, doi: 10.1023/A:1018054314350.

[13] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[14] P. Chen and P. Popovich, *Correlation: Parametric and Nonparametric Measures* (Sage University Papers Series). Newbury Park, CA, USA: Sage, 2002, pp. 137–139. [Online]. Available: https://books.google.ie/books?id=UN4nAQAAIAAJ

[15] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul./Oct. 1948.

[16] D. S. Nascimento and A. L. Coelho, "Ensembling heterogeneous learning models with boosting," in *Proc. Int. Conf. Neural Inf. Process.* Bangkok, Thailand: Springer, 2009, pp. 512–519.

[17] J. A. Nelder and R. W. Wedderburn, "Generalized linear models," *J. Roy. Stat. Soc., A (Gen.)*, vol. 135, no. 3, pp. 370–384, 1972.

[18] J. Hastie, T. Trevor, and F. Robert, *The Elements of Statistical Learning*, 2nd ed. New York, NY, USA: Springer, 2009.

[19] P. Cortez and M. J. Embrechts, "Using sensitivity analysis and visualization techniques to open black box data mining models," *Inf. Sci.*, vol. 225, pp. 1–17, Mar. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025512007098

[20] P. Cortez, *Rminer: Data Mining Classification Regression Methods, R Package Version 1.4.2*. Comprehensive R Archive Network, 2016. [Online]. Available: https://CRAN.R-project.org/package=rminer

[21] H. A. Chipman, E. I. George, and R. E. McCulloch, "BART: Bayesian additive regression trees," *Ann. Appl. Statist.*, vol. 4, no. 1, pp. 266–298, Mar. 2010.

[22] J. H. Friedman, "Rejoinder: Multivariate adaptive regression splines," *Ann. Statist.*, vol. 19, no. 1, pp. 123–141, Mar. 1991.

[23] H. Deng, "Guided random forest in the RRF package," 2013, *arXiv:1306.0237*. [Online]. Available: http://arxiv.org/abs/1306.0237

[24] S. Ramaswamy, K. N. Ross, E. S. Lander, and T. R. Golub, "A molecular signature of metastasis in primary solid tumors," *Nature Genet.*, vol. 33, no. 1, pp. 49–54, Jan. 2003.

[25] S. L. Pomeroy, P. Tamayo, M. Gaasenbeek, L. M. Sturla, M. Angelo, M. E. McLaughlin, J. Y. Kim, L. C. Goumnerova, P. M. Black, and C. Lau, "Prediction of central nervous system embryonal tumour outcome based on gene expression," *Nature*, vol. 415, no. 6870, pp. 436–442, 2002.

[26] L. J. Van't Veer, "Gene expression profiling predicts clinical outcome of breast cancer," *Nature*, vol. 415, no. 6871, p. 530, 2002.

[27] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine, "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," *Proc. Nat. Acad. Sci. USA*, vol. 96, no. 12, pp. 6745–6750, Jun. 1999.

[28] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander, "Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring," *Science*, vol. 286, no. 5439, pp. 531–537, Oct. 1999.

[29] A. A. Alizadeh, M. B. Eisen, R. E. Davis, C. Ma, I. S. Lossos, A. Rosenwald, J. C. Boldrick, H. Sabet, T. Tran, and X. Yu, "Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling," *Nature*, vol. 403, no. 6769, p. 503, 2000.

[30] D. T. Ross, U. Scherf, M. B. Eisen, C. M. Perou, C. Rees, P. Spellman, V. Iyer, S. S. Jeffrey, M. Van de Rijn, and M. Waltham, "Systematic variation in gene expression patterns in human cancer cell lines," *Nature Genet.*, vol. 24, no. 3, pp. 227–235, 2000.

[31] D. Singh, P. G. Febbo, K. Ross, D. G. Jackson, J. Manola, C. Ladd, P. Tamayo, A. A. Renshaw, A. V. D'Amico, J. P. Richie, E. S. Lander, M. Loda, P. W. Kantoff, T. R. Golub, and W. R. Sellers, "Gene expression correlates of clinical prostate cancer behavior," *Cancer Cell*, vol. 1, no. 2, pp. 203–209, Mar. 2002.

[32] J. Khan, J. S. Wei, M. Ringner, L. H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C. R. Antonescu, and C. Peterson, "Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks," *Nature Med.*, vol. 7, no. 6, pp. 673–679, 2001.

[33] M. N. Wright and A. Ziegler, "Ranger: A fast implementation of random forests for high dimensional data in C++ and r," *J. Stat. Softw.*, vol. 77, no. 1, pp. 1–17, 2017.

[34] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002. [Online]. Available: https://CRAN.R-project.org/doc/Rnews/

[35] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, vol. 2. New York, NY, USA: Curran Associates, 2012, pp. 2951–2959. [Online]. Available: http://dl.acm.org/citation.cfm?id=2999325.2999464

[36] R. Johnson and T. Zhang, "Learning nonlinear functions using regularized greedy forest," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 5, pp. 942–954, May 2014.

[37] F. Nan and V. Saligrama, "Adaptive classification for prediction under a budget," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4727–4737.

[38] F. Nan, J. Wang, and V. Saligrama, "Pruning random forests for prediction on a budget," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2334–2342.

[39] T. Chen, T. He, M. Benesty, V. Khotilovich, and Y. Tang, "Xgboost: Extreme gradient boosting," in *R package version 0.4-2*. Comprehensive R Archive Network, 2015, pp. 1–4.

[40] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3146–3154.

BRUNA WUNDERVALD received the bachelor's degree (Hons.) in statistics from the Federal University of Paraná, Brazil, in 2018. She is currently pursuing the Ph.D. degree in statistics with Maynooth University, Ireland. Her research interests include the wide variety of machine learning methods, including dimensionality reduction, Bayesian additive regression trees (BART), variational inference, and probabilistic graphical models.

ANDREW C. PARNELL received the Ph.D. degree in statistics from the University of Sheffield, U.K. He is currently a Hamilton Professor with the Hamilton Institute, Maynooth University, Ireland. His research interests include statistics and machine learning for large structured data sets in a variety of application areas, including tree-based machine learning methods, Bayesian additive regression trees (BART), extreme value theory, spatial statistics, zero-inflation modeling, and missing data analysis. He has been heavily involved in the commercialization of research. He is currently a Principal Investigator with the SFI I-Form Advanced Manufacturing Centre, a funded investigator in the SFI Insight Centre for Data Analytics, and an Expert Reviewer for the Intergovernmental Panel on Climate Change.

KATARINA DOMIJAN received the Ph.D. degree in statistics from Trinity College in Dublin, Ireland. She is currently a Lecturer of statistics with the Department of Mathematics and Statistics, Maynooth University. Her main research interests rely on Bayesian modeling of classification problems in data with large feature spaces, including feature selection, model visualization, and interpretability.