RANDOM FORESTS AND REGULARIZATION

by

Siyu Zhou

B.S. in Mathematics, The Hong Kong University of Science and Technology, 2014

M.Phil. in Mathematics, The Hong Kong University of Science and Technology, 2016

Submitted to the Graduate Faculty of

the Dietrich School of Arts and Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2022

UNIVERSITY OF PITTSBURGH

DIETRICH SCHOOL OF ARTS AND SCIENCES

This dissertation was presented

by

Siyu Zhou

It was defended on

June 16, 2022

and approved by

Lucas K. Mentch, Ph.D., Department of Statistics

Yu Cheng, Ph.D., Department of Statistics

Satish Iyengar, Ph.D., Department of Statistics

Larry Wasserman, Ph.D., Department of Statistics and Data Science, Carnegie Mellon University

Dissertation Director: Lucas K. Mentch, Ph.D., Department of Statistics

RANDOM FORESTS AND REGULARIZATION

Siyu Zhou, PhD

University of Pittsburgh, 2022

Random forests have a long-standing reputation as excellent off-the-shelf statistical learning methods. Despite their empirical success and numerous studies on their statistical properties, a full and satisfying explanation for their success has yet to be put forth. This work takes a step in this direction by demonstrating that random-feature-subsetting provides an implicit form of regularization, making random forests more advantageous in low signal-to-noise ratio (SNR) settings. Moreover, this is not a tree-specific finding but can be extended to ensembles of base learners constructed in a greedy fashion. Inspired by this, we find inclusion of additional noise features can serve as another implicit form of regularization and thereby lead to substantially more accurate models. As a result, intuitive notions of variable importance based on improved model accuracy may be deeply flawed, as even purely random noise can routinely register as statistically significant. Along these lines, we further investigate the effect of pruning trees in random forests. Despite the fact that full depth trees are recommended in many textbooks, we show that tree depth should be seen as a natural form of regularization across the entire procedure with shallow trees preferred in low SNR settings.

Keywords: Random Forests, Bagging, Regularization, Interpolation, Ridge Regression, Model Selection.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 INTRODUCTION

Since their inception in 2001, random forests (RFs) [Breiman, 2001] have remained among most popular and successful off-the-shelf statistical machine learning methods with a well-established record in numerous scientific fields. To name a few, in the area of bioinformatics, Díaz-Uriarte and De Andres [2006] recommended RFs should be part of the standard tool-box for gene selections for diagnostic purposes in clinical trials and Mehrmohamadi et al. [2016] established links between tumour metabolism and epigenetics with RFs. Svetnik et al. [2003] demonstrated RFs' capability of delivering accurate performance, handling high-dimensional data and maintaining ease of training and computational efficiency, making them suited for for a QSAR modeling and compound classification in modern drug discovery and development process. In ecology, RFs' high accuracy was established once more for both regression and classification tasks by Prasad et al. [2006] and Cutler et al. [2007] respectively. Furthermore, RFs were also beneficial to modern technology advancements such as image recognition [Bernard et al., 2007, Huang et al., 2010, Guo et al., 2011, Fanelli et al., 2013], 3D object recognition [Shotton et al., 2011] and so on. In a recent large-scale empirical study [Fernández-Delgado et al., 2014], RFs were found to be the top classifiers against hundreds of alternatives compared on 121 datasets, which represented the whole UCI [Dua and Graff, 2017] database at that time.

Such consistent successes of RFs across various domains naturally lead to the study of its mathematical and statistical properties, the first of which was an upper bound on the generalization error of RFs given by Breiman [2001]. Despite the simplicity of the algorithm itself, difficulty in developing rigorous mathematical analysis of the original algorithm in Breiman [2001] originates from two essential ingredients of the procedure, the Classification and Regression Trees (CART) [Breiman et al., 1984]-splitting scheme and the bagging process

[Breiman, 1996].

The CART splitting scheme came from the CART algorithm for tree construction by choosing the *best* cut perpendicular to the axes. This criterion depends on both the features and response, and thus is difficult to analyze. Instead, many studies focused on *nonadaptive* forests, where the splitting criterion either is completely random or depends on features only. Lin and Jeon [2006] provided a lower bound to the rate of convergence of the MSE of random forests with nonadaptive splitting schemes by introducing a potential nearest neighbourhood framework, the consistency of which was later studied in Biau and Devroye [2010]. Biau et al. [2008] established consistency of the purely random forest considered in Breiman [2000b] where each internal node is split along a randomly chosen feature at a random location. The consistency of the centered random forest studied in Breiman [2004] where the split was at the midpoint (random) along a strong (weak) feature was proved by Biau [2012] and generalized to cases when the best splits were chosen based on a second independent sample. Scornet [2016] further generalized the results in Biau et al. [2008] and Biau [2012] by proving the respective consistency of nonadaptive forests and q quantile forests where internal nodes are split around quantiles while Klusowski [2019] improved the rate of convergence in Biau [2012].

Another difficulty results from the bagging procedure, which stabilizes estimates by aggregating outputs of trees built on bootstrapped samples that contain duplicated observations. Breiman [2004] and Lin and Jeon [2006] omitted the bagging procedure while many later work concentrating on RFs constructed with subsamples (where observations are resampled without replacement) rather than bootstrap samples have established important statistical properties of RFs. The first consistency for Breiman's original random forests on an additive true underlying model was given by Scornet et al. [2015] while Mentch and Hooker [2016] provided the first result on the asymptotic normality of RFs' predictions in the framework of infinite-order generalized U-statistics, accompanied by a testing procedure for variable importance which was extended in Mentch and Hooker [2017] for testing additivity of underlying models. Coleman et al. [2019] proposed a more computationally efficient permutation-based procedure for testing variable importance, which scales easily to big data setting and is more feasible for pratical scientific use. Wager and Athey [2018] established

the consistency and asymptotic normality for honesty and causal forests. Assumptions to achieve asymptotic normal predictions were weakened by Peng et al. [2019] who also provided Berry-Essen bounds to quantify the rate of convergence.

Apart from these, Sexton and Laake [2009] and Wager et al. [2014] provided estimations for the variance of RFs predictions. Lopes et al. [2019b] and Lopes et al. [2019a] developed a bootstrap method for measuring the algorithm convergence in the classification and regression setting respectively. The RF methodology has also been extended to other areas such as clustering [Yan et al., 2013], survival analysis [Hothorn et al., 2005, Ishwaran et al., 2008, Cui et al., 2017], quantile regression [Meinshausen, 2006], online learning [Yi et al., 2012, Lakshminarayanan et al., 2014]and reinforcement learning [Zhu et al., 2015], to name a few. Biau and Scornet [2016] gives a more detailed and comprehensive guide.

Despite RFs' well established records and numerous studies on their mathematical and statistical properties, as mentioned in a recent review paper Biau and Scornet [2016], "present results are insufficient to explain in full generality the remarkable behaviour of random forests". Although many studies [Genuer et al., 2008, Bernard et al., 2009, Genuer et al., 2010, Duroux, Roxane and Scornet, Erwan, 2018, Scornet, 2017, Probst and Boulesteix, 2017, Probst et al., 2019b] experimented with tuning the RF procedure, the main takeaways from these works have been high-level and heuristic and the main conclusion is that including more trees in the forest helps stabilize predictions while tuning other parameters can sometimes provide an improvement in accuracy. There have been efforts as well connecting RFs with other frameworks such as kernel estimates [Arlot and Genuer, 2014, Scornet, 2016, Olson and Wyner, 2018] and neural network [Welbl, 2014, Biau et al., 2019]. Wyner et al. [2017] considered RFs as "self-averaging interpolators" and hypothesized that such behaviour led to the success of RFs. In the same manner, Belkin et al. [2019] put forth the more general and now very popular idea of "double descent" risk curve which suggests improved model performance be gained once the complexity goes beyond interpolation threshold and provided empirical evidences with several different models including neural networks and RFs. However, as discussed in later chapters, these idea have significant issues.

In Chapter 2, we seek to provide an explanation for the success of RFs from the degrees-of-freedom point of view by isolating the extra randomness of RFs at tree splits and demon-

strating that such randomness provides an implicit regularization effect similar to ridge regression [Hoerl and Kennard, 1970] and lasso [Tibshirani, 1996]. In Chapter 3, we further investigate that other forms of regularization on ensembles of trees can result in improved performance, just as in RFs. Surprisingly, the inclusion of noise features (conditionally) independent of responses is indeed one option and can produce dramatical improvement, even more than that given by optimally tuned RFs in some settings. This has a crucial impact on how we consider and measure variable importance. Chapter 4 is devoted to the inherent but often neglected regularization effect from tree depth and argues that the noticeable jumps in random forest accuracy are the result of simple averaging rather than interpolation. We conclude with a discussion in Chapter 5.

2.0 RANDOMIZATION AS REGULARIZATION: AN EXPLANATION FOR THE SUCCESS OF RANDOM FORESTS

The work in this chapter argues that the RF's success is due to an implicit regularization effect of the additional randomness. The following sections pull heavily from Mentch and Zhou [2020b].

2.1 INTRODUCTION

The work presented in this chapter offers a concrete explanation for the role played by the extra randomness most commonly injected into the base learners in RF procedures. Instead of assuming that RFs simply do "work well", we take a more principled approach in trying to isolate the effects of that additional randomness and determine when its inclusion results in improved accuracy over a baseline approach like bagging that uses non-randomized base learners. In particular, we argue that the additional randomness serves to regularize the procedure, making it highly advantageous in low signal-to-noise ratio settings. Speculation along these lines was hypothesized informally in Hastie et al. [2009] who observe that RFs sometimes behave similarly to ridge regression.

To drive home this point, we further demonstrate that incorporating similar randomness into alternative (non tree-based) model selection procedures can result in improved predictive accuracy over existing methods in exactly the settings where such improvements would be expected. In particular, inspired by recent work on degrees of freedom for model selection by Tibshirani [2015] and Hastie et al. [2020], we consider two randomized forward selection procedures for linear models designed as analogues to classical bagging and RFs and demon-

strate the same kind of regularization properties. Our findings in this setting are thus similar in spirit to those produced by Wager et al. [2013] who demonstrate a regularization effect arising from dropout training applied to generalized linear models.

The remainder of this paper is laid out as follows. In Section 2.2 we formalize the RF procedure and continue the above discussion, providing something of a literature review of recent RF analyses as well as a more detailed overview of the shortcomings of existing explanations for their success. In Section 2.3 we discuss degrees of freedom for model selection procedures and demonstrate that within a traditional RF context, more randomness results in procedures with fewer degrees of freedom. We emphasize and build upon this finding in Section 2.4 by demonstrating in numerous settings using both real and synthetic data that the relative improvement in accuracy offered by RFs appears directly related to the relative amount of signal contained within the data. Finally, in Section 2.5 we introduce the linear model forward-selection-style analogues for bagging and RFs and produce near identical results, finding in particular that in noisy low-dimensional settings, injecting randomness into the selection procedure can outperform even highly competitive explicit regularization methods such as the lasso. Example code for the simulations and experiments presented is available at `https://github.com/syzhou5/randomness-as-regularization`.

2.2 RANDOM FORESTS AND EXISTING EXPLANATIONS

We begin by formalizing the RF framework in which we will work in the following sections. Unless otherwise noted, throughout the remainder of this chapter we will consider a general regression framework in which we observe (training) data of the form $\mathcal{D}_n = \{\boldsymbol{Z}_1, ..., \boldsymbol{Z}_n\}$ where each $\boldsymbol{Z}_i = (\boldsymbol{X}_i, Y_i)$, $\boldsymbol{X}_i = (X_{1,i}, ..., X_{p,i})$ denotes a vector of p features, $Y \in \mathbb{R}$ denotes the response, and the variables have a general relationship of the form

$$Y = f(\boldsymbol{X}) + \epsilon \tag{2.1}$$

where ϵ is often assumed to be independent noise with mean 0 and variance σ^2.

To construct a tree, we begin by resampling $a_n \leq n$ observations from $\mathcal{D}_n$ with or without replacement. The original RF formulation utilized bootstrapping so that $a_n = n$ and the sampling is done with replacement, though a number of recent theoretical advances have been made by instead considering subsampling (without replacement) with $a_n = o(n)$. At each step, `mtry` $\leq p$ eligible features are selected uniformly at random, among which the optimal split is obtained by maximizing the CART criterion [Breiman et al., 1984]. Specifically, an internal node t is split in an axis-aligned fashion into left and right daughter nodes of the form $t_L = \{\boldsymbol{x} \in t : x_j \leq s\}$ and $t_R = \{\boldsymbol{x} \in t : x_j > s\}$ whenever the decision is made to split the feature X_j at s. The particular variable and split location are chosen from among those available as that pair which minimizes the resulting within-node variance of the offspring in regression settings or maximizes the empirical reduction in Gini impurity in classification settings. The tree continues to split until the number of observations in each cell is less than the pre-specified `nodesize` or whenever the number of terminal nodes (leaves) reaches `maxnodes`. In the case of regression, which will be the main focus of this dissertation, to obtain the prediction at any given point $\boldsymbol{x}$, the response values are averaged across all observations that fall into the same leaf as $\boldsymbol{x}$. To form a RF, the procedure is repeated B times and the final prediction is simply the average across all B tree-level predictions. In classification settings, the standard approach is to form final estimates via a majority vote at both the tree and forest level. Algorithm 1 provides a more detailed summary of this process, which follows closely to Algorithm 1 in Biau and Scornet [2016]. Readers less familiar with trees and forests are invited to see Biau and Scornet [2016] for a more detailed discussion.

Mathematically, for a given point $z = (\boldsymbol{x}, y)$, a RF prediction at $\boldsymbol{x}$ takes the form

$$\hat{y} = \mathrm{RF}(\boldsymbol{x}; \mathcal{D}_n, \Theta) = \frac{1}{B} \sum_{b=1}^{B} T(\boldsymbol{x}; \mathcal{D}_n, \Theta_b) \tag{2.2}$$

where the base-learners T are typically tree-based models constructed on some resample of the original data $\mathcal{D}_n$ and the randomness involved in the procedure is indexed by Θ_b.

Throughout the literature on RFs, it is common to succinctly contain all randomness in the single term Θ_b as in equation (2.2) above. We note however that for our purposes below, it may be convenient to consider this more explicitly as $\Theta_b = (\Theta_{\mathcal{D},b}, \Theta_{mtry,b})$. Written in this form, $\Theta_{\mathcal{D},b}$ serves to select the resample of the original data utilized in the b^{th} tree. While

Algorithm 1 Breiman's (regression) random forest

Input: Training set $\mathcal{D}_n$, number of trees $B > 0$, $a_n \in \{1, \ldots, n\}$, `mtry`$\in \{1, \ldots, p\}$, `nodesize`$\in \{1, \ldots, a_n\}$, and $\boldsymbol{x} \in \mathcal{X}$

Output: Prediction of the random forest at $\boldsymbol{x}$

for $b = 1, \ldots, B$ **do**

Select a_n points, with (or without) replacement, uniformly in $\mathcal{D}_n$. In the following steps, only these a_n observations are used.

Set $\mathcal{P} = (\mathcal{X})$ the list containing the cell associated with the root of the tree.

Set $\mathcal{P}_{\text{final}} = \emptyset$ an empty list.

while $\mathcal{P} \neq \emptyset$ **do**

Let A be the first element of $\mathcal{P}$.

if A *contains less than* `nodesize` *points or if all* $\boldsymbol{X}_i \in A$ *are equal* **then**

Remove the cell A from the list $\mathcal{P}$.

$\mathcal{P}_{\text{final}} \leftarrow Concatenate(\mathcal{P}_{\text{final}}, A)$.

else

Select uniformly, without replacement, a subset $\mathcal{M}_{\text{try}} \subset \{1, \ldots, p\}$ of cardinality `mtry`.

Select the best split in A by optimizing the CART-split criterion along the coordinates in $\mathcal{M}_{\text{try}}$.

Cut the cell A according to the best split. Call A_L and A_R the two resulting cells.

Remove the cell A from the list $\mathcal{P}$.

$\mathcal{P} \leftarrow Concatenate(\mathcal{P}, A_L, A_R)$.

Compute the predicted value of the b^{th} tree at $\boldsymbol{x}$ equal to the average of the Y_i falling in the cell of $\boldsymbol{x}$ in the partition $\mathcal{P}_{\text{final}}$.

Compute the random forest estimate at the query point $\boldsymbol{x}$.

much recent work has focused on subsampled RFs, here we consider the B resamples to be bootstrap samples as originally put forth in Breiman [2001]. The second randomization

component $\Theta_{mtry,b}$ then determines the $\texttt{mtry} \leq p$ candidate features to be split at each node in the b^{th} tree. When $\texttt{mtry} = p$, the procedure reduces to bagging [Breiman, 1996].

2.2.1 Explanations for Random Forest Success

The original reasoning behind RFs provided by Breiman [2001] was based on an extension of the randomized tree analysis given in Amit and Geman [1997]. Breiman showed that the accuracy of any randomized ensemble depends on two components: the strength (accuracy) of the individual base-learners and the amount of dependence between them. Thus, the original motivation for a procedure like RFs might be seen from a statistical perspective as akin to the classic bias-variance tradeoff. In the same way that some procedures (e.g. the lasso [Tibshirani, 1996, Chen et al., 2001]) consider trading a small amount of bias in exchange for a large reduction in variance, RF ensembles might be seen as trading a small amount of accuracy at the base-learner level (by injecting the extra randomness) for a large reduction in between-tree correlation. Hastie et al. [2009] provide a thorough, high-level discussion of this effect in showing that the `mtry` parameter serves to reduce the variance of the ensemble.

However, this discussion from Breiman [2001] might be better seen as motivation for why a randomized ensemble could *potentially* improve accuracy rather than an explanation as to why RFs in particular *do* seem to work well. Breiman himself experiments with different kinds of randomness in the original manuscript and suggests that in practice users can also experiment with different forms to try and determine what works best in particular settings. Furthermore, in his concluding remarks, Breiman notes that while the additional randomness at the base learner level helps to reduce the variance of the ensemble, the magnitude of improvement often seen with RFs suggested to him that perhaps it somehow also "act[s] to reduce bias" but that ultimately "the mechanism for this [was] not obvious." In the years since, it has been shown quite clearly that the benefits sometimes seen with RFs are the result of variance reduction alone; see Hastie et al. [2009] for a more complete discussion.

In recent work, Wyner et al. [2017] take a more definitive stance, conjecturing that both RFs and AdaBoost [Freund et al., 1996] work well because both procedures are "self-

averaging interpolators" that fit the training data perfectly while retaining some degree of smoothness due to the averaging. The key to their success, they argue, is that in practice, datasets often contain only small amounts of noise and these algorithms are able to mitigate the effects of noisy data points by localizing their effect so as to not disturb the larger regions where the data consists mostly of signal. Indeed, the authors acknowledge that the procedures "do in fact overfit the noise – but only the noise. They do not allow the overfit to metastasize to modestly larger neighborhoods around the errors."

2.2.2 Random Forests and Interpolation

As the central claim of Wyner et al. [2017] is that RFs "work not in spite, but because of interpolation" we now make this notion and argument explicit.

Definition 1 (Interpolation)**.** *A classifier (or regressor)* $\hat{f}$ *is said to be an* interpolating *classifier (regressor) if for every training point* $(\boldsymbol{x}_j, y_j) \in \mathcal{D}_n$, $\hat{f}(\boldsymbol{x}_j) = y_j$.

This definition of an interpolating classifier is taken directly from Wyner et al. [2017]; for completeness and because it will be directly relevant to the immediate conversation, we expand the definition so as to apply in the same fashion to regression contexts.

Consider first the classification setting wherein the response $Y \in \{a_1, ..., a_k\}$ and we seek to utilize the training data $\mathcal{D}_n$ to construct a classifier $\hat{f}_n : \mathcal{X} \mapsto \{a_1, ..., a_k\}$. Consider a particular observation $z = (\boldsymbol{x}, y) \in \mathcal{D}_n$. In order to be more explicit and without loss of generality, suppose that $y = a_1$. Given B resamples of the original data $\mathcal{D}_1^*, ..., \mathcal{D}_B^*$, whenever trees are fully grown so that each terminal node contains only a single observation, it must necessarily be the case that $T(\boldsymbol{x}; \mathcal{D}_i^*, \Theta_i) = a_1$ (i.e. the tree predicts the correct class a_1 for $\boldsymbol{x}$) whenever $(\boldsymbol{x}, y) \in \mathcal{D}_i^*$. Thus, in order for the RF classifier to select the correct class for $\boldsymbol{x}$, (i.e. $\text{RF}(\boldsymbol{x}) = a_1$) it suffices to ensure that $(\boldsymbol{x}, y)$ is selected in a plurality (or simple majority in the case of binary classification) of the resamples. When bootstrap samples are used, it is well known that the probability of each observation appearing is approximately 0.632. Thus, given B bootstrap samples, the probability that the observation $(\boldsymbol{x}, y)$ appears in at least half of these resamples is approximately

$$p_{int}(B) = 1 - Bin(B/2;\, n = B, p = 0.632)$$

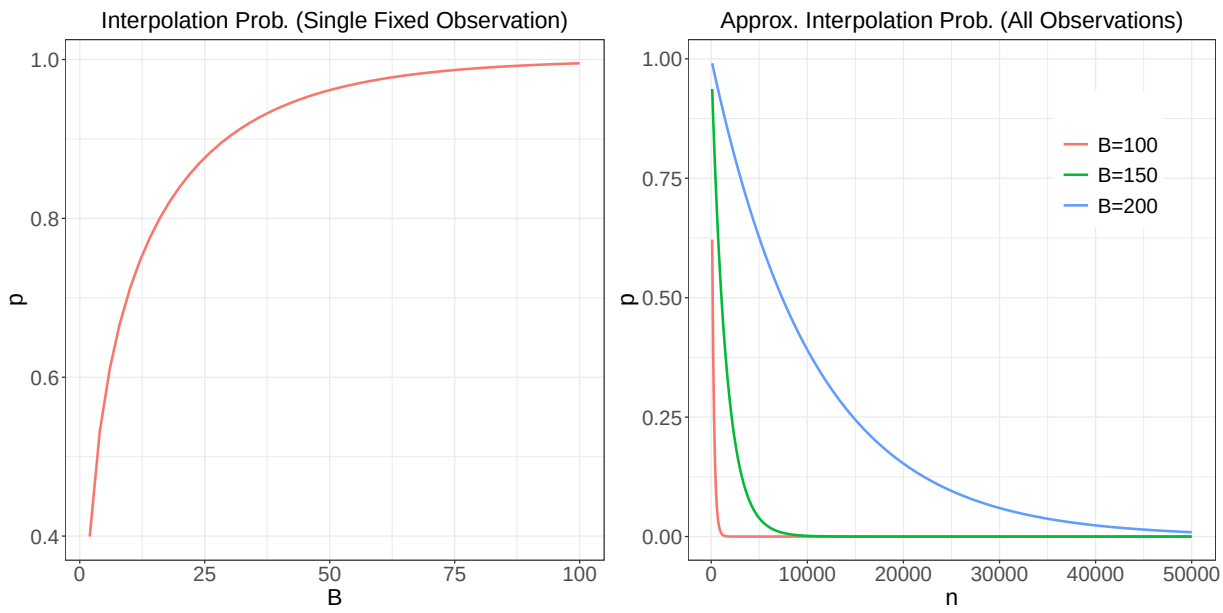

Figure 2.1: (Left): Interpolation probability vs. number of trees (B) for a single observation. (Right): Approximate interpolation probabilities vs. sample sizes for RF classifiers built with different numbers of trees. Both plots pertain to the binary classification setting.

where $Bin(z; n, p)$ denotes the binomial cdf evaluated at the point z with parameters n (the number of trials) and p (the probability of success in each trial). For even moderately large B, this probability is quite large; see the left plot in Figure 2.1. Thus, for binary classification problems, the probability of interpolating any given training observation is large whenever B is also moderately large.

Note however that according to the definition above, a classifier is only designated as an interpolator if it interpolates *all* training observations. While an exact calculation for the probability of all n points appearing in at least half of the bootstrap samples is somewhat involved, we can approximate it with $(p_{int}(B))^n$, the calculation that would result if the interpolation probabilities were independent for each observation. Plots of this quantity are shown in the right plot in Figure 2.1 across a range of sample sizes for $B = 100, 150,$ and 200. In each case, for fixed B, the (approximate) probability that the classifier is an interpolator tends to 0 as $n \to \infty$, suggesting that in order for RF classifiers to necessarily interpolate,

the number of bootstrap replicates B must be treated as an increasing function of n. Though perhaps obvious, we stress that this is not generally the manner in which such estimators are constructed. In all software with which we are familiar, default values of B are set to a fixed moderate size, independent of the size of the training set. The `bagging` function in the `ipred` package in `R` [Peters et al.], for example, takes 25 bootstrap replicates by default. Recent work from Lopes et al. [2019b] has also provided a means of estimating the algorithmic variance of RF classifiers and shown that it sometimes vanishes quite quickly after relatively few bootstrap samples. Thus, while it's possible to construct RFs in such a way that they necessarily interpolate with high probability in classification settings, it is not clear that RFs would generally be constructed in this fashion in practice and thus it is also not clear that the interpolation-based explanation offered by Wyner et al. [2017] is sufficient to explain the strong performance of RFs, even in specific contexts. It is also worth noting that on certain datasets where RFs happen to produce good models with low generalization error, they may likely also fit quite well on the training data, perhaps even nearly interpolating. This, however, is certainly possible for any modeling procedure and thus in no way would aid in explaining the particular success of RFs.

2.2.3 Shortcomings of Current Explanations

Before continuing with our critique, it's worth pausing to note where the existing explanations are in agreement. Both Breiman [2001] and Wyner et al. [2017] seem to largely agree on the following points:

1. Random forests and boosting behave in a very similar fashion and thus their success should be able to be explained in a very similar fashion [Breiman [2001] pages 6, 20, Section 7; Wyner et al. [2017] entire paper].
2. Random forests and boosting generally outperform most other competing methods (e.g. Dietterich [2000] and Breiman [2000a]) in terms of minimizing generalization error and substantially outperform bagging [Breiman [2001] page 10; Wyner et al. [2017] page 3].
3. Random forests generally seem to be robust to outliers and noise [Breiman [2001] pages 10, 21; Wyner et al. [2017] pages 4, 12, 17, 20, 32, 35].

4. Boosting tends to perform well and not overfit even when the ensemble consists of many deep trees [Breiman [2001] page 21; Wyner et al. [2017] page 8].

Points 1 and 4 are largely irrelevant to the discussion in the remainder of this chapter as we focus exclusively on random forests; we include these points here only in the interest of completeness. Point 3 has been alluded to in numerous papers throughout the years and has likely been key to the sustained popularity of the random forest procedure.

We take slight issue with the now popular wisdom in the second point, that random forests simply "are better" than bagging or other similar randomized approaches. While this does seem to be the case surprisingly often in practice on real-world datasets (see, for example, the recent large-scale comparison from Fernández-Delgado et al. [2014] discussed in the introduction) it is certainly not a universal truth and, in our view, is a potentially naive foundation on which to build a theory for explaining their success. As discussed above, Breiman [2001] does provide some motivation for why a randomized ensemble might sometimes outperform its nonrandomized counterpart in showing that the generalization error of a classifier can be bounded above by a function of base-learner accuracy and correlation. Breiman stops short, however, of providing any more explicit explanation for the role played by the randomness or in what situations that randomness might be expected to help the most. Wyner et al. [2017], on the other hand, seem to largely ignore the role of randomness altogether. The explanation the authors provide for random forest success would seem to apply equally well to bagging. In the sections below, we focus our attention heavily on determining when the inclusion of such randomness provides the greatest benefit and provide an explicit characterization of the role it plays.

It is also important to stress that the theories offered by Breiman [2001] and Wyner et al. [2017] pertain only to the classification setting, whereas our focus is primarily on regression. The interpolation hypothesis put forth by Wyner et al. depends on an even stricter setup whereby trees are built to full depth, bootstrapping (or at least subsampling without replacement at a rate of at least $0.5n$) is used to generate resamples, and the number of trees B grows with n at a sufficiently fast rate. Previous work, however, has repeatedly shown that random forests can still achieve a high degree of predictive accuracy when trees are not built to full depth [Duroux, Roxane and Scornet, Erwan, 2018], and/or are constructed via

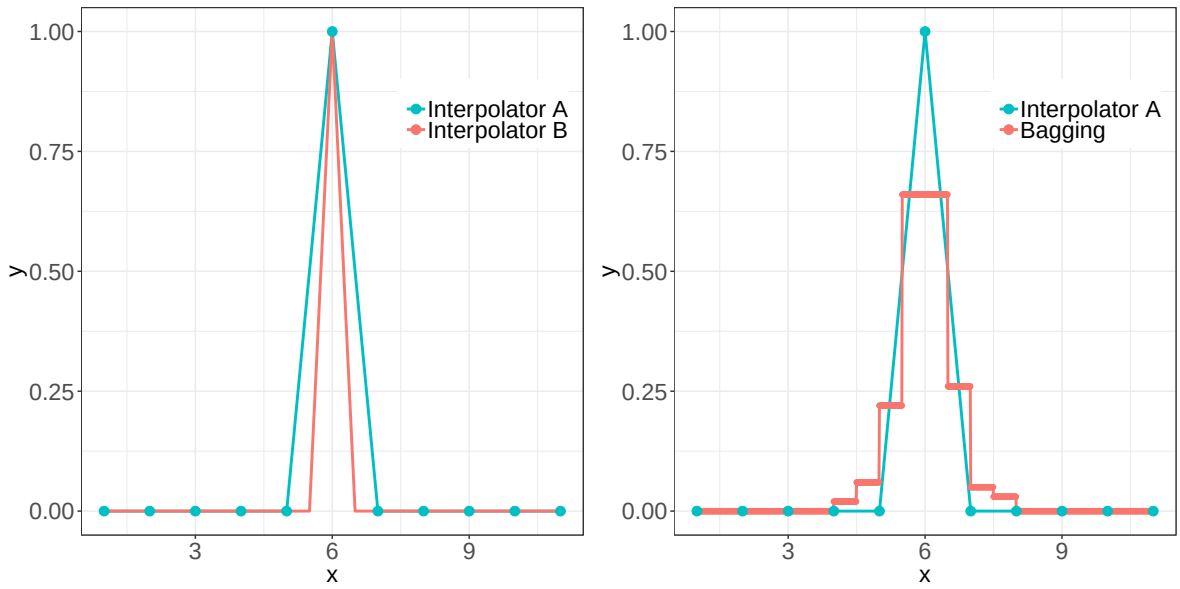

Figure 2.2: Left: Two interpolating regressors on toy data. Right: An interpolating regressor and the predicted regression function resulting from bagging with 100 fully-grown trees.

subsampling [Zaman and Hirose, 2009, Mentch and Hooker, 2016, Wager and Athey, 2018], and/or when relatively few trees are built [Lopes et al., 2019b].

To see why random forests cannot be considered interpolators in a regression setting, even when individual trees are built to full depth, note that the final regression estimate is taken as the average of predictions across all trees rather than the majority vote. While it's certainly clear that an average of interpolators is itself an interpolator since for every training point $(\boldsymbol{x}, y)$

$$\frac{1}{B}\sum_{i=1}^{B}\hat{f}_i(\boldsymbol{x}) = \frac{1}{B}\sum_{i=1}^{B} y = y\,,$$

the bootstrapping mechanism in play with random forests precludes the possibility of interpolating on $\mathcal{D}_n$ with exceedingly high probability. As noted above, each bootstrap sample will omit, on average, 36.8% of the original observations and thus individual trees, even if fully grown, will not, in general, fit perfectly on that out-of-sample (out-of-bag) data. In other words, while a fully-grown tree will necessarily interpolate the observations selected within its corresponding bootstrap sample, it will generally not fit perfectly to all training

observations. For a given point $(\boldsymbol{x}, y)$ in the training data, random forest regression estimates at x are therefore a weighted average of y and the other response values observed, and hence with exceedingly high probability, the random forest will not interpolate.

Figure 2.2 demonstrates this effect clearly. The left panel shows two hand-crafted interpolating functions – Interpolator A and Interpolator B – on a simple toy dataset while the right panel shows one of the same interpolators along with the predicted regression function resulting from bagging with 100 regression trees, each grown to full depth. In this toy example motivated by the examples shown in Section 3.2 of Wyner et al. [2017], our training data consists of 11 points each of the form $(i, 0)$ for $i = 1, ..., 11$ except for $i = 6$ where we instead have the observation $(6, 1)$. Denote the location of this point by x^*. Wyner et al. [2017] contend that if the observed response is considered "unusually noisy" at x^*, then interpolating estimators can perform well by "localizing" the effect of this noise as is seen in the left panel of Figure 2.2. Indeed, we can see from this plot that both interpolators still fit the remaining data perfectly despite the presence of the noisy observation. However, as can be seen in the right-hand panel, whenever we treat this as a regression problem and build trees to full depth, the random forest (in this case, simple bagging since we have only 1 feature) does not interpolate, but instead looks to be attempting to smooth-out the effect of the outlying observation.

This, however, stands in opposition to the reasoning provided in Wyner et al. [2017]. Here the authors are highly critical of the traditional statistical notion of signal and noise and seem to take some issue with the general regression setup given in (2.1). To computer scientists, they claim, in many problems "there is no noise in the classical sense. Instead there are only complex signals. There are residuals, but these do not represent irreducible random errors." But if the widespread empirical success of random forests is really "not in spite, but because of interpolation" as claimed, then one must believe that real-world data is generally low-noise, a claim argued against firmly by, for example, Hastie et al. [2020]. Crucially, this means that not only are data largely free of what scientists may think of as classical kinds of noise like measurement error, but also that all sources of variation in the response Y can be explained almost entirely by the available set of predictor variables $X_1, ..., X_p$.

Perhaps most importantly, if the success of random forests is the result of interpolation and interpolation is beneficial because most real-world datasets have a high signal-to-noise ratio (SNR), then random forests ought not perform well at all on datasets with low SNRs. Consider again the plot on the left-hand-side of Figure 2.2. If the outlying point x^* at $(6, 1)$ is actually the only "good signal" while the rest of the data are noisy, then the interpolators shown would be isolating signal rather than noise and hence be performing quite poorly.

But this is exactly the opposite of what we see with random forests in practice. In the following sections, we show repeatedly on both real and synthetic data that relative to procedures like bagging that utilize non-randomized base learners, the benefit of the additional randomness is most apparent in low SNR settings. In Section 2.3 we show that the `mtry` parameter has a direct effect on the degrees of freedom (dof) associated with the procedure, with low values of `mtry` (i.e. more randomness) corresponding to the least flexible model forms with the fewest dof. Given this, in Section 2.4 we go on to show that as expected, the advantage offered by random forests is most dramatic at low SNRs, and that this advantage is eventually lost to bagging at high SNRs. We also consider the problem from a slightly different perspective and show that the optimal value of `mtry` is almost perfectly (positively) correlated with the SNR. We posit that this behavior is due to a regularizing effect caused by the randomness and bolster this claim by demonstrating the same relatively surprising results hold in simpler linear model setups where randomness is injected into a forward selection process.

2.3 RANDOM FORESTS AND DEGREES OF FREEDOM

Recall from the previous section that we assume data of the form $\mathcal{D}_n = \{Z_1, ..., Z_n\}$ where each $Z_i = (\boldsymbol{X}_i, Y_i)$, $\boldsymbol{X}_i = (X_{1,i}, ..., X_{p,i})$ denotes a vector of p features, $Y \in \mathbb{R}$ denotes the response, and the variables have a general relationship of the form $Y = f(\boldsymbol{X}) + \epsilon$. Assume further that the errors $\epsilon_1, ..., \epsilon_n$ are uncorrelated with mean 0 and (common) variance σ^2. Given a particular regression estimate $\hat{f}$ that produces fitted values $\hat{y}_1, ..., \hat{y}_n$, the degrees of

freedom [Efron, 1986, Efron and Tibshirani, 1990, Tibshirani, 2015] of $\hat{f}$ is defined as

$$df(\hat{f}) = \frac{1}{\sigma^2} \sum_{i=1}^{n} \text{Cov}(\hat{y}_i, y_i). \tag{2.3}$$

The degrees of freedom (dof) of a particular estimator is generally understood as a measure of its flexibility; estimators with high dof depend more heavily on the particular values observed in the original data and hence are higher variance. Understanding the dof associated with various estimation procedures can provide valuable insights into their behavior as well as the situations when they might be expected to perform better or worse relative to a set of alternative methods. Tibshirani [2015] took an important step in this regard, showing that adaptive procedures like best subset selection (BSS) and forward stepwise selection (FS), even when selecting a model with k terms, had more than k dof because of the increased dependence on the data incurred through the selection aspect. These additional dof were coined the *search* degrees of freedom. More recently, Hastie et al. [2020] provided a thorough collection of simulations to demonstrate the predictive advantages of regularized procedures like the lasso and relaxed lasso over BSS and FS, especially in low signal-to-noise ratio (SNR) settings where the SNR is defined as

$$\text{SNR} = \frac{\text{Var}(f(\boldsymbol{x}))}{\text{Var}(\epsilon)}.$$

Much of the work in the following sections was inspired by the approach taken in Hastie et al. [2020] and various portions of the work below follow closely to the setups considered there.

We begin our work by estimating the dof of random forests under various values of `mtry`. In linear model contexts, the dof for different estimators is generally shown by plotting the estimated dof against the average number of nonzero coefficients in the selected models. In our context with tree-based estimators, we use `maxnodes` – the maximum number of terminal nodes that any tree within the forest can have – as an analogue. For any given forest with fixed value of `mtry`, we should expect that as trees are allowed to grow deeper (i.e. `maxnodes` takes larger values), the estimators should become more sensitive to the data and hence incur higher dof.

We consider two general model forms: a linear model

$$Y = X\beta + \epsilon = X_1\beta_1 + \cdots + X_p\beta_p + \epsilon$$

and the model

$$Y = 0.1e^{4X_1} + \frac{4}{1 + e^{-20(X_2 - 0.5)}} + 3X_3 + 2X_4 + X_5 + \epsilon\,,$$

which we refer to as 'MARSadd' as it is additive and first appeared in the work on Multivariate Adaptive Regression Splines (MARS) by Friedman [1991]. Features in the MARSadd model are sampled independently from Unif$(0, 1)$. For the linear model, in line with Hastie et al. [2020], we consider three different settings:

- **Low**: $n = 100$, $p = 10$, $s = 5$
- **Medium**: $n = 500$, $p = 100$, $s = 5$
- **High-10**: $n = 100$, $p = 1000$, $s = 10$

where n is the total (training) sample size, p denotes the total number of features generated, and $s \leq p$ is the number of features with a nonzero coefficient thus considered signal. Rows of $X \in \mathbb{R}^{n\times p}$ are independently drawn from $N(0, \Sigma)$, where $\Sigma \in \mathbb{R}^{p\times p}$ has entry $(i, j) = \rho^{|i-j|}$. We take $\rho = 0.35$ and set the first s components of β equal to 1 with the rest set to 0. This setup corresponds to the general sampling scheme and 'beta-type 2' setting from Hastie et al. [2020]. For both models here as well as throughout the majority of the remainder of this chapter, we consider sampling the noise as $\epsilon \sim N(0, \sigma^2 I)$ where σ^2 is chosen to produce a corresponding SNR level ν, so that, for example, in the linear model case, we take

$$\sigma^2 = \frac{\beta^T \Sigma \beta}{\nu}.$$

Finally, in most previous literature, $\texttt{mtry} \leq p \in \mathbb{Z}^+$ denotes the number of features eligible for splitting at each node. Here and throughout the remainder of the chapter, we adopt a slightly different (but equivalent) convention by defining `mtry` as the *proportion* of eligible features so that $\texttt{mtry} \in (0, 1]$. This is purely for readability and ease of interpretation so that readers may more immediately see whether the number of available features is large or small (relative to p) regardless of the particular model setup being considered.

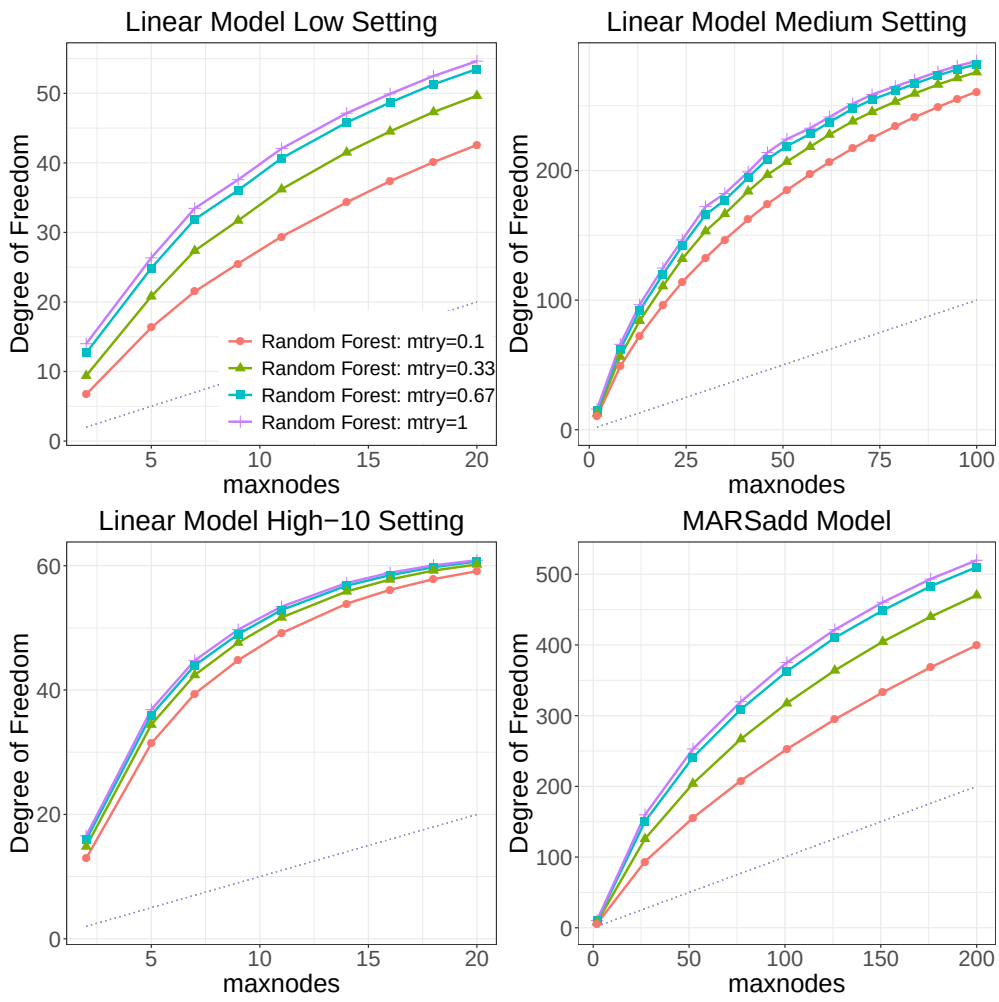

Figure 2.3: Degrees of freedom for random forests at different levels of `mtry`.

Results are shown in Figure 2.3. In each of the four model setups we take the SNR to be equal to 3.52 and estimate the dof for random forests with `mtry` equal to 0.1, 0.33, 0.67, and 1. Note that when `mtry` $= 1$ the model reduces to bagging [Breiman, 1996] and `mtry` $= 0.33$ corresponds to the standard choice of $p/3$ eligible features at each node in regression settings, as is the default in most software. The forests are constructed using the `randomForest` package in `R` [Liaw et al., 2002] with the default settings for all arguments except for `mtry` and `maxnodes`. Each point in each plot in Figure 2.3 corresponds to a Monte Carlo estimate of the dof formula given in (2.3) evaluated over 500 trials.

Several clear patterns are apparent in Figure 2.3. First and perhaps most obviously, as conjectured above, in each case we see that the dof increases as `maxnodes` increases and trees can be grown to a greater depth. Each plot also shows the same general concave increasing shape for each forest. Furthermore, the estimated dof function for each forest lies above the diagonal (shown as a dotted line in each plot), supporting the general notion formalized in Tibshirani [2015] that adaptive procedures like the tree-based models employed here incur additional dof as a result of this search.

More importantly for our purposes, in each plot in Figure 2.3 we see that at every fixed level of `maxnodes`, the dof increases with `mtry`. In particular, bagging (`mtry` = 1) always contains more dof than the default implementation for random forest regression (`mtry` = 0.33). Finally, we note that the patterns seen in these plots also hold for numerous other regression functions and SNRs that were experimented with; Figure A1 in Appendix A shows the results of the same experiments above carried out at a much lower SNR of 0.09 and the findings are nearly identical.

2.4 RANDOM FOREST PERFORMANCE VS SIGNAL-TO-NOISE RATIO

The empirical results in the preceding section suggest that the `mtry` parameter in random forests is directly tied to its dof with larger values resulting in estimators with higher dof and more flexibility. Based on this intuition and the results for linear estimators provided in Hastie et al. [2020], we should therefore expect that random forests with smaller values of `mtry` to perform well in noisy settings while bagging should perform best – potentially even better than random forests – at high SNRs. We now investigate this more formally.

2.4.1 Relative Performance on Synthetic Data

We begin by comparing the relative performance of random forests and bagging on simulated data across a range of plausible SNRs. In addition to the linear model setup described in

the previous section, we now include the additional regression function

$$Y = 10\sin(\pi X_1 X_2) + 20(X_3 - 0.05)^2 + 10X_4 + 5X_5 + \epsilon$$

which we refer to as 'MARS' because like the additive model used previously, it first appeared in the MARS paper [Friedman, 1991], though note that unlike the previous model, it contains explicit interactions between the features. This particular MARS model has proven popular in random forest publications in recent years, appearing, for example, in Biau [2012] and Mentch and Hooker [2016]. In the simulation setups described below, we consider the medium setting for the linear model ($n = 500$, $p = 100$, $s = 5$) and for the MARS model, we take $p = s = 5$ and consider sample sizes of $n = 200, 500$, and 10000. Features for the linear model are generated in the same fashion as above and those in the MARS model are sampled independently from Unif$(0, 1)$.

As in the previous section, the variance σ^2 of the noise term is chosen so as to induce particular SNRs. Here, following Hastie et al. [2020], we consider 10 SNR values $\nu = 0.05, 0.09, 0.14, ..., 6.00$ equally spaced between 0.05 and 6 on the log scale. Forests are again constructed using the `randomForest` package with the default settings except in the case of bagging where the default value of `mtry` is changed so that all features are available at each split.

Here, in comparing the performance of "random forests" against "bagging", we stress that we are merely assessing the difference in predictive accuracies between forests constructed with `mtry` $= 0.33$ (traditional random forests) versus those built with `mtry` $= 1$ (bagging). To compare the relative performance for a fixed model setup with fixed SNR, we first generate a training dataset and then evaluate the mean squared error (MSE) for both bagging and random forests on an independently generated test dataset consisting of 1000 observations. This entire process is then repeated 500 times for each setting and we record the average difference in accuracy (Error(Bagg) − Error(RF)) across these repetitions.

Results are shown in Figure 2.4. Each plot shows the average difference in test errors versus the SNR; note that positive differences indicate that random forests (`mtry` $= 0.33$) are outperforming bagging (`mtry` $= 1$). In each of the four regression setups shown in Figure 2.4, the improvement in accuracy seen with random forests decreases as the SNR

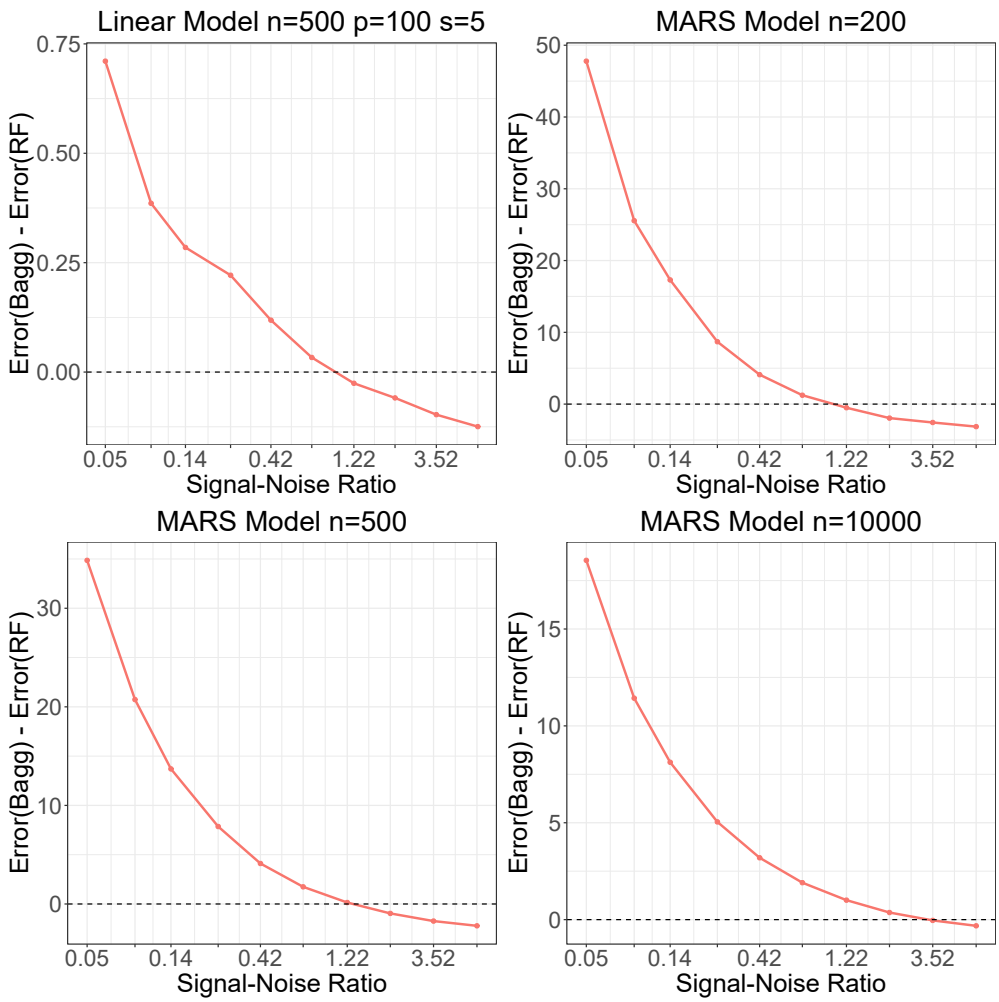

Figure 2.4: Differences in test errors between bagging and random forests. Positive values indicate better performance by random forests.

increases. For large SNR values, bagging eventually begins to outperform the traditional random forests. Note also that for the MARS function, random forests seem to retain their relative improvement longer (i.e. for larger SNRs) with larger training samples. Given these results, the conventional wisdom that random forests simply "are better" than bagging seems largely unfounded; rather, the optimal value of `mtry` seems to be a function of the SNR.

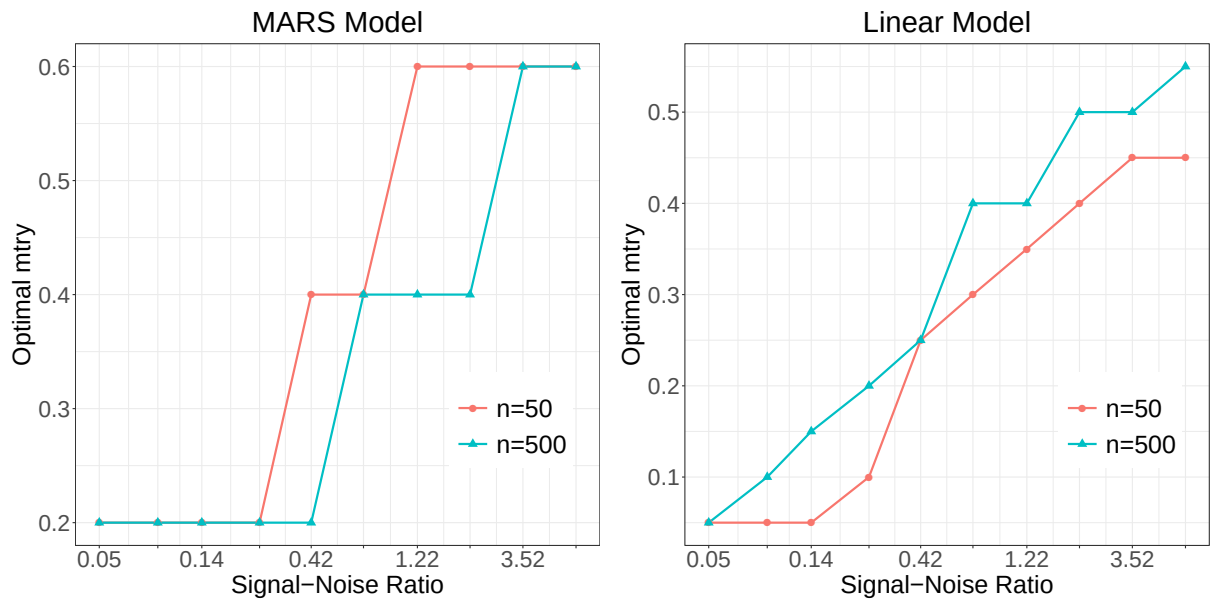

Figure 2.5: Optimal value of `mtry` vs SNR for the MARS and linear model.

2.4.2 Optimal `mtry` vs SNR

The results above indicate that random forests (`mtry` $= 0.33$) generally seem to outperform bagging (`mtry` $= 1$) unless the SNR is large. We now reverse the direction of this investigation and estimate the optimal value of `mtry` across various SNR levels.

Here, as above, we consider both the MARS model and a linear model. In the same fashion as in previous simulations, the errors are sampled from a $N(0, \sigma^2)$ where σ^2 is chosen to produce a particular SNR and we consider the same 10 SNR values as above. For the MARS model, we take $p = s = 5$ and generate features independently from Unif$(0, 1)$ and for the linear model, we take $p = 20$ and $s = 10$ with features drawn from $N_p(0, \Sigma)$ where the (i, j) entry of Σ is given by $\rho^{|i-j|}$ with $\rho = 0.35$. The first $s = 10$ coefficients in the linear model are set equal to 1 with the rest set equal to 0.

For both models, we consider (training) sample sizes of both $n = 50$ and $n = 500$ and generate an independent test set of the same size. We construct forests using all possible values of `mtry` with the remaining options at the default settings in `randomForest`. The

entire process is repeated 500 times and the `mtry` value corresponding to the forest with the lowest average test error for each setting is selected. The results are shown in Figure 2.5. As expected, corroborating the findings above, the optimal value of `mtry` increases with the SNR and the same general pattern emerges for both models and sample sizes. Figure A2 in Appendix A shows a slightly different calculation where the optimal `mtry` value on each of the 500 iterations is determined and the overall mean is then calculated. Here too we see exactly the same general pattern in that as the SNR increases, so does the optimal value of `mtry`.

2.4.3 Relative Performance on Real Data

The work above presents strong empirical evidence that the relative improvement in predictive accuracy seen with random forests is largest at low SNRs and more generally, that the optimal value of `mtry` appears to be a direct (increasing) function of the SNR. These results, however, pertain only to those particular simulation settings that some may argue are highly idealized. Real-world data may contain far more complex relationships and thus we now explore whether the same general findings above also appear in more natural contexts.

To investigate this, we utilize 10 datasets intended for regression from the UCI Machine Learning Repository [Dua and Graff, 2017]. Because most of these datasets are low-dimensional, five additional high-dimensional datasets were also included, four of which were downloaded from `openml.org` [Vanschoren et al., 2013] with the other (`AquaticTox`) taken from the `R` package `QSARdata`. Summaries of these datasets are provided in Table 2.1. For datasets containing missing values (`csm` and `fb`), the corresponding rows of data were removed. Here we do not know the true SNR and thus to compare the relative performance of bagging and random forests, we inject additional random noise ϵ into the response, where each $\epsilon \sim N(0, \sigma^2)$ and σ^2 is chosen as some proportion α of the variance of the original response variable. We consider $\alpha = 0, 0.01, 0.05, 0.1, 0.25$ and 0.5 where $\alpha = 0$ corresponds to the case where no additional noise is added and performance is thus compared on the original data. To compare performance, we measure the relative test error (RTE)

$$\text{RTE} = \frac{\widehat{Err}(\text{Bagg}) - \widehat{Err}(\text{RF})}{\hat{\sigma}_y^2} \times 100\% \tag{2.4}$$

Table 2.1: Summary of real-world data utilized. For datasets where no reference was specified, a reference to early work utilizing the data is given.

Dataset	p	n
Abalone Age [`abalone`] [Waugh, 1995]	8	4177
Bike Sharing [`bike`] [Fanaee-T and Gama, 2014]	11	731
Bioston Housing [`boston`] [Harrison Jr and Rubinfeld, 1978]	13	506
Concrete Compressive Strength [`concrete`] [Yeh, 1998]	8	1030
CPU Performance [`cpu`] [Ein-Dor and Feldmesser, 1987]	7	209
Conventional and Social Movie [`csm`] [Ahmed et al., 2015]	10	187
Facebook Metrics [`fb`] [Moro et al., 2016]	7	499
Parkinsons Telemonitoring [`parkinsons`] [Tsanas et al., 2009]	20	5875
Servo System [`servo`] [Quinlan, 1993]	4	167
Solar Flare [`solar`] [Li et al., 2000]	10	1066
Aquatic Toxicity [`AquaticTox`] [He and Jurs, 2005]	468	322
Molecular Descriptor Influencing Melting Point [`mtp2`] [Bergström et al., 2003]	1142	274
Weighted Holistic Invariant Molecular Descriptor [`pah`] [Todeschini et al., 1995]	112	80
Adrenergic Blocking Potencies [`phen`] [Cammarata, 1972]	110	22
PDGFR Inhibitor [`pdgfr`] [Guha and Jurs, 2004]	320	79

where $\widehat{Err}$(Bagg) and $\widehat{Err}$(RF) denote the 10-fold cross-validation error on bagging and random forests, respectively, and $\hat{\sigma}_y^2$ is the empirical variance of the original response. For each setting on each dataset, the process of adding additional random noise is replicated 500 times and the results are averaged. Once again, forests are constructed using the `randomForest` package with the default settings except for fixing `mtry` $= 1$ for bagging and `mtry` $= 0.33$ for random forests.

Results are shown in Figure 2.6 with low-dimensional datasets shown in the left plot and high-dimensional datasets shown on the right. Note that to aid in presentation, these display the *shifted* RTE rather than the raw calculation in (2.4). For a given proportion of additional noise α, let RTE(α) denote the corresponding relative test error. The shifted RTE at noise level α is then defined as RTE(α) - RTE(0). This ensures that the relative error for each dataset begins at the origin thereby allowing us to present all results in a single easy-to-interpret plot. Error bars correspond to ± 1 standard deviation across 500 trials.

In both plots in Figure 2.6, the same general pattern appears as has been seen in the subsections above: as we increase the amount of additional noise inserted into the models, the relative improvement in predictive accuracy seen with random forests becomes more

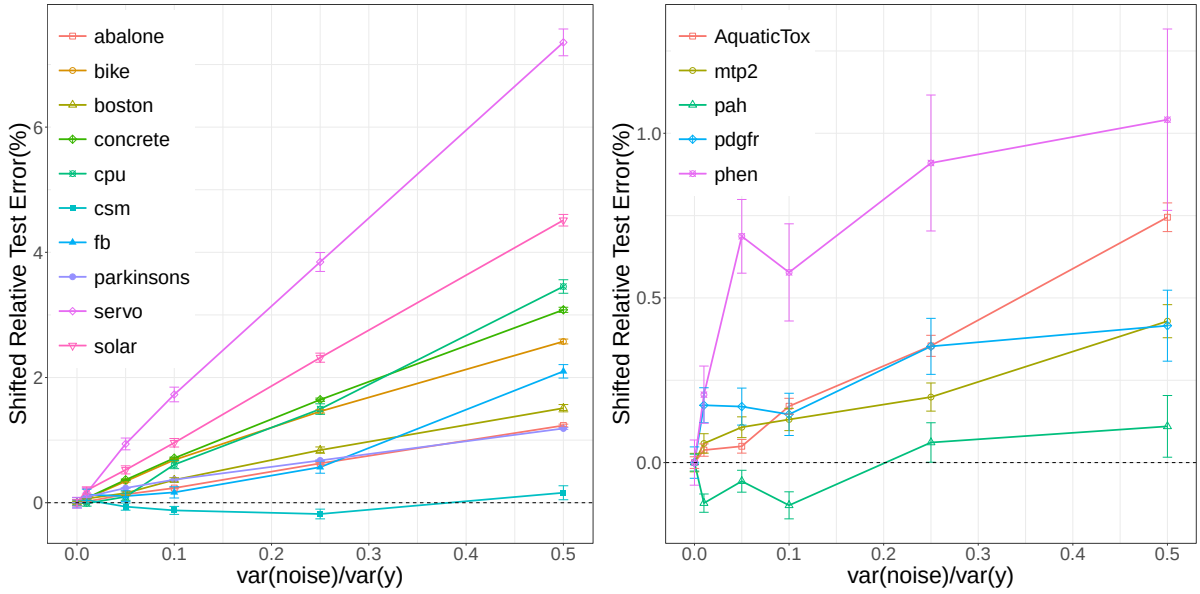

Figure 2.6: Shifted RTE on real data where additional noise is added. The left plot shows results on low-dimensional datasets taken from the UCI repository; the right plot shows results on high-dimensional datasets.

pronounced. The only slight exceptions are seen on the low-dimensional `csm` dataset and the high-dimensional `pah` dataset were the relative error appears to decrease by a small amount before eventually coming back up when large amounts of noise are added. It is not clear why the initial temporary drops occur in these two datasets, though it is worth noting that even the largest magnitudes of drops are quite small at approximately 0.2% and 0.13% in the `csm` and `pah` datasets, respectively.

2.5 RANDOMIZED FORWARD SELECTION

The results from the previous sections suggest that the optimal value of `mtry` for random forests is highly data-dependent, with smaller values preferred in noisy settings and bagging (`mtry` $= 1$) being preferred when very little noise is present. These findings are much in line

with the classic understanding of random forests as a means by which the variance of the ensemble is reduced as a by-product of reducing the correlation between trees. The benefits of this variance reduction are most apparent at low SNRs.

In our view, however, this remains only a partial explanation. While randomizing the collection of features eligible for splitting at each node is one way to reduce between-tree correlation, it is certainly not the only means by which this can be accomplished. Breiman [2001] experimented with alternative implementations finding that even very naive approaches like adding random noise to the outputs of each tree could sometimes be beneficial. But as discussed in the opening sections, Breiman [2001], like many others after, also noted that the particular approach where features are randomly precluded from splitting at each node seemed to produce substantially more accurate predictions than other strategies for reducing between-tree correlation. Why this was the case, however, was not clear and has remained a subject of speculation in the nearly two decades following the inception of the procedure.

As already briefly mentioned above, Hastie et al. [2020] recently provided an extended comparison of several variable selection procedures including the lasso [Tibshirani, 1996, Chen et al., 2001], relaxed lasso [Meinshausen, 2007], forward stepwise selection (FS), and best subset selection (BSS). The relaxed lasso estimator utilized in Hastie et al. [2020] takes the form

$$\hat{\beta}_{\text{relax}}(\lambda, \gamma) = \gamma \hat{\beta}_{\text{lasso}}(\lambda) + (1 - \gamma)\hat{\beta}_{\text{LS|lasso}}(\lambda)$$

where $\hat{\beta}_{\text{LS|lasso}}(\lambda)$ denotes the vector of coefficient estimates obtained via least squares (LS) when computed on only those variables selected via the lasso and filled in with 0 for variables not selected. Perhaps the most striking takeaway from their study is that in low-dimensional settings where $n > p$, the more aggressive procedures (FS and BSS) with higher dof are generally not competitive with a regularized approach like the lasso at low SNRs but eventually produce more accurate predictions when the SNR becomes large. Relaxed lasso, taking the weighted average of lasso and LS-after-lasso estimates, possesses the ability to effectively trade-off the amount of regularization needed depending on the SNR and always seems to perform well.

For these kinds of estimators whose inner-workings are better understood, the reasoning behind the results observed is relatively straightforward. In low SNR settings, procedures

like the lasso and relaxed lasso that explicitly regularize the problem can prevent overfitting to the noise by applying shrinkage to the coefficient estimates of the selected features. Given that we see the same general pattern here – random forests (`mtry` = 0.33) outperforming bagging (`mtry` = 1) except at high SNRs – it is reasonable to suspect that the additional randomness in random forests is playing a similar regularization role. Indeed, by randomly not allowing certain features to be split, random forests may be seen as effectively shrinking the potential influence of features, with the amount of shrinking being proportional to the amount of additional randomness added (with smaller values of `mtry` inducing more randomness). Speculation to this effect is described in Hastie et al. [2009].

Importantly however, if this is the kind of underlying effect that allows random forests to perform well in practice, such an effect should not be limited to tree-based ensembles. Indeed, if regression trees are seen as merely a complex form of forward selection, then if we were to create bagged and randomized versions of a standard forward selection procedure – analogues to the traditional tree-based versions of bagging and random forests – we should expect to see the same general patterns of relative improvement. In the following sections, we propose two such ensemble-ized extensions of forward selection and confirm that not only do similar patterns emerge, but that these new procedures exhibit surprisingly strong performance relative to alternative procedures.

2.5.1 Degrees of Freedom for Randomized Forward Selection

We begin by formalizing the notion of randomized forward selection (RandFS), which can be seen as a random forest analogue to traditional forward stepwise selection (FS). For any subset of the feature indices $\mathcal{S}$, define $\boldsymbol{X}_{\mathcal{S}}$ as the matrix of feature values whose index is in $\mathcal{S}$ and $P_{\mathcal{S}}$ as the projection matrix onto the column span of $\boldsymbol{X}_{\mathcal{S}}$. To carry out randomized forward selection (`RandFS`), we begin by drawing B bootstrap samples from the original data and performing forward selection on each. However, like random forests, at each step, only a random subset of remaining features are eligible to be included in the model. The process continues until the desired model size (depth) d is obtained and the final predictions are taken as an average over the predictions generated by each individual model. When

Algorithm 2 Randomized Forward Selection (`RandFS`)

procedure RANDFS($\mathcal{D}_n, B, d,$ `mtry`)

for $b = 1, \ldots, B$ **do**

Draw bootstrap sample $\mathcal{D}^{(b)} = \{(\boldsymbol{X}_i^{(b)}, Y_i^{(b)})\}_{i=1}^n$ from original data $\mathcal{D}_n$

Initialize empty active set $A_0 = \{0\}$

for $k \in 1 \ldots d$ **do**

Select subset of `mtry` $\times\, p$ features uniformly at random, denoted F_k

Select $j_k = \text{argmin}_{j \in F_k} \left\| Y^{(b)} - P_{A_{k-1} \cup \{j\}} Y^{(b)} \right\|_2^2$

Update active set $A_k = A_{k-1} \cup \{j_k\}$

Update coefficient estimates $\hat{\beta}^{(b)}$ as

$$\hat{\beta}^{(b)}_{A_k} = \text{argmin}_\beta \left\| Y^{(b)} - \boldsymbol{X}^{(b)}_{A_k} \beta \right\|^2, \quad \hat{\beta}^{(b)}_{A_k^c} = 0$$

Compute final coefficient estimates $\hat{\beta} = \frac{1}{B} \sum_{b=1}^B \hat{\beta}^{(b)}$

Compute predictions $\hat{Y} = X\hat{\beta}$

`mtry` $= 1$ so that all features are eligible at each step, we refer to the procedure as *bagged* forward selection (`BaggFS`). A summary of the procedure is given in Algorithm 2.

We begin by estimating the dof for RandFS as well as for FS, lasso, and relaxed lasso. Here we follow the same initial setup utilized in Hastie et al. [2020] where we assume a linear model $Y = X\beta + \epsilon$ with $n = 70$, $p = 30$, and $s = 5$. Rows of X are sampled independently from $N_p(0, \Sigma)$, where the $(i,j)^{th}$ entry of Σ takes the form $\rho^{|i-j|}$ with $\rho = 0.35$ and errors are sampled independently from $N(0, \sigma^2)$ with σ^2 chosen to satisfy a particular SNR, in this case 0.7. The first s components of β are set equal to 1 with the rest equal to 0.

The plots in Figure 2.7 show the estimated dof for the various methods against the number of nonzero coefficient estimates produced. Each point in each plot corresponds to a Monte Carlo estimate of the dof formula given in (2.3) evaluated over 500 iterations. As expected, the plot on the right shows quite clearly that as with random forests, larger values of `mtry` produce RandFS procedures with more dof. More surprising is the relative relationship of the RandFS models to the more classical procedures. The plot on the left

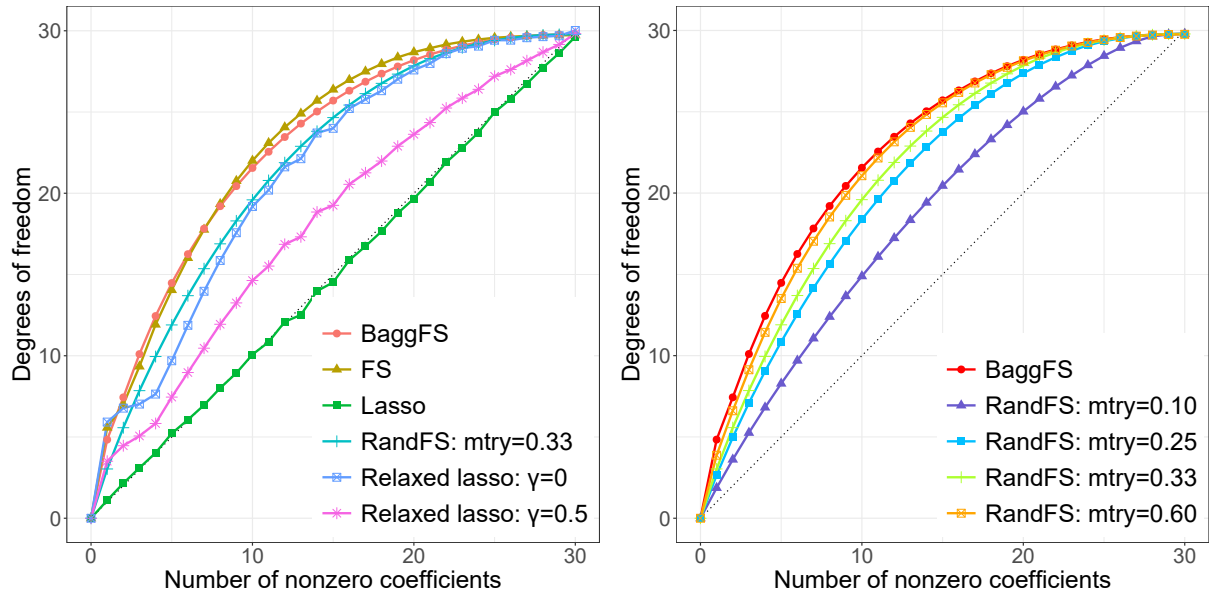

Figure 2.7: Estimated dof for forward selection (FS), bagged forward selection (BaggFS), randomized forward selection (RandFS), lasso, and relaxed lasso.

shows that the dof for BaggFS is almost identical to that of standard FS. The standard `mtry` value of 0.33 produces a RandFS model with dof very similar to that of the relaxed lasso with $\gamma = 0$, corresponding to least squares after lasso. Smaller values of `mtry` appear to offer even more regularization, producing dof similar to relaxed lasso with larger γ values, corresponding to an estimate where more weight is put on the original lasso coefficient estimates.

2.5.2 Relative Performance of Randomized Forward Selection

Building on the intuition from previous sections as well as that provided in Hastie et al. [2020], the dof results in Figure 2.7 suggest that we should expect to see RandFS models with small values of `mtry` have a potential advantage in predictive accuracy relative to FS and BaggFS at low SNRs. We now compare the performance of RandFS relative to BaggFS and the more classical procedures.

Here we consider several linear model setups taken directly from Hastie et al. [2020] and similar to those considered in Section 2.3. We consider four settings:

- **Low**: $n = 100$, $p = 10$, $s = 5$
- **Medium**: $n = 500$, $p = 100$, $s = 5$
- **High-5**: $n = 50$, $p = 1000$, $s = 5$
- **High-10**: $n = 100$, $p = 1000$, $s = 10$.

As above, rows of X are independently drawn from $N(0, \Sigma)$, where $\Sigma \in \mathbb{R}^{p \times p}$ has entry (i, j) $= \rho^{|i-j|}$ with $\rho = 0.35$ and where we set the first s components of β equal to 1 with the rest set to 0, corresponding to the beta-type 2 setting in Hastie et al. [2020]. Noise is once again sampled from $N(0, \sigma^2 I)$ where σ^2 is chosen to produce a corresponding SNR level ν and we consider the same 10 values $\nu = 0.05, 0.09, 0.14, ..., 6.00$ utilized above.

Tuning of the FS, lasso, and relaxed lasso procedures is done in exactly the same fashion as in Hastie et al. [2020]. In all cases, tuning parameters are optimized on an independent validation set of size n. For the low setting, the lasso shrinkage parameter λ follows the default `glmnet` settings being tuned across 50 values ranging from small to large fractions of $\lambda_{\max} = \| X^T Y \|_\infty$. For relaxed lasso, λ is tuned across the same 50 values and the γ parameter that weights the average of the lasso and LS-after-lasso estimates is chosen from 10 equally spaced values between 0 and 1. The depth of the models in FS, BaggFS, and RandFS are tuned across $d = 0, 1, 2, ..., 10$. Note that for BaggFS and RandFS, a selected depth of d means that each individual model is built to a depth of d and the final average is then taken; since different individual models will generally select different features, the final averaged model will generally contain more than d features. In addition to considering the default RandFS (`mtry` $= 0.33$) and BaggFS (`mtry` $= 1$), we also consider tuning the `mtry` in RandFS across 10 equally spaced values between 0.1 and 1. In the medium, high-5, and high-10 settings, the λ parameter in lasso and relaxed lasso is tuned across 100 values rather than 50 and model depths for FS, BaggFS, and RandFS are tuned across $d = 0, 1, 2, ..., 50$; all other setups remain the same.

To measure performance, we again follow the lead of Hastie et al. [2020] and calculate the test error relative to the Bayes error rate. Specifically, given a test point $z_0 = (x_0, y_0)$

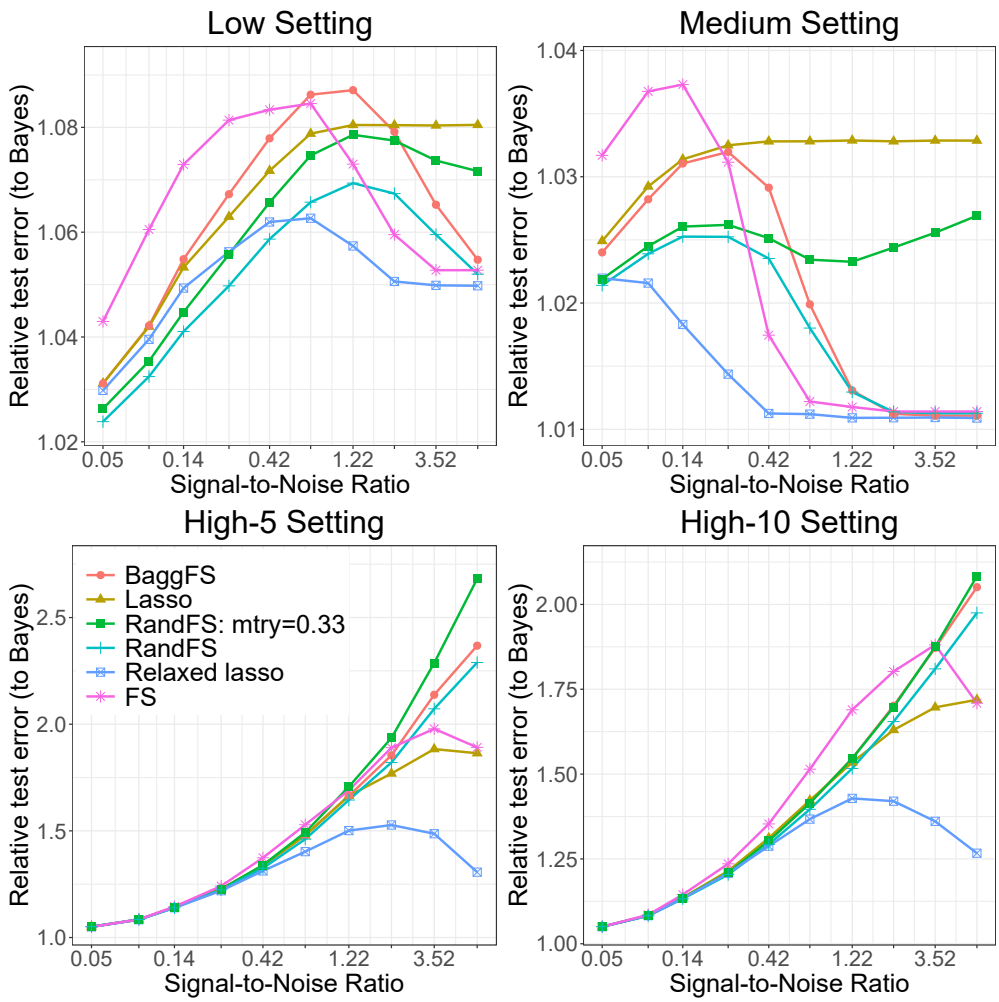

Figure 2.8: Performance of FS, BaggFS, RandFS, lasso and relaxed lasso across SNR levels for linear models in the low, medium, high-5, and high-10 settings.

with $y_0 = x_0^T\beta + \epsilon_0$ and $\epsilon_0 \sim N(0, \sigma^2)$, the relative test error (RTE) to Bayes of a regression estimate $\hat{\beta}$ is given by

$$\mathrm{RTE}(\hat{\beta}) = \frac{\mathbb{E}(y_0 - x_0^T\hat{\beta})^2}{\sigma^2} = \frac{(\hat{\beta}-\beta)^T\Sigma(\hat{\beta}-\beta)+\sigma^2}{\sigma^2}.$$

Results are shown in Figure 2.8; the same plots with error bars corresponding to ± 1 standard deviation are shown in Appendix A. Each point in each plot corresponds to an average over 100 replications. As expected, the explicit regularizers (lasso and relaxed lasso)

perform well in the high-dimensional settings and at low SNRs. In the high-dimensional settings in particular, most methods perform similarly at low SNRs but for larger SNRs, relaxed lasso begins to perform substantially better.

In the low and medium settings, BaggFS largely performs as expected with respect to classical FS. At low SNRs, the variance stabilization offered by BaggFS allows it to outperform FS but that advantage dies out at medium SNRs and both procedures appear similarly optimal relative to the others at high SNRs. It can also be seen in these settings that as originally hypothesized above, RandFS with fixed `mtry` $= 0.33$ outperforms BaggFS at low SNRs but loses the advantage at high SNRs.

The performance of RandFS in these settings with respect to the other procedures, however, is what is perhaps most surprising. Note that in the low setting, the RandFS procedure with tuned `mtry` outperforms all other methods – including lasso and relaxed lasso – until the mid-range SNR values. In the medium setting, RandFS exhibits a similar property to that of relaxed lasso, performing very well at low SNRs but adapting (likely selecting larger values of `mtry`) to also perform very well at large SNRs. Even more remarkable is the performance of RandFS when `mtry` $= 0.33$ and is not tuned – in both the low and medium settings, even this default RandFS outperforms the lasso across all SNRs.

2.5.3 Randomization as Implicit Shrinkage

Before concluding our work, we provide some additional intuition into the implicit regularization that appears to be taking place with the randomized ensembles (random forests and RandFS) studied in previous sections. This is more apparent and easily described in the more classical linear model forward selection setting with RandFS, though the same kind of effect is likely present with random forests, which might be seen as simply a more complex form of randomized forward selection.

As above, suppose we have data of the form $\mathcal{D}_n = \{Z_1, ..., Z_n\}$ where each $Z_i = (\boldsymbol{X}_i, Y_i)$, $\boldsymbol{X}_i = (X_{1,i}, ..., X_{p,i})$ denotes a vector of p features, $Y \in \mathbb{R}$ denotes the response, and the variables have a general relationship of the form $Y = f(X) + \epsilon$. Now suppose that we obtain a regression estimate $\hat{f}_{\text{RFS}} = X\hat{\beta}_{\text{RFS}}$ by averaging over B models, each built via randomized

forward selection on $\mathcal{D}_n$ to a depth of d. Note that this is identical to the RandFS procedure described above except that for technical reasons, models are built on the original data each time rather than on bootstrap samples. Each of the B individual models produces an estimate of the form

$$\hat{\beta}^{(b)}_{\text{RFS}} = \hat{\beta}^{(b)}_0 + X^{(b)}_{(1)}\hat{\beta}^{(b)}_{(1)} + \cdots + X^{(b)}_{(d)}\hat{\beta}^{(b)}_{(d)}$$

where $X^{(b)}_{(j)}$ is the feature selected at the j^{th} step in the b^{th} model and $\hat{\beta}^{(b)}_{(j)}$ is the corresponding coefficient estimate. Now consider a particular feature, say X_1, and suppose that the ordinary least squares (OLS) estimator exists and that the OLS coefficient estimate for X_1 is given by $\hat{\beta}_{1,\text{OLS}}$. More generally, given an active set $\mathcal{A} \subset \{1, ..., p\}$ containing a subset of feature indices, let $\hat{\beta}_{1,\text{OLS}|\mathcal{A}}$ denote the OLS estimate of the coefficient for X_1 when calculated over only the features with indices in $\mathcal{A}$.

Given an orthogonal design matrix, for any indexing set $\mathcal{A}$, $\hat{\beta}_{1,\text{OLS}|\mathcal{A}}$ is equal to $\hat{\beta}_{1,\text{OLS}}$ whenever $1 \in \mathcal{A}$ and equal to 0 otherwise. Thus, if $\mathcal{A}_b$ denotes the indices of those features selected for inclusion in the b^{th} model, then $\hat{\beta}^{(b)}_1 = \hat{\beta}_{1,\text{OLS}}$ if X_1 is selected (i.e. $1 \in \mathcal{A}_b$) and $\hat{\beta}^{(b)}_1 = 0$ otherwise. The final coefficient estimate produced by RandFS is thus of the form

$$\hat{\beta}_{1,\text{RFS}} = \frac{1}{B}\sum_{i=1}^{B} \beta^{(b)}_1 = \alpha_1 \cdot \hat{\beta}_{1,\text{OLS}} + (1 - \alpha_1) \cdot 0 = \alpha_1 \cdot \hat{\beta}_{1,\text{OLS}}$$

where $0 \leq \alpha_1 \leq 1$ denotes the proportion of models in which X_1 appears. In practice, for each feature X_k, the selection proportion α_k will depend on the particular data observed, the value of `mtry`, the depth d of each model, and the importance of X_k relative to the other features. In particular though, so long as d is relatively large, α_k should still be expected to be somewhat large even for moderately small values of `mtry` whenever X_k is a relatively important feature because it will have a very good chance of being included in the model if it is made eligible at any step. Thus, in this sense, not only does RandFS have a shrinkage effect on each variable, but variables that appear more important by virtue of being included in many models will be shrunken by less than those included only occasionally.

In the RandFS setup, this varying amount of shrinkage is a by-product of the adaptive, forward-selection nature in which the models are fit. We now show that when the adaptivity is removed and the linear sub-models are instead fit via standard OLS, the resulting shrinkage

becomes more uniform. In a very recent study released almost simultaneously to this work, LeJeune et al. [2020] demonstrate similar results for these kinds of OLS ensembles; we encourage interested readers to see this work for further results.

Assuming i.i.d. data of the form above, suppose that the true relationship between the features and response is given by

$$Y_i = \boldsymbol{X}_i'\boldsymbol{\beta} + \epsilon_i \tag{2.5}$$

where $\boldsymbol{\epsilon} = (\epsilon_1, ..., \epsilon_n)$ are i.i.d. and independent of the original data and each noise term ϵ_i has mean 0 and variance σ_ϵ^2. Denote $\boldsymbol{Y} = [Y_i]_{i=1}^n \in \mathbb{R}^n$ and $\boldsymbol{X} = [\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n]' \in \mathbb{R}^{n\times p}$. Assume $p < n$ so that the OLS estimator on the original data exists. Now suppose that we form a regression estimate $\hat{\boldsymbol{\beta}}^{ens}$ by averaging across B different OLS models, each built using only a subset of $m < p$ features selected uniformly at random. The following result gives that this average of OLS estimators is equivalent to ridge regression with shrinkage penalty $\lambda = \frac{p-m}{m}$.

Theorem 1. *Under the data setup given above, assume that $n > p$ and the design matrix $\boldsymbol{X}$ is orthogonal. Then*

$$\hat{\boldsymbol{\beta}}^{ens} \xrightarrow{B\to\infty} \frac{m}{p}\hat{\boldsymbol{\beta}}^{OLS}$$

where $\hat{\boldsymbol{\beta}}^{ens}$ denotes the estimate formed by averaging across B different OLS models, each built using only a subset of $m < p$ features selected uniformly at random, and $\hat{\boldsymbol{\beta}}^{OLS}$ denotes the standard OLS estimate on the original data.

Proof: For $b = 1 \ldots B$, let $S_b \subseteq \{1, \ldots, p\} = [p]$ denote the set of indices of the m features selected in the b^{th} model. Let $\boldsymbol{S}_b$ be the $p \times m$ subsampling matrix obtained by selecting the columns from I_p corresponding to the indices in S_b. The b^{th} model estimate is given by

$$\hat{\boldsymbol{\beta}}^{(b)} = \boldsymbol{S}_b \left(\boldsymbol{S}_b'\boldsymbol{X}'\boldsymbol{X}\boldsymbol{S}_b\right)^{-1}\boldsymbol{S}_b'\boldsymbol{X}'\boldsymbol{Y}$$

which, by orthogonality of $\boldsymbol{X}$, can be written as

$$\hat{\boldsymbol{\beta}}^{(b)} = \boldsymbol{S}_b\boldsymbol{S}_b'\boldsymbol{X}'\boldsymbol{Y} = \boldsymbol{S}_b\boldsymbol{S}_b'(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y} = \boldsymbol{S}_b\boldsymbol{S}_b'\hat{\boldsymbol{\beta}}^{OLS}.$$

Averaging across the B individual models gives

$$\hat{\boldsymbol{\beta}}^{ens} = \frac{1}{B}\sum_{b=1}^{B}\hat{\boldsymbol{\beta}}^{(b)} = \frac{1}{B}\sum_{b=1}^{B}\boldsymbol{S}_b\boldsymbol{S}_b'\hat{\boldsymbol{\beta}}^{OLS}. \tag{2.6}$$

Finally, let $\boldsymbol{C}$ denote the $p \times p$ diagonal matrix where $\boldsymbol{C}_{jj}$ is the number of times that the j^{th} feature is selected in the B base models. Then we have

$$\hat{\boldsymbol{\beta}}^{ens} = \frac{1}{B}\boldsymbol{C}\hat{\boldsymbol{\beta}}^{OLS} \xrightarrow{B\to\infty} \frac{m}{p}\hat{\boldsymbol{\beta}}^{OLS}.$$

■

The above result explicitly shows the shrinkage that occurs when an ensemble estimate is formed by averaging across B models, each of which uses only a randomly selected subset of the available features. We now demonstrate that when base models are constructed by subsampling both features and observations, a similar result holds in expectation. LeJeune et al. [2020] showed that the optimal risk of such estimators is equivalent to that of the optimal ridge estimator. Here we follow the same setup to explicitly examine the expectation of this kind of estimator.

Assume that the rows of $\boldsymbol{X}$ are i.i.d. with mean $\mathbf{0}$ and variance I_p and consider an ensemble of B total OLS base models constructed as follows. For $b = 1 \dots B$, subsample m features uniformly at random and let $S_b \subseteq \{1, \dots, p\} = [p]$ denote the set of indices corresponding to the features selected in the b^{th} model. Let $\boldsymbol{S}_b$ denote the $p \times m$ subsampling matrix obtained by selecting the columns from I_p corresponding to the indices in S_b. Similarly, subsample t observations and let $T_b \subseteq [n]$ denote the set of the indices of observations selected in the b^{th} model and let $\boldsymbol{T}_b$ denote the $n \times t$ subsampling matrix obtained by selecting the columns from I_n corresponding to the indices in T_b. Let $\mathcal{S}$ and $\mathcal{T}$ denote the collections of all possible S_b and T_b, respectively. Finally, assume that all subsampling is carried out independently and define $\hat{\boldsymbol{\beta}}^{ens,ss}$ as the ensemble estimate formed by averaging across the B subsampled OLS models. The following result gives that on average, $\hat{\boldsymbol{\beta}}^{ens,ss}$ produces an estimate equal to the true coefficient $\boldsymbol{\beta}$ shrunk by a factor of $\frac{m}{p}$.

Theorem 2. *Under the setup given above, assume that $m < t - 1$ and $Y_i = \boldsymbol{X}_i'\boldsymbol{\beta} + \epsilon_i$ as in (2.5). Then*

$$\mathbb{E}\left(\hat{\boldsymbol{\beta}}^{ens,ss}\right) = \frac{m}{p}\boldsymbol{\beta} = \frac{1}{1 + \frac{p-m}{m}}\boldsymbol{\beta}.$$

Proof: For $b = 1 \dots B$, LeJeune et al. [2020] showed that each individual model estimate can be written as

$$\hat{\boldsymbol{\beta}}^{(b)} = \boldsymbol{S}_b \left(\boldsymbol{T}_b'\boldsymbol{X}\boldsymbol{S}_b\right)^+ \boldsymbol{T}_b'\boldsymbol{Y}$$

where $(\cdot)^+$ denotes the Moore-Penrose pseudoinverse. The ensemble estimate is thus given by

$$\hat{\boldsymbol{\beta}}^{ens,ss} = \frac{1}{B}\sum_{b=1}^{B} \hat{\boldsymbol{\beta}}^{(b)} = \frac{1}{B}\sum_{b=1}^{B} \boldsymbol{S}_b \left(\boldsymbol{T}_b'\boldsymbol{X}\boldsymbol{S}_b\right)^+ \boldsymbol{T}_b'\boldsymbol{Y}. \tag{2.7}$$

Now, given the assumed linear relationship between Y_i and $\boldsymbol{X}_i$, the expectation of $\hat{\boldsymbol{\beta}}^{ens,ss}$ with respect to $\boldsymbol{\epsilon}$ but conditional on $\boldsymbol{X}$, $\mathcal{S}$, and $\mathcal{T}$, is given by

$$\mathbb{E}_{\boldsymbol{\epsilon}}\left(\hat{\boldsymbol{\beta}}^{ens,ss}\right) = \frac{1}{B}\sum_{b=1}^{B} \boldsymbol{S}_b \left(\boldsymbol{T}_b'\boldsymbol{X}\boldsymbol{S}_b\right)^+ \boldsymbol{T}_b'\boldsymbol{X}\boldsymbol{\beta}.$$

Applying a further result from LeJeune et al. [2020] showing that $\boldsymbol{S}_b\boldsymbol{S}_b' + \boldsymbol{S}_b^c\boldsymbol{S}_b^{c\prime} = I_p$, we have

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\epsilon}}\left(\hat{\boldsymbol{\beta}}^{ens,ss}\right) &= \frac{1}{B}\sum_{b=1}^{B} \boldsymbol{S}_b \left(\boldsymbol{T}_b'\boldsymbol{X}\boldsymbol{S}_b\right)^+ \boldsymbol{T}_b'\boldsymbol{X}\left(\boldsymbol{S}_b\boldsymbol{S}_b' + \boldsymbol{S}_b^c\boldsymbol{S}_b^{c\prime}\right)\boldsymbol{\beta} \\
&= \frac{1}{B}\sum_{b=1}^{B} \left(\boldsymbol{S}_b \left(\boldsymbol{T}_b'\boldsymbol{X}\boldsymbol{S}_b\right)^+ \boldsymbol{T}_b'\boldsymbol{X}\boldsymbol{S}_b\boldsymbol{S}_b'\boldsymbol{\beta} + \boldsymbol{S}_b \left(\boldsymbol{T}_b'\boldsymbol{X}\boldsymbol{S}_b\right)^+ \boldsymbol{T}_b'\boldsymbol{X}\boldsymbol{S}_b^c\boldsymbol{S}_b^{c\prime}\boldsymbol{\beta}\right) \\
&= \frac{1}{B}\sum_{b=1}^{B} \left(\boldsymbol{S}_b\boldsymbol{S}_b'\boldsymbol{\beta} + \boldsymbol{S}_b \left(\boldsymbol{T}_b'\boldsymbol{X}\boldsymbol{S}_b\right)^+ \boldsymbol{T}_b'\boldsymbol{X}\boldsymbol{S}_b^c\boldsymbol{S}_b^{c\prime}\boldsymbol{\beta}\right).
\end{aligned}$$

By the assumption that $\mathbb{E}(\boldsymbol{X}_i) = \boldsymbol{0}$ and $\mathrm{Var}(\boldsymbol{X}_i) = I_p$, $\boldsymbol{X}\boldsymbol{S}_b$ and $\boldsymbol{X}\boldsymbol{S}_b^c$ are independent with mean $\boldsymbol{0}$. Thus, the expectation of $\hat{\boldsymbol{\beta}}^{ens,ss}$ with respect to the training data $\boldsymbol{\epsilon}$ and $\boldsymbol{X}$ and conditional on $\mathcal{S}$ and $\mathcal{T}$ is given by

$$\mathbb{E}_{\boldsymbol{\epsilon},\boldsymbol{X}}\left(\hat{\boldsymbol{\beta}}^{ens,ss}\right) = \frac{1}{B}\sum_{b=1}^{B} \boldsymbol{S}_b\boldsymbol{S}_b'\boldsymbol{\beta} = \frac{1}{B}\boldsymbol{C}\boldsymbol{\beta} \tag{2.8}$$

where $\boldsymbol{C}$ is a diagonal matrix with $\boldsymbol{C}_{jj}$ equaling the number of times that the j^{th} feature is selected for $j = 1, ..., p$. Since both the features and observations are subsampled uniformly at random, we have

$$\mathbb{E}_{\boldsymbol{\epsilon},\boldsymbol{X},\mathcal{S},\mathcal{T}}\left(\hat{\boldsymbol{\beta}}^{ens,ss}\right) = \frac{m}{p}\boldsymbol{\beta} = \frac{1}{1+\frac{p-m}{m}}\boldsymbol{\beta}$$

as desired. ■

Note that in both theorems above, the ensemble estimates are shrunk by a factor of m/p and since features are selected uniformly at random (u.a.r.) for each model, this corresponds simply to the probability of each feature being chosen in each model. As discussed initially, in the RandFS framework where each base model is constructed in an adaptive fashion, the amount of shrinkage applied to each feature will be affected by its relative importance in predicting the response. Indeed, notice from the proofs of both theorems above that if features were selected in any non-u.a.r. so that some features were more likely to be selected than others, then so long as those selection probabilities were independent of the original data, only the final lines in the proofs would need changed. In particular, for features more likely to be selected, the expected corresponding diagonal entry of the $\boldsymbol{C}$ matrix would be larger, resulting in less shrinkage on the corresponding coefficient.

2.6 DISCUSSION

The results in the previous sections provide substantial evidence that the `mtry` parameter – the distinguishing feature of random forests – has an implicit regularization effect on the procedure. Much like the tuning parameter λ in explicit regularization procedures like lasso and ridge regression, `mtry` serves to mitigate overfitting to particular features. Thus, contrary to conventional wisdom, random forests are not simply "better" than bagging, but rather, the relative success of random forests depends heavily on the amount of noise present in the problem at hand. At low to moderate SNRs, random forests are seen to perform substantially better than bagging whereas bagging becomes far more competitive in high SNR regimes. This suggests further that at least in regression contexts, the success of random forests

is not in fact due to any kind of potential interpolation as was recently hypothesized in Wyner et al. [2017]. In Section 2.5 we showed that the same kinds of patterns emerged for ensemble-ized extensions of classic forward selection. Our findings suggest that these modified forward selection procedures intended as bagging and random forest analogues – BaggFS and RandFS, respectively – may sometimes provide a substantial improvement over standard forward selection, especially at low SNRs.

The obvious question is then, "Why do random forests *appear* to work so well in practice on real datasets?" As discussed in the introduction, there is certainly considerable evidence that random forests do often perform very well in a variety of settings across numerous scientific domains. In our view, the clear answer is that in practice, many datasets simply *are* quite noisy. Hastie et al. [2020] provide a thorough discussion along these lines, arguing that although commonly employed in simulations, on real data, SNRs as large as 5 or 6 are extremely rare. If this is indeed the case, it would explain why random forests are so often viewed as inherently superior to bagging.

Finally, we note that our findings are very much in line both with previous findings and with common practice. In practical applications, random forests are generally used "off-the-shelf" without any tuning of the `mtry` parameter. The plots corresponding to the low and medium settings in Figure 2.8 may shed some light on this: though not always optimal, the procedure with a fixed `mtry` value of 0.33 generally performs quite well and thus, in terms of practical guidance, this default value seems to be as good a starting place as any. On the other hand, the results provided above do strongly support the notion discussed in many previous studies [Díaz-Uriarte and De Andres, 2006, Genuer et al., 2008, Bernard et al., 2009, Genuer et al., 2010, Probst et al., 2019b] that the `mtry` parameter can significantly influence performance. Just as with explicit regularization procedures, the results above suggest that the `mtry` parameter in random forests can be thought of as controlling the amount of shrinkage and regularization and thus is best tuned in practice. For large datasets where tuning could introduce a computational burden, the recent results in Lopes et al. [2019b] suggest that at least in some cases, the algorithmic variance may quickly diminish after relatively few bootstrap samples and thus the tuning and validation could potentially be done reasonably well even on relatively small ensembles.

3.0 NOISE FEATURES AS REGULARIZATION: AUGMENTED BAGGING AND VARIABLE IMPORTANCE

In this chapter, we explore the impact of other forms of regularization on ensembles of trees. In particular, expanding the feature space with extra noise features leads to improved performance of bagging. Apart from the fact that this result in and of itself is surprising and counter-intuitive, it leads to reflections on an appropriate measure of variable importance, which is of interest across all of sciences. The following sections are mainly pulled from Mentch and Zhou [2020a].

3.1 INTRODUCTION

In the last chapter, we have demonstrated that the additional randomness utilized in random forests was simply an implicit form of regularization. The `mtry` parameter in random forests that dictates the number of available features at each split could therefore been seen as akin to the λ shrinkage penalty in explicit regularization methods like ridge regression [Hoerl and Kennard, 1970] and lasso [Tibshirani, 1996]. The random subsampling of features helped the trees to avoid overfitting and that this was particularly beneficial in low signal-to-noise ratio settings. LeJeune et al. [2020] demonstrated a similar effect for ensembles consisting of linear model base learners fit via ordinary least squares (OLS).

The idea that the randomness in random forests serves as a means of regularization not only eliminates some of the mystery of their sustained success but also suggests that alternative modifications to the standard bagging procedure that also induce some means of regularization may produce similar gains in accuracy. In this work, we introduce one such

alternative idea we refer to as *augmented bagging* (AugBagg) wherein the original feature space is augmented with additional noise features generated conditionally independent of the response, after which the standard bagging procedure is carried out. Very recent work by Kobak et al. [2020] showed that under certain conditions, including particular forms of additional random noise features in the regression can also improve the performance of linear models. As a result, performing minimum-norm least squares on an augmented design with increasingly many features each with increasingly small variance can be seen as equivalent to ridge regression on the original design.

The work in this chapter in the context of bagging and random forests uncovers findings that are arguably even more surprising and troubling. First, unlike the somewhat strict requirements in Kobak et al. [2020], the presence of additional noise features seems to often help regularize the model regardless of their individual variance or dependence on each other. Most alarmingly, in many instances, we show that this simple act of adding extra random noise features to the model can greatly *improve* its out-of-sample predictive accuracy *over even the most optimally tuned model on the original design*. Rather than making a bad model worse as most would presume, the addition of otherwise useless random noise features can have precisely the opposite effect.

This finding has crucial implications for the ways in which we measure and test feature importance. In black-box contexts where traditional measures like p-values may be unavailable or difficult to obtain, numerous recent studies have formally proposed methods to evaluate feature importance by measuring the change in accuracy when the features of interest are dropped from the model [Mentch and Hooker, 2016, 2017, Lei et al., 2018, Coleman et al., 2019, Williamson et al., 2021]. The implicit logic in such procedures feels intuitive and obvious: if the response can be more accurately predicted when a supplemental collection of features are included in the model, then those additional features must hold some information about the response beyond whatever is offered by the original collection of features. This work, however, demonstrates that this need not be the case. Rather, in some instances, particularly when the data itself are quite noisy, independent random noise features can improve predictions when added to a model. This can thus lead to situations that feel almost paradoxical in which, depending on the assumptions made and the type of

test deployed, noise features that are completely independent of the response may routinely register as statistically significant. Much further discussion on the implications of this finding is included in the latter sections of this work along with a proposed solution.

The remainder of this chapter is laid out as follows. In Section 3.2 we formally introduce the AugBagg procedure and in Section 3.3 we provide numerous simulations and real-data experiments to demonstrate its surprisingly competitive predictive performance. In Section 3.4 we provide theoretical motivation for the AugBagg procedure, building upon very recent results established for other learning procedures. Implications for measuring and testing variable importance are discussed in Section 3.5, where we also suggest a more robust alternative testing framework in which tests for feature importance maintain the nominal level for noise features, even when such features are capable of producing non-trivial gains in accuracy.

3.2 AUGMENTED BAGGING

Throughout the remainder of this chapter, we assume data of the form $\mathcal{D}_n = \{\boldsymbol{Z}_1, ..., \boldsymbol{Z}_n\}$ where each ordered pair $\boldsymbol{Z}_i = (\boldsymbol{X}_i, Y_i)$ consists of a feature vector $\boldsymbol{X}_i = (X_{1,i}, ..., X_{p,i})$ and response $Y_i \in \mathbb{R}$. Given B bootstrap samples of the data, the bagging procedure [Breiman, 1996] generates a prediction at $\boldsymbol{x}$ of the form

$$\hat{y}_{\text{Bagg}} = \frac{1}{B} \sum_{b=1}^{B} T(\boldsymbol{x};\, \omega_b, \mathcal{D}_n) \tag{3.1}$$

where the randomness ω_b serves only to select the bootstrap sample on which the b^{th} model T is trained. Whenever the randomness is assumed to select both the bootstrap sample as well as the $\texttt{mtry} < p$ eligible features at each internal node, it leads to the same equation for the random forest prediction $\hat{y}_{\text{RF}}$ as in Section 2.2.

The augmented bagging (AugBagg) procedure we introduce here represents a straightforward extension of classical bagging. Beginning with the original dataset $\mathcal{D}_n$, we create an augmented dataset $\mathcal{D}_n^*$ consisting of additional noise features generated conditionally independent of Y. This augmented dataset thus takes the form $\mathcal{D}_n^* = \{\boldsymbol{Z}_1^*, ..., \boldsymbol{Z}_n^*\}$ where each

$\boldsymbol{Z}_i^*$ now denotes an ordered triplet $(\boldsymbol{X}_i, \boldsymbol{N}_i, Y_i)$ consisting of the original features $\boldsymbol{X}_i$ and response Y_i, but also an additional set of noise features $\boldsymbol{N}_i = (N_{1,i}, ..., N_{q,i})$. The original bagging procedure is then performed on this augmented feature space so that the AugBagg output produces predictions of the form

$$\hat{y}_{\text{AugBagg}} = \frac{1}{B}\sum_{b=1}^{B} T((\boldsymbol{x}, \boldsymbol{n}); \omega_b, \mathcal{D}_n^*) \tag{3.2}$$

where $\boldsymbol{n}$ can be filled in with random draws from the additional noise features.

Importantly, we insist only that $\boldsymbol{N}$ be generated conditionally independent of Y given $\boldsymbol{X}$. This thus allows for additional noise features to be correlated with the original features. The noise features, however, are still sampled at random so that even if duplicate observations $\boldsymbol{x}_i = \boldsymbol{x}_j$ appear in the original data, it need not be the case that $(\boldsymbol{x}_i, \boldsymbol{n}_i) = (\boldsymbol{x}_j, \boldsymbol{n}_j)$. As demonstrated in the following sections, the manner in which noise features are generated can greatly impact performance.

Many of the simulations and experiments carried out below follow the classical definitions and settings of bagging and random forests in which base learners are assumed to be full-depth CART-style trees, as these are the kinds of models most frequently employed in practice and available by default in software. Note that whenever the randomness ω is assumed to select both the bootstrap sample as well as the $\texttt{mtry} < p$ eligible features at each internal node as in the case of random forests, the resulting prediction $\hat{y}_{\text{RF}}$ can be written in the same general form as (3.1). The invariance of CART-style trees to feature scaling presents an additional benefit here, as somewhat less precision is needed in generating the additional noise features for the augmented bagging procedure. We stress, however, that our findings to come are not tree-based and in particular, that the regularization effect offered via augmenting with noise features should be seen ultimately as a by-product of model averaging rather than the specific kinds of base learners that are utilized.

3.3 SIMULATIONS AND REAL DATA EXAMPLES

We now present a number of simulation studies to demonstrate the effectiveness of the AugBagg procedure in practice. To begin, we consider a standard linear model of the form $Y = \boldsymbol{X}\beta + \epsilon$ with $n \times p$ design matrix, the rows of which are i.i.d. multivariate normal $\mathcal{N}_p(\mathbf{0}, \Sigma)$ where $\Sigma \in \mathbb{R}^{n \times p}$ has entry $(i, j) = \rho^{|i-j|}$ with $\rho = 0.35$. The form of this covariance corresponds to that utilized frequently in Chapter 2 and to the 'beta-type 2' setup utilized in Hastie et al. [2020]. The original data includes $n = 100$ observations, $p = 5$ original signal features with $\beta_1 = \cdots = \beta_5 = 1$, and q additional i.i.d. noise features sampled from $\mathcal{N}(0, 1)$ independent of $\boldsymbol{X}$ are then added with q ranging from 1 to 250. As in Chapter 2 and Hastie et al. [2020], the noise term ϵ is sampled from $\mathcal{N}(0, \sigma_\epsilon^2)$ with σ_ϵ^2 chosen to satisfy a particular signal-to-noise ratio (SNR), given in this context by $\beta^T \Sigma \beta / \sigma_\epsilon^2$.

Figure 3.1 shows the performance of the AugBagg procedure where bagging is performed with unpruned trees utilizing both the original p signal features as well as the q additional noise features. Horizontal lines in the background of each plot correspond to random forests at different levels of `mtry` built using only the original $p = 5$ features. Each plot corresponds to a different SNR (0.01, 0.05, 0.09, or 0.14) and shows the relative test error, defined as the test MSE calculated on an independent, randomly generated test set of 1000 observations, scaled by σ_ϵ^2. Each point in each plot corresponds to the error averaged over 500 iterations with error bars showing ± 1 standard deviation. Note that in each case, the random forest error grows as `mtry` increases and so in particular, bagging on only the original 5 features (i.e. a random forest with `mtry` $= 5$) is the worst-performing model. At the lowest SNR of 0.01, however, *augmented* bagging appears to continually improve with q, easily surpassing even the best random forest once approximately $q = 25$ additional noise features are added into the model. Thus, the act of simply adding additional noise features into the model transforms the least accurate model (bagging, or, a random forest with `mtry` $= 5$) into one better than the best model built on the original data (a random forest with `mtry` $= 1$). The results are similar, though less dramatic, when the SNR is increased to 0.05. When the SNR is increased to 0.09, the performance of AugBagg appears to level-off around $q = 50$, never achieving that of the optimal random forest with `mtry` $= 1$. Finally, when the SNR is 0.14,

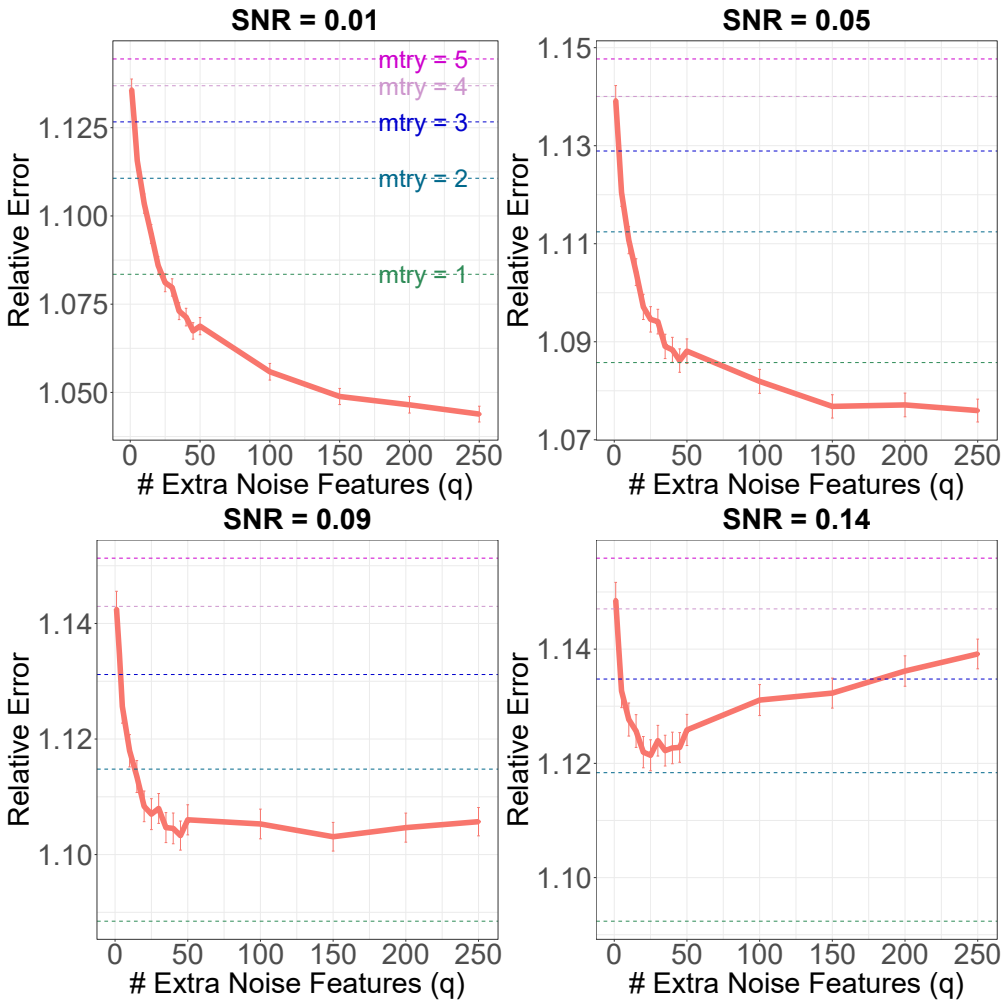

Figure 3.1: Performance of Augmented Bagging as q additional independent noise features are added to the model as compared with random forests and traditional bagging (`mtry` $= 5$) built on the original data. Each point in each plot corresponds to the average error after repeating the experiment 500 times with error bars showing ± 1 standard deviation.

the additional noise features appear to help until approximately $q = 50$, after which point the performance begins to deteriorate.

The results in Figure 3.2 expand these simulations. The data and model setup remain the same but the results are explored over a wider range of 8 SNRs, starting at 0.01 and then ranging from 0.05 to 2.07, equally spaced on the log scale. With the exception of the

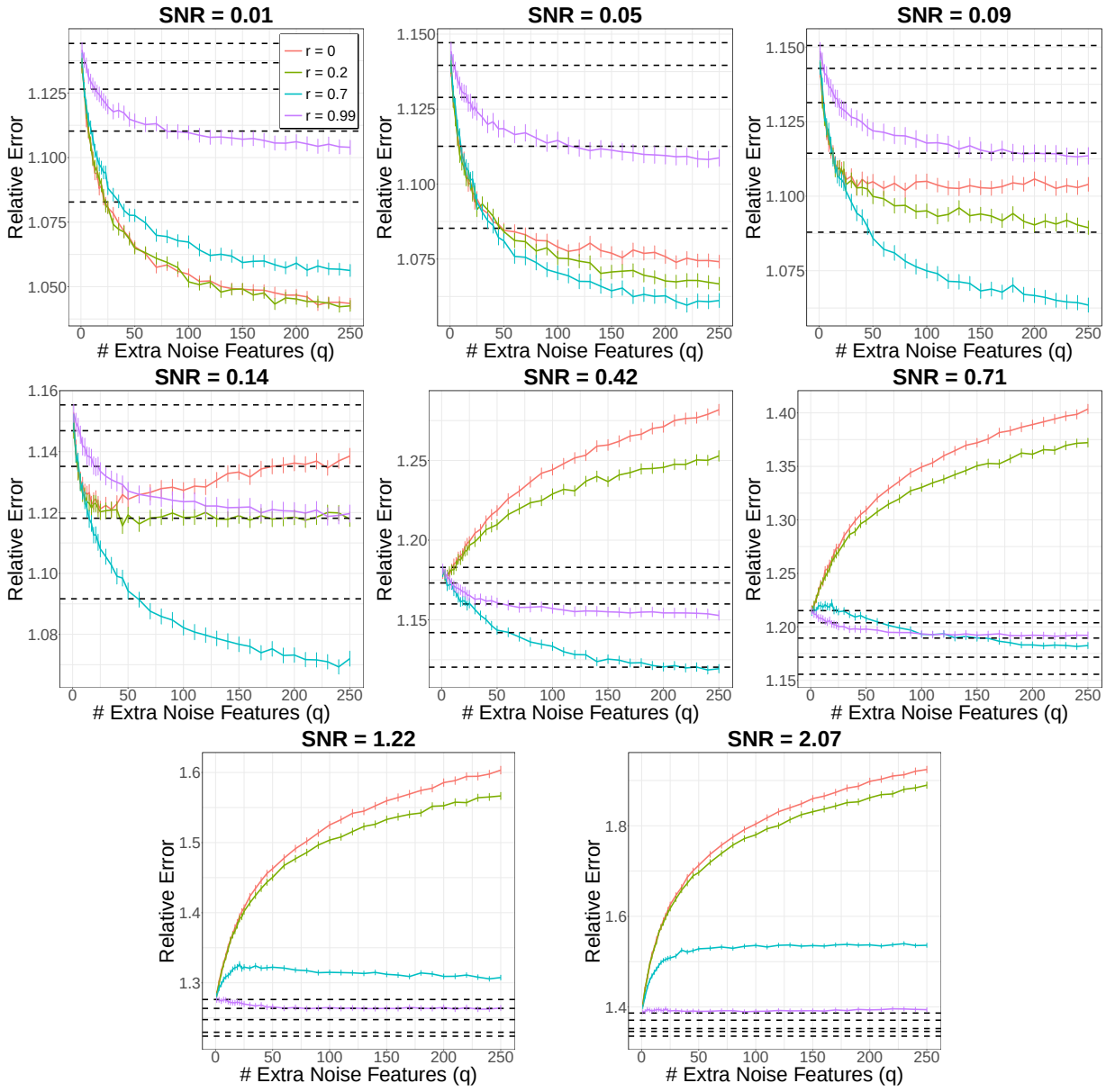

Figure 3.2: Performance of AugBagg compared against random forests as additional noise variables are added to the model. Different colored lines in each plot correspond to different correlation strengths between the original and noisy additional features. Each black horizontal dashed line in each plot corresponds to the performance of random forests at a fixed value of `mtry`.

lowest SNR of 0.01, the remaining sequence of SNRs is the same as was recently employed by Hastie et al. [2020] and correspond to a proportion of variance explained (PVE), defined as $\text{SNR}/(1+\text{SNR})$, of 0.01 on the low end (SNR=0.01) and 0.67 on the high end (SNR=2.07). In addition to the additional noise features sampled independently of $\boldsymbol{X}$, here we consider the addition of noisy features that are correlated with one of the first 5 signal features. In a similar fashion to knockoffs [Barber et al., 2015, Candes et al., 2018], such noise features are thus independent of the response Y given $\boldsymbol{X}$. To generate such features, we first select an original feature X at random and generate a standard normal $Z \sim \mathcal{N}(0,1)$. For a given level of correlation r, each of the additional q features then take the form

$$N = rX + \sqrt{(1-r^2)}Z. \tag{3.3}$$

In each of the plots in Figure 3.2 we consider correlations of $r = 0, 0.2, 0.7$, and 0.99 for batches of additional features ranging in size from 1 to 250. Performance is measured in the same fashion and estimates are averaged over 500 replications for each point in each plot. In the following discussion, we will use the shorthand $AB(q,r)$ to denote an AugBagg model with q additional noise features, each of which has correlation r with one of the features in the original dataset.

Figure 3.2 presents a very interesting and telling story in terms of how the additional noise features are influencing performance and how that influence changes across different SNR levels. Looking only at Figure 3.1 where the noise features are independent of both the response and the original features, one might suspect that this phenomenon occurs only at very low SNRs. Looking at Figure 3.2 however, we see that when the noise features are correlated with the original features, improvements in model accuracy are seen even at relatively high SNRs.

At the lowest SNRs of 0.01 and 0.05, we see that in every case, the AugBagg models are improving with the number of extra noise features q. Once $q > 50$, all AugBagg models begin to outperform even the best random forest, with the exception of $AB(q, 0.99)$ where very highly correlated noise features are added. At SNR = 0.09, much the same story is present but now only $AB(q, 0.7)$ outperforms the optimal random forest and again this transition happens around $q = 50$. At SNR = 0.14 we begin to see an interesting shift where the

performance of the independent noise model $AB(q, 0)$ begins to deteriorate with q. When the SNR grows to 0.42 and 0.71, this effect is much more pronounced with $AB(q, 0)$ and $AB(q, 0.2)$ both deteriorating with q. At the largest SNRs of 1.22 and 2.07, $AB(q, 0.99)$ is now the only model not deteriorating substantially with q.

3.3.1 Experiments on Real World Data

The previous simulations demonstrate that the AugBagg procedure can lead to substantial gains in accuracy over the baseline bagging procedure on synthetic datasets. Following a very similar setup to Chapter 2, we now investigate its performance on a variety of real-world datasets. The same datasets in 2.1 are used, except `parkinsons` due to high computation cost; a total of nine low-dimensional ($p < n$) and five high-dimensional ($p > n$) datasets are included.

In implementing the AugBagg procedure, we consider tuning both the number of additional noise features q as well as the level of correlation r. Since different datasets have different numbers of original features, q is tuned over $p/2$, p, $3p/2$ and $2p$. The correlation strength r is tuned over 0, 0.1, 0.4, 0.7 and 0.9. In datasets with mixed feature types, each additional noise feature is chosen to be correlated with one randomly selected continuous feature from the original data. As in 2, because the true SNR of real-world data is unknown, we inject further noise of the form $\epsilon \sim N(0, \sigma_\epsilon^2)$ into the response in order to observe trends in changes in model performance when the amount of noise grows larger relative to that in the original data. The variance of the noise σ_ϵ^2 is chosen as some proportion of $\hat{\sigma}_y^2$, the estimated variance of the original response Y. Performance is measured in terms of relative test error (RTE), defined as

$$\text{RTE} = \frac{\widehat{Err}(bagging) - \widehat{Err}(AugBag)}{\hat{\sigma}_y^2} \times 100\% \tag{3.4}$$

with positive values indicating superior performance by AugBagg. Here $\widehat{Err}$ is obtained via 10-fold cross validation.

Results are shown in Figure 3.3. In every case, the performance of the tuned AugBagg procedure increases as more noise is added to the response, as demonstrated by the positive

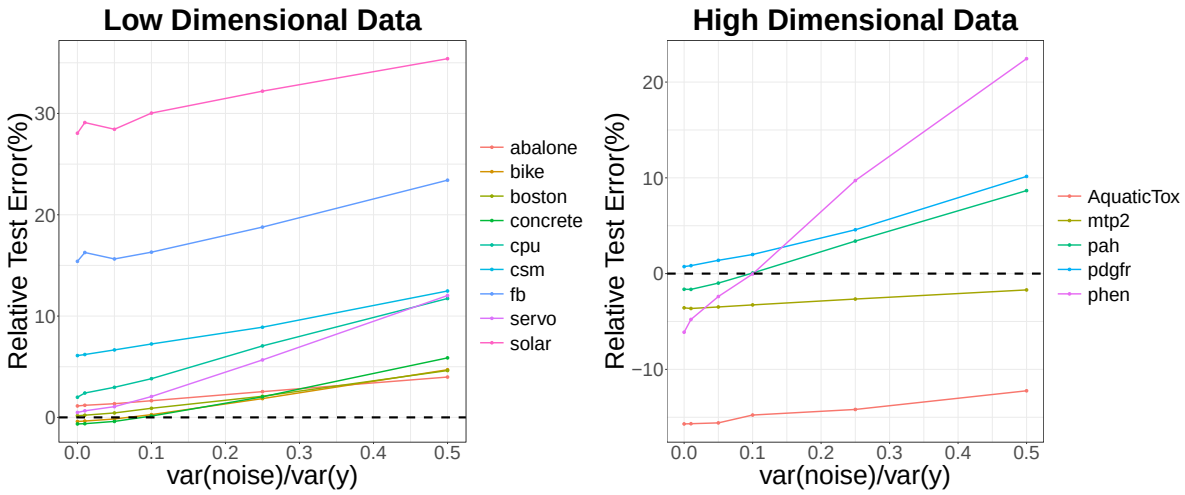

Figure 3.3: Relative test error (RTE) on real datasets with additional noise added onto the response. Left: low-dimensional datasets. Right: high-dimensional datasets.

slope displayed for each dataset. In 12 of the 14 datasets, AugBagg quickly begins to outperform bagging on the original data with substantial improvements occurring as more noise is injected. Furthermore, it is interesting to note that in the two cases where traditional bagging remains superior, both datasets (`AquaticTox` and `mtp2`) are high-dimensional and, in fact, contain the largest number of original features out of all datasets considered ($p = 468$ and 1142, respectively). In these cases, it is quite possible that many of the original features are themselves noisy and thus the additions we make are of no further benefit. Indeed, an optimally tuned lasso model built on the `AquaticTox` and `mtp2` datasets selects only (approximately) 17% and 4% of the features, respectively.

3.4 THEORETICAL MOTIVATION AND ANALOGOUS RESULTS

In the following three subsections, we draw upon recent results on interpolation and implicit regularization in order to provide some theoretical motivation for the practical success of the AugBagg procedure.

3.4.1 Randomization as Regularization

In Chapter 2, we argue that the success of random forests is due in large part to a kind of implicit regularization offered by the `mtry` parameter governing the number of features available for splitting at each node. Moreover, this behavior is not tree-specific, but holds for any ensemble consisting of forward-selection-style base learners in which the available features are randomly restricted at each step. Specifically, we consider RandFS in which the base model proceeds in the same fashion as a standard linear model forward selection process but only a randomly selected subset of the remaining features are eligible to be added to the model at each step.

Given an orthogonal design, discussions in Section 2.5.3 suggest that for any given feature X_j, the coefficient estimate given by RandFS is in the form of $\alpha_j \cdot \hat{\beta}_{j,OLS}$ where $\hat{\beta}_{j,OLS}$ is the ordinary least squares (OLS) coefficient estimate for X_j and α_j corresponds to the proportion of individual RandFS models in which X_j was included.

In this sense, the RandFS procedure can be seen as producing shrinkage and the amount of shrinkage α_j on each feature depends on both the probability that the feature is made eligible and the probability that the feature is actually selected if made available. While the latter probability depends on the particular modeling technique and loss function employed, the probability of being made eligible is a direct function of only `mtry`.

But the previous statement is only valid under the typical "fixed p" setup where the dimensionality of the feature space is assumed fixed. Suppose instead that `mtry` is held fixed and that the procedure is repeated on an augmented feature space where more noise variables are added. Then under the same setup as above, it's clear that α_j decreases as a function of the number of extra noise features q since each original feature will thus have a lower probability of being made eligible. However, even for large values of `mtry`, we argue further that the probability of being selected once eligible also decreases as q increases and that such a decrease can be particularly dramatic for features only weakly related to the response. Indeed, given an original feature X_j not perfectly correlated with the response Y in this linear model setting, if we generate additional independent random noise features, eventually some will appear more correlated with Y just by random chance and the weaker

the correlation between X_j and Y, the fewer the number of noise features we would expect to need to generate in order to see this. Put simply, as more noise features are added to the model, the probability that some of those new features will appear at least as important as X_j grows with q. Thus, even for large values of `mtry` where the procedure begins to resemble that of bagging, the augmented version of the procedure may produce a similar kind of regularization and shrinkage to that offered by traditional random forests.

3.4.2 AugBagg and OLS Ensembles

While our work in Chapter 2 utilizes linear model forward selection settings in order to better illustrate the regularization effect of random forests, in work appearing around the same time, LeJeune et al. [2020] provided an in-depth analysis focused on ensembles where each base learner is simply a linear model constructed on a subsample of features and observations with coefficients estimated via ordinary least squares. As in Chapter 2, the authors observe that feature subsampling at the base-learner stage produces a regularization effect, concluding that for optimally-tuned subsampling rates, the asymptotic risk of the OLS ensemble is equal to the asymptotic risk of ridge regression, an explicit regularization procedure. Here we review the setup utilized in LeJeune et al. [2020] and demonstrate that the same procedure applied to an augmented design is equivalent to one in which more shrinkage is applied to the original data.

Assume now that we have data of the form $\boldsymbol{Z}_1, ..., \boldsymbol{Z}_n$ where each $\boldsymbol{Z}_i = (\boldsymbol{X}_i, Y_i)$ and

$$Y_i = \boldsymbol{X}_i'\beta + \epsilon_i$$

where $Y_i \in \mathbb{R}$ denotes the response, the features $\boldsymbol{X}_i \in \mathbb{R}^p$ are drawn i.i.d. from $\mathcal{N}_p(0_{p\times 1}, \Sigma)$, and the ϵ_i are i.i.d. with mean 0 and variance σ_ϵ^2 and are independent of $\boldsymbol{X}$.

To build OLS ensembles, we draw B submatrices by applying row subsampling to the observations and column subsampling on $\boldsymbol{X} = [\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n]'$. Let S_b and T_b denote the sets of column and row indices, respectively, selected in the b^{th} model, while $\boldsymbol{S}_b$ and $\boldsymbol{T}_b$ denote the subsampling matrices obtained by selecting the the columns from I_p and I_n corresponding

to the indices in S_b and T_b. Let $\mathcal{S}$ and $\mathcal{T}$ denote the entire collections of all possible S_b and T_b, respectively. For each base learner, the OLS minimum-norm estimator is given by

$$\hat{\beta}^{(b)} = \boldsymbol{S}_b \left(\boldsymbol{T}_b' \boldsymbol{X} \boldsymbol{S}_b\right)^+ \boldsymbol{T}_b Y$$

where $(\cdot)^+$ denotes the Moore-Penrose pseudoinverse, so that the estimated coefficients of the ensemble are thus given by

$$\hat{\beta}^{ens} = \frac{1}{B} \sum_{b=1}^{B} \boldsymbol{S}_b \left(\boldsymbol{T}_b' \boldsymbol{X} \boldsymbol{S}_b\right)^+ \boldsymbol{T}_b Y.$$

The risk of $\hat{\beta}^{ens}$

$$R(\hat{\beta}^{ens}) \triangleq \mathbb{E}_{\boldsymbol{x}} \left[\left\langle \boldsymbol{x}, \beta - \hat{\beta}^{ens} \right\rangle^2 \right] = \left\langle \beta - \hat{\beta}^{ens}, \Sigma(\beta - \hat{\beta}^{ens}) \right\rangle$$

is defined as the expected squared error at an independent point $\boldsymbol{x}$, where the $\langle \cdot, \cdot \rangle$ notation denotes the Frobenius inner product. LeJeune et al. [2020] then employ the following assumptions to allow for a more precise evaluation of the risk.

Assumption 1. *(Finite Subsampling) The subsets in the collections $\mathcal{S}$ and $\mathcal{T}$ are selected at random such that $|S_b| < |T_b| - 1$ and that the following hold:*

1. *$Pr(j \in S_b) = \frac{|S_b|}{p}$ for all $j \in [p] = \{1, 2, \ldots, p\}$*
2. *$Pr(m \in T_b) = \frac{|T_b|}{n}$ for all $m \in [n]$*
3. *The subsets $S_1, S_2, \ldots, S_B, T_1, \ldots, T_B$ are conditionally independent given the row subsample sizes $(|T_b|)_{b=1}^B$.*

Assumption 2. *(Asymptotic Subsampling) For some $\alpha, \eta \in [0, 1]$, the subsets in the collections $\mathcal{S}$ and $\mathcal{T}$ are selected randomly such $|S_b|/p \xrightarrow{a.s.} \alpha$ as $p \to \infty$ and $|T_b|/n \xrightarrow{a.s.} \eta$ as $n \to \infty$ for all $b \in [B]$.*

Furthermore, it is assumed that $\Sigma = I_p$, that $\|\beta\|_2 = 1$, and that $p/n \to \gamma$ with $\eta > \alpha\gamma$ as $n, p \to \infty$.

Under these assumptions, conditional on the subset sizes, the expected risk of the bias and variance over $\boldsymbol{X}$, $\mathcal{S}$ and $\mathcal{T}$ converge almost surely as follows:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{X},\mathcal{S},\mathcal{T}}\left[\text{bias}(\hat{\beta}^{ens})\right] &\xrightarrow[a.s.]{p,n\to\infty} \frac{B-1}{B}\left(\frac{(1-\alpha)^2}{1-\alpha^2\gamma}\right) + \frac{1}{B}\left(\frac{\eta(1-\alpha)}{\eta-\alpha\gamma}\right) \\
&\xrightarrow{B\to\infty} Bias(\alpha,\gamma) := \frac{(1-\alpha)^2}{1-\alpha^2\gamma} \\
\mathbb{E}_{\boldsymbol{X},\mathcal{S},\mathcal{T}}\left[\text{var}(\hat{\beta}^{ens})\right] &\xrightarrow[a.s.]{p,n\to\infty} \frac{B-1}{B}\left(\frac{\sigma^2\alpha^2\gamma}{1-\alpha^2\gamma}\right) + \frac{1}{B}\left(\frac{\sigma^2\alpha\gamma}{\eta-\alpha\gamma}\right) \\
&\xrightarrow{B\to\infty} Var(\alpha,\gamma) := \frac{\sigma^2\alpha^2\gamma}{1-\alpha^2\gamma}
\end{aligned}$$

Thus, for an OLS ensemble built with subsamples drawn such that $|S_b| = \lfloor \alpha p \rfloor$ and $|T_b| = \lfloor \eta n \rfloor$ with $p/n \to \gamma$ and ensemble size $B \to \infty$, $\mathbb{E}_{\boldsymbol{X},\mathcal{S},\mathcal{T}}\left[\text{bias}(\hat{\beta}^{ens})\right]$ and $\mathbb{E}_{\boldsymbol{X},\mathcal{S},\mathcal{T}}\left[\text{var}(\hat{\beta}^{ens})\right]$ will converge almost surely to $Bias(\alpha,\gamma)$ and $Var(\alpha,\gamma)$ respectively. Notice that for fixed γ, $Bias(\alpha,\gamma)$ is decreasing in α while $Var(\alpha,\gamma)$ is increasing in α.

Now suppose that the same kind of subsampled OLS ensemble is constructed on an augmented feature space where $\boldsymbol{X}$ is augmented with $\boldsymbol{N} = [\boldsymbol{N}_1, \ldots, \boldsymbol{N}_n]' \in \mathbb{R}^{n\times q}$, and where the $\boldsymbol{N}_i$ are drawn i.i.d. from $N_q(0_{q\times 1}, I_q)$. Let $S_b^\star$ and $T_b^\star$ denote the subsampling indices on the b^{th} model constructed on this augmented design $[\boldsymbol{X}\ \boldsymbol{N}]$ and suppose that the subsampling sizes remain the same as in the OLS ensemble constructed on the original data so that $|S_b^\star| = |S_b|$ and $|T_b^\star| = |T_b|$. Furthermore, suppose that the number of additional features $q \to \infty$ as $p \to \infty$ such that $\frac{q}{p} \to \theta$ for some constant $\theta > 0$. Under these assumptions,

$$\begin{aligned}
\frac{|S_b|}{p+q} &\to \frac{\alpha}{1+\theta} = \alpha^\star \\
\frac{p+q}{n} &\to (1+\theta)\gamma = \gamma^\star,
\end{aligned}$$

and so $\mathbb{E}_{\boldsymbol{X},\mathcal{S},\mathcal{T}}\left[\text{bias}(\hat{\beta}^{ens})\right]$ and $\mathbb{E}_{\boldsymbol{X},\mathcal{S},\mathcal{T}}\left[\text{var}(\hat{\beta}^{ens})\right]$ converge to $Bias(\alpha^\star,\gamma^\star)$ and $Var(\alpha^\star,\gamma^\star)$, respectively. More specifically,

$$Var(\alpha^\star,\gamma^\star) = \frac{\sigma^2\alpha^{\star 2}\gamma^\star}{1-\alpha^{\star 2}\gamma^\star} = \frac{\sigma^2\alpha^2\gamma}{1+\theta-\alpha^2\gamma}$$

is decreasing with $\theta \geq 0$, so $Var(\alpha^\star, \gamma^\star) \leq Var(\alpha, \gamma)$. Similarly, under the assumption that $\eta > \alpha\gamma$,

$$Bias(\alpha^\star, \gamma^\star) = \frac{(1-\alpha^\star)^2}{1-\alpha^{\star 2}\gamma^\star} = \frac{(1+\theta-\alpha)^2}{(1+\theta)^2 - (1+\theta)\alpha^2\gamma}$$

is increasing with $\theta \geq 0$, so $Bias(\alpha^\star, \gamma^\star) \geq Bias(\alpha, \gamma)$. Thus, constructing an OLS ensemble on an augmented design leads to a more regularized estimator with increased bias and decreased variance – the same effect as would be found by constructing the ensemble on the original design with the same η but a smaller subsampling rate.

3.4.3 Implicit Regularization and Ridge Regression

In addition to the work described above, an intriguing collection of work has emerged in recent years on the so-called "double-descent" phenomenon coined by Belkin et al. [2019], whereby the generalizability error of models may sometimes continue to improve beyond the point of interpolation where training error vanishes. Hastie et al. [2019] followed up this work with an impressive and thorough analysis on the behavior of minimum norm interpolation for high-dimensional least squares estimators. While this work focused on the "ridgeless" setting, interesting related results have also been established for ridge and kernel ridge regression. Kobak et al. [2020] showed that for a standard ridge estimator of the form

$$\hat{\beta}_\lambda = (\boldsymbol{X}'\boldsymbol{X} + \lambda I)^{-1}\boldsymbol{X}'Y$$

the optimal penalty λ can be 0 or negative even when $p \gg n$. In particular, this may happen when the majority of signal comes from a small subset of high-variance features due to an implicit regularization effect offered by a larger collection of relatively low-variance noise features. In very recent work, Jacot et al. [2020] consider ridge estimators acting on a (possibly larger) transformed feature space consisting of Gaussian random features and show that such an estimator with ridge penalty λ is close to a kernel ridge regression estimator with effective penalty $\tilde{\lambda}$ where $\tilde{\lambda} > \lambda$. d'Ascoli et al. [2020] consider a similar random feature setup in investigating the double descent behavior of neural networks and provide a thorough review of much of the recent work on interpolation where we would refer interested readers.

In motivating the AugBagg procedure proposed above, we turn to a key result from Kobak et al. [2020]. As above, assume we have (original) training data of the form $(\boldsymbol{X}, Y)$ where $y = \boldsymbol{x}'\beta + \epsilon$ and let $\hat{\beta}_\lambda$ denote the ridge estimator of $\beta \in \mathbb{R}^p$. Now consider a new estimator $\hat{\beta}_q$ formed by performing minimum norm least squares and taking only the first p elements after augmenting $\boldsymbol{X}$ with q additional i.i.d. noise features, each with mean 0 and variance λ/q. The theorem below shows that augmenting the original design with low-variance noise features produces an equivalent regularization effect to ridge regression.

Theorem 3. [Kobak et al. [2020]] *Under the setup described above,*

$$\hat{\beta}_q \xrightarrow[q\to\infty]{a.s.} \hat{\beta}_\lambda.$$

Furthermore, for any $\boldsymbol{x}$, let $\hat{y}_\lambda = \boldsymbol{x}'\hat{\beta}_\lambda$ denote the ridge prediction and let $\hat{y}_{Aug}$ be the prediction generated by the augmented model that includes the additional q parameters using $\boldsymbol{x}$ extended with q random elements generated in the same fashion. Then

$$\hat{y}_{Aug} \xrightarrow[q\to\infty]{a.s.} \hat{y}_\lambda.$$

Kobak et al. [2020] go on to note that a direct but surprising consequence of this result is that "*adding random predictors with some fixed small variance could in principle be used as an arguably bizarre but viable regularization strategy similar to ridge regression.*" Furthermore, the final statement in Theorem 3 implies that the expected MSE of the augmented model (i.e. the non-truncated model that includes the q additional noise features) converges to the MSE of the ridge estimator as $q \to \infty$. In particular, note that when the optimal λ is non-zero, the augmented model with noise features generated according to the procedure outlined above will outperform the model that utilizes only the original data.

Figure 3.4 gives a demonstration of this surprising result. Here we utilize the same linear model setup described in previous sections with $n = 100$ observations, $p = 75$ features, the first $s = 5$ of which are signal with a coefficient equal to 1. For each SNR, we begin by generating 100 independent datasets and perform cross-validation on each to obtain 100 estimates of the optimal value of λ; the final estimate $\hat{\lambda}_{opt}$ is taken as the median across these. Then, for each combination of SNR and q, we generate an independent training set

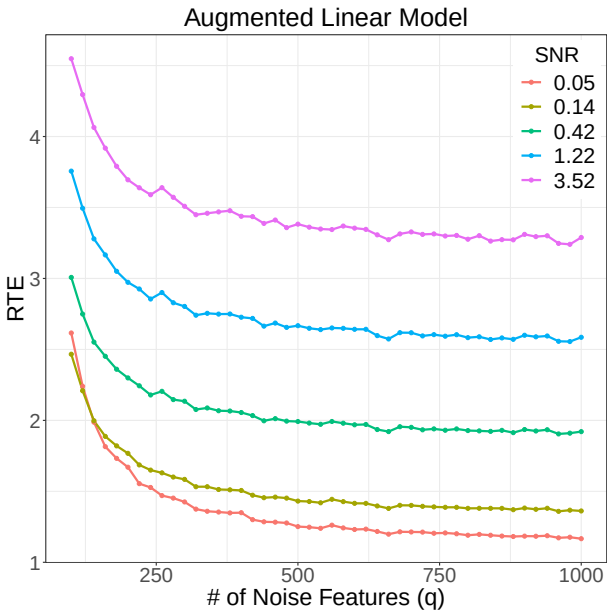

Figure 3.4: Performance of augmented linear model across different SNRs as increasingly many noise features are added to the model.

where the q additional noise features are sampled i.i.d. from $\mathcal{N}(0, \hat{\lambda}_{opt}/q)$. The minimum-norm OLS estimator is then calculated via the singular value decomposition and the relative test error is recorded on an independent test set with 100 observations. The entire process is repeated 100 times and the average relative test error is shown in Figure 3.4. In each case, we see clearly that the model error decreases as more noise features are added into the model.

Suppose now that we build ensembles of estimators of the kind in Theorem 3 by drawing B subsamples, constructing the estimators on each subsample, and averaging. Similar to the setup used above in LeJeune et al. [2020], let $T_b \subseteq [n]$ be the set of indices of selected observations in the b^{th} subsample and let $\boldsymbol{T}_b$ be the $n \times |T_b|$ matrix obtained by selecting columns from I_n corresponding to the indices in T_b. Construct $\hat{\beta}_q^{(b)}$ as above based on $\boldsymbol{T}_b'\boldsymbol{X}$ and $\boldsymbol{T}_b'Y$, which denote the design matrix and response, respectively, corresponding to the observations selected in b^{th} subsample. The final ensemble coefficient estimate formed by

averaging the augmented minimum norm estimators is given by

$$\hat{\beta}^{ens} = \frac{1}{B}\sum_{b=1}^{B}\hat{\beta}_q^{(b)}$$

where, by Theorem 3,

$$\hat{\beta}^{ens} = \frac{1}{B}\sum_{b=1}^{B}\hat{\beta}_q^{(b)} \xrightarrow[q\to\infty]{a.s.} \frac{1}{B}\sum_{b=1}^{B}\hat{\beta}_\lambda^{(b)}$$

with

$$\hat{\beta}_\lambda^{(b)} = (\boldsymbol{X}'\boldsymbol{T}_b\boldsymbol{T}_b'\boldsymbol{X} + \lambda I_p)^{-1}\boldsymbol{X}'\boldsymbol{T}_b\boldsymbol{T}_b'Y.$$

Now consider an orthogonal setting where $\boldsymbol{X}\boldsymbol{X}' = I_n$ and let η denote the subsampling rate so that $|T_b|/n \to \eta \in (0, 1]$. Let C be a $n \times n$ diagonal matrix where C_{ii} is the number of times that the i^{th} observation appears in the B subsamples and let $\lambda_q = \frac{1+\lambda-\eta}{\eta} \geq \lambda$. Using the Woodbury matrix identity, a ridge estimator with penalty λ can be rewritten as

$$\hat{\beta}_\lambda = \frac{1}{1+\lambda}\boldsymbol{X}'Y,$$

and

$$\begin{aligned}
\hat{\beta}^{ens} \xrightarrow[q\to\infty]{a.s.} \frac{1}{B}\sum_{b=1}^{B}\hat{\beta}_\lambda^{(b)} &= \frac{1}{B}\sum_{b=1}^{B}(\boldsymbol{X}'\boldsymbol{T}_b\boldsymbol{T}_b'\boldsymbol{X} + \lambda I_p)^{-1}\boldsymbol{X}'\boldsymbol{T}_b\boldsymbol{T}_b'Y \\
&= \frac{1}{B}\sum_{b=1}^{B}(\lambda^{-1}I_p - (\lambda(\lambda+1))^{-1}\boldsymbol{X}'\boldsymbol{T}_b\boldsymbol{T}_b'\boldsymbol{X})\boldsymbol{X}'\boldsymbol{T}_b\boldsymbol{T}_b'Y \\
&= \frac{1}{B}\sum_{b=1}^{B}\frac{1}{1+\lambda}\boldsymbol{X}'\boldsymbol{T}_b\boldsymbol{T}_b'Y \\
&= \frac{1}{1+\lambda}\frac{1}{B}\boldsymbol{X}'CY \\
&\xrightarrow{B\to\infty} \frac{\eta}{1+\lambda}\boldsymbol{X}'Y \\
&= \frac{1}{1+\lambda_q}\boldsymbol{X}'Y \\
&= \hat{\beta}_{\lambda_q}.
\end{aligned}$$

Thus, in this simple case, an ensemble of minimum-norm least squares estimators constructed on an augmented design produces an estimate equivalent to one produced via ridge

regression on the original design. Furthermore, the shrinkage produced by the ensemble is stronger than that of each individual base model.

On a final note, we stress that the purpose of producing this result is not to advocate for this kind of augmented bagging over ridge regression. Indeed, given the equivalence just described paired with the fact that ridge regression is both well-established and naturally motivated, it's difficult to imagine practical settings in which augmented bagging would offer any distinct advantage. Rather, we offer the above results primarily to make explicit the shrinkage that is produced by augmented bagging – a fact that has crucial implications for measuring and testing variable importance.

3.5 IMPLICATIONS FOR VARIABLE IMPORTANCE

Within any kind of black-box supervised learning framework, establishing a valid means of measuring the importance of features is of utmost importance. Indeed, in such non-parametric regimes where model fit and behavior remain largely hidden from view, understanding how features contribute information to the prediction is paramount for scientists and practitioners. In the context of bagging and random forests specifically, Breiman's original out-of-bagg (oob) [Breiman, 2001] importance scores are one such popular measure, though many issues such as a preference for correlated features and those with many categories have been noted in the years following their introduction [Strobl et al., 2007, Nicodemus et al., 2010, Toloşi and Lengauer, 2011]. As a result, various formal hypothesis testing procedures have recently been developed to more accurately assess the importance of features in such ensembles. Unfortunately, as demonstrated in the following subsection, even these more rigorous tests are sometimes vulnerable to highly misleading results due to the potentially beneficial effects of noisy features described in the previous sections.

Algorithm 3 Random Forest Permutation Test [Coleman et al., 2019]

Input: Original training set $\mathcal{D}_n$, test set $\mathcal{D}_{\text{test}}$, number of permutations P

Create alternative data $\mathcal{D}_n^*$

Build ensemble RF with $\mathcal{D}_n$ and predict at $\mathcal{D}_{\text{test}}$

Build ensemble RF^* with $\mathcal{D}_n^*$ and predict at $\mathcal{D}_{\text{test}}$

Compute difference in errors $d_0 = MSE(RF^*) - MSE(RF)$

for i in $1 : P$ **do**

Randomly shuffle base models between ensembles to form RF_i and RF_i^*

Compute permuted difference in errors $d_i = MSE(RF_i^*) - MSE(RF_i)$

Calculate p-value $p = \frac{1}{P+1}\left[1 + \sum_{i=1}^{P} I(d_0 > d_i)\right]$

3.5.1 Hypothesis Tests for Importance

Recently, Mentch and Hooker [2016] proposed a formal hypothesis testing procedure for measuring feature importance in random forests. Given a generic relationship of the form $y = f(\boldsymbol{x}) + \epsilon$, the authors consider partitioning the original set of features $\boldsymbol{X}$ into two groups, $\boldsymbol{X}_0$ and $\boldsymbol{X}_{\text{test}}$, where the latter group contains the features of interest so that a null hypothesis of the form

$$H_0 : f(\boldsymbol{X}_0, \boldsymbol{X}_{\text{test}}) = f_0(\boldsymbol{X}_0) \tag{3.5}$$

may be rejected whenever the features in $\boldsymbol{X}_{\text{test}}$ make a significant contribution to predicting the response. The authors propose to evaluate the hypothesis in (3.5) by constructing two separate random forest models: one constructed on the original data and one constructed on an altered dataset where the features in $\boldsymbol{X}_{\text{test}}$ are either substituted for randomized replacements independent of the response or dropped from the model entirely. Predictions from each forest are then computed at a number of test points and the differences are combined to form an appropriate test statistic. Coleman et al. [2019] recently proposed a permutation-based alternative to this test. Here again, two forests are constructed in the same fashion as just described, but trees are then randomly permuted across forests and the new difference in accuracy between forests is recorded. That process is then repeated many times to form the null distribution of accuracy differences to which the original difference

in accuracy can be compared. An outline of this test is given in Algorithm 3. Note that this nonparametric test avoids the need for explicit variance calculation and as a result is far more computationally efficient and scalable. Also note that these tests are carried out below in the context of subsampled bagging (i.e. with non-random trees), though this can be seen as merely a special case of random forests with the `mtry` value set equal to the total number of features available.

Crucially, these tests ultimately rely on measuring the difference between either raw predictions or predictive accuracy between two tree-based ensembles constructed on different training sets. Both papers advocate for replacing the features under investigation with randomized alternatives, noting that the tests can potentially produce spurious results when features are instead dropped from the second model, though neither provides a detailed explanation as to why this occurs. Elsewhere in the literature, alternative tests specifically propose to evaluate feature importance by measuring the drop in performance when the features in question are removed from the model. Such is the case, for example, with the **L**eave-**O**ut-**Co**variates (LOCO) measure proposed by Lei et al. [2018] in the context of conformal inference and most recently in the tests proposed by Williamson et al. [2021]. Furthermore, though often done informally, it remains common throughout the broader scientific literature for authors to argue for the importance of particular variables based on decreases in model performance when such variables are excluded.

The results presented in the sections above present a substantial concern with such measures. In particular, if model performance can be improved simply by adding randomly generated features that are (at least conditionally) independent of the response, then observing a significant improvement in accuracy when a particular set of features is included does not imply that any relationship to the response or even the other covariates need exist.

To emphasize this point, we implement the test for variable importance recently developed in Coleman et al. [2019] and investigate its behavior under simulated settings. We utilize the same linear model setup as in previous sections with $p = 5$ original signal features sampled from $\mathcal{N}_p(\mathbf{0}, \Sigma)$ with $\Sigma_{ij} = \rho^{|i-j|}$ and $\rho = 0.35$ and consider adding q additional noise features to test for importance. These noise features are either independent of the original five features or are correlated with a randomly selected signal feature with correlation

strength r. Thus, relative to the sort of generic null hypothesis specified in (3.5), our default set of features consist of the original signals so that $\boldsymbol{X}_0 = (X_1, ..., X_5)$ and the features under investigation are those additional noise features being added, $\boldsymbol{X}_{\text{test}} = (N_1, ..., N_q)$. As done previously, the error in the model is adjusted to produce a pre-specified SNR.

To carry out the procedure in Coleman et al. [2019], for each test, we create a training set $\mathcal{D}_n$ with $n = 500$ observations and a test set $\mathcal{D}_{Test}$ with 1000 observations. Let $\boldsymbol{X}$ denote the original $n \times (p+q)$ design matrix and $\boldsymbol{X}^*$ denote the design matrix where the q noise features of interest are either dropped or replaced with a random substitute. Thus, for "drop tests", $\boldsymbol{X}^*$ will be of dimension $n \times p$ whereas for "replacement tests", $\boldsymbol{X}^*$ will be of dimension $n \times (p + q)$. We construct two decision tree ensembles, each with 100 trees. Each tree in the first ensemble is built on a subsample of size 100 from the original training data $(Y, \boldsymbol{X})$; each tree in the second ensemble is built on a subsample of size 100 from the modified data $(Y, \boldsymbol{X}^*)$. Each ensemble here is thus constructed via *subbagging*, though trees are still non-random built to full depth. After recording the original error difference between the two ensembles, trees are randomly shuffled between ensembles a total of 1000 times and each time this new permuted error difference is recorded to form the null distribution. The null hypothesis that the q noise features are not important is rejected whenever the original error difference lies in the upper quantile of the null distribution of permuted error differences. This entire procedure is then repeated 500 times to form empirical rejection probabilities.

Figure 3.5 shows the probability of rejecting H_0 and concluding the additional noise features are important across various SNRs and numbers of additional features when those features are either dropped or replaced by null substitutes. For these as well as each of the tests deployed below, we set the nominal level to the standard $\alpha = 0.05$ so that if the tests are performing as intuitively expected, we should only see the null hypothesis to be rejected (indicating that the noise features are significant) about 5% of the time. However, it is readily apparent that for the drop tests (Figure 3.5 Left Column), rejections routinely happen well over 5% of the time. This is particularly evident at low SNRs when many additional noise features correlated with the original five features are added where we see (Figure 3.5 Bottom Left) rejection rates surpassing even 50%.

In previous work both in Mentch and Hooker [2016] and Coleman et al. [2019], the

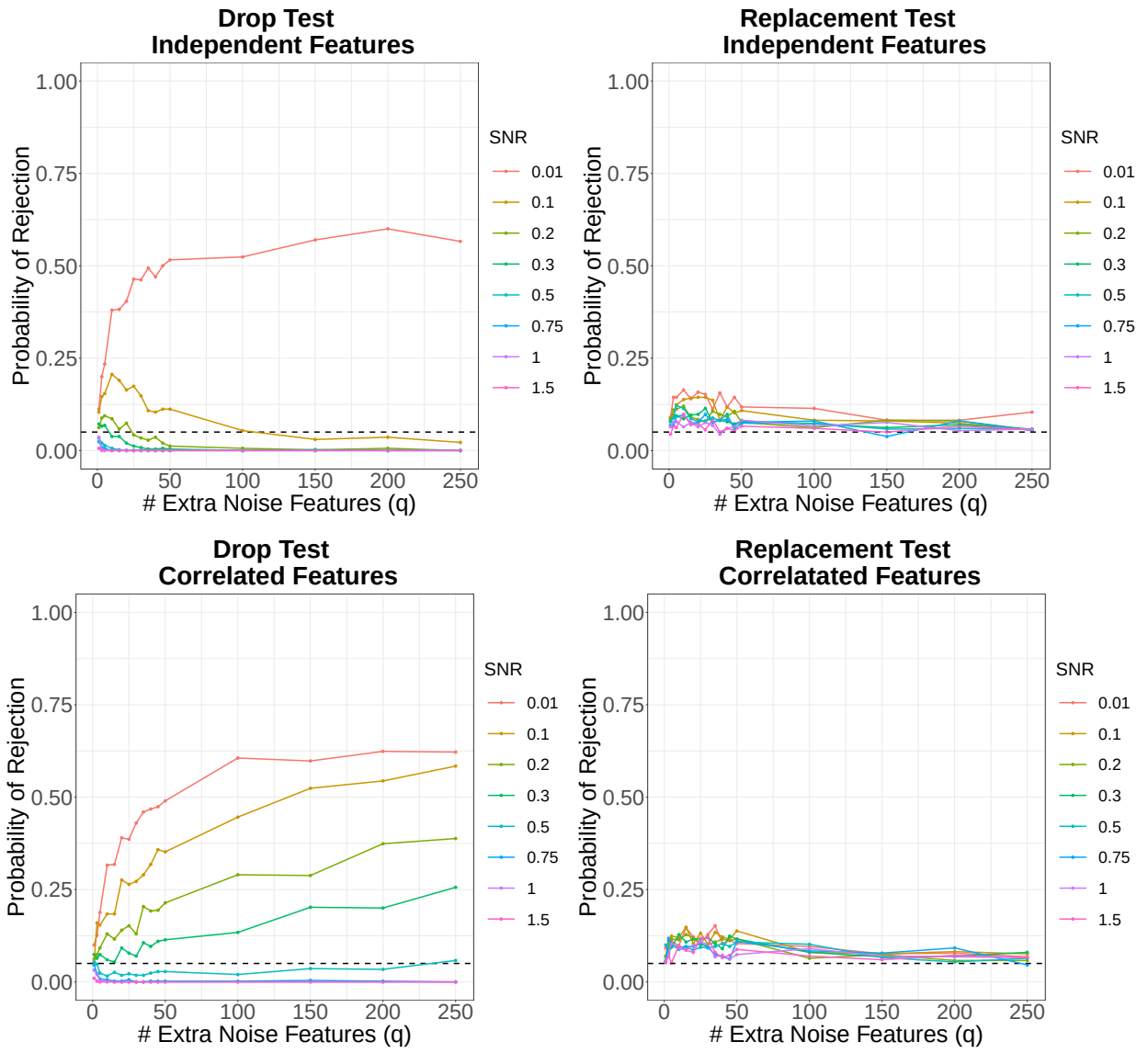

Figure 3.5: Probability of rejecting the null hypothesis and concluding an additional independent set of noise features are important when dropping the features in question (left column) vs replacing the features in question (right column) when those features are independent (top row) vs correlated (bottom row).

authors claim that the testing procedures developed within are more robust whenever the features under investigation are replaced by randomly generated substitutes rather than being dropped from the model entirely. And indeed, from the right column of Figure 3.5

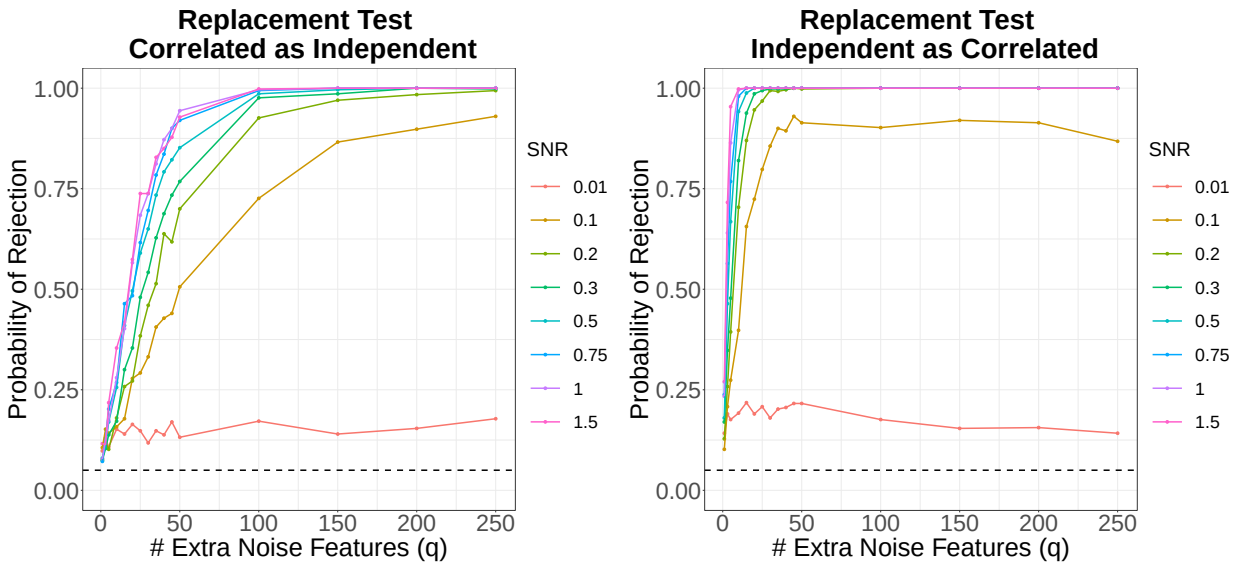

Figure 3.6: Probability of rejecting the null hypothesis using replacement tests where correlated features are replaced with independent features (left) and where independent features are replaced with correlated features (right).

it is readily observed that regardless of the SNR or the dependence structure of the noise features on the original features, these replacement tests appear to be far better behaved. Note that these rejection rates do lie very slightly above the nominal rate of 5%, however. This is because the tests developed in Coleman et al. [2019] are valid only asymptotically and in particular, rely on a notion of asymptotic independence between the base models (in this case, trees), which can be achieved asymptotically by subsampling at sufficiently slow rates.

Unfortunately, carrying out accurate replacement-style tests in practice is easier said than done. In the plots shown in the right-hand column of Figure 3.5, the replacement noise features are sampled from exactly the same distribution as the original noise features being tested for importance. In practice, of course, the distribution of the features in question is unknown. Figure 3.6 compares the performance of these replacement tests whenever noise features of one kind are replaced by noise features of another kind. On the left, the original noise features are randomly correlated with an original signal feature at $r = 0.7$ and

these features are replaced with independent noise features. Here we again notice quite a troubling trend: the test has a very high probability of rejecting across all but the lowest SNRs and this probability appears to increase with q. Perhaps even worse is the fact that the rejection probabilities appear to be increasing at a faster rate at higher SNRs. Thus, even in "good data" settings, it appears that such tests are very likely to cause correlated noise features to appear important whenever testing against the performance of a model using only independent noise (or, for example, permutations of the original features) as a substitute. While this setting is likely most representative of what might often happen in practice, for completeness, we also consider the opposite setting in the plot on the right of Figure 3.6 where independent noise features are replaced with ones correlated with a randomly selected feature in $\boldsymbol{X}_0$. Here again we see the same kind of troubling results. These results highlight the potential issues with replacing features by randomized replacements from a different distribution and thus might suggest some promise for procedures involving knockoff variables [Barber et al., 2015, Candes et al., 2018] that explicitly attempt to generate randomized replacements from the same distribution as the original copies. Indeed, recent work by Hooker et al. [2021] suggests such approaches can sometimes offer a drastic improvement, even in low SNR settings.

3.5.2 Intrinsic vs Extrinsic Testing

Though troubling, these results above should not be at all surprising given the empirical results in Section 3.3 that showed strong evidence of improved performance when additional noise features are added to the model. These tests simply make clear that such improvements are routinely large enough to register statistical significance. We caution readers from drawing too much from the particular rejection probabilities shown in the left column of Figure 3.5. These empirical results should in no way be seen as guidelines for how often or under what settings such tests will produce inflated rejection proportions. Rather, the amount by which these kinds of tests inflate the anticipated rejection proportion will depend entirely on the relationships within the data as well as the power of the particular testing procedure employed. Indeed, similar testing procedures with higher power could potentially

reject even more often than shown in Figure 3.5 for the same datasets. By the same reasoning, ensembles consisting of base learners other than trees may also reject more or less often.

The tests carried out above from Coleman et al. [2019] are what a recent work by Williamson et al. [2021] referred to as *extrinsic* tests in that the results are model-specific. Formally, when MSE is the measure of error employed, the hypotheses in Coleman et al. [2019] can be written as

$$\begin{aligned} H_0 &: \mathbb{E}(MSE_{RF}(\mathcal{D}_{\text{test}})) = \mathbb{E}(MSE_{RF^*}(\mathcal{D}_{\text{test}})) \\ H_1 &: \mathbb{E}(MSE_{RF}(\mathcal{D}_{\text{test}})) < \mathbb{E}(MSE_{RF^*}(\mathcal{D}_{\text{test}})) \end{aligned} \tag{3.6}$$

where the test set $\mathcal{D}_{\text{test}}$ is assumed fixed and the expectation is taken across the training data and any additional randomness involved with the construction of the base learners. By contrast, Williamson et al. [2021] recently put worth a framework for testing *intrinsic* or *population-level* (model-agnostic) notions of variable importance. In particular, the authors consider defining the importance of a collection of features $\mathcal{S}$ as the amount of *oracle predictiveness* lost when those features are excluded. While the framework is flexible enough so as to allow for various forms of importance measures, in our context here, the most natural corresponding hypotheses for this kind of intrinsic test can be written as

$$\begin{aligned} H_0 &: E(Y - E(Y|X))^2 = E(Y - E(Y|X_{-\mathcal{S}}))^2 \\ H_1 &: E(Y - E(Y|X))^2 < E(Y - E(Y|X_{-\mathcal{S}}))^2. \end{aligned} \tag{3.7}$$

Comparing the hypotheses in (3.6) to those in (3.7), one may wonder why we applied the extrinsic tests in Coleman et al. [2019] rather than the intrinsic tests in Williamson et al. [2021]. Indeed, given that the extrinsic tests reject so often, the intrinsic alternative may appear to be the natural solution and even the more direct way of addressing the question really of interest in most practical settings. Unfortunately, while this may be true in theory, valid application of such intrinsic testing procedures requires several strong assumptions, including, for example, that the estimators converge to the true conditional expectations at a rate of $n^{-1/4}$. This is, of course, difficult to guarantee for flexible learning procedures like the bagging and random forest procedures that we employ here consisting of CART-style trees as base learners.

Appendix B contains more detail on the mechanics of how intrinsic tests in Williamson et al. [2021] can be carried out. We also demonstrate that in this case, nearly identical steps can be taken to produce an analogous extrinsic test. Note from the plots in Appendix B that this extrinsic analogue of the test in Williamson et al. [2021] produces results very qualitatively similar to those in Section 3.5.1 that utilize the extrinsic test in Coleman et al. [2019].

The fact that extrinsic tests exist that can be carried out in nearly identical fashion to those of analogous intrinsic tests highlights the slipperiness of this issue. Indeed, put simply, it would seem that the primary difference between intrinsic and extrinsic tests largely boils down to the assumptions one is willing to make. Practitioners should thus take extreme care in considering the necessary assumptions before claiming to have conducted a valid intrinsic test. Likewise, readers should always regard claims that a valid intrinsic test was carried out with guarded skepticism and an eye toward whether the necessary assumptions are truly met in the context at hand. Suppose, for example, that one carries out a particular test and finds a significant result; that is, the test rejects the null hypothesis and therefore suggests that a particular collection of features is important. If the test was a valid intrinsic test where all necessary assumptions are met, then one can conclude that there is evidence that those features really do hold unique predictive power for the response not contained in the other features. On the other hand, if those assumptions are not met, the test should therefore be seen as only an extrinsic test and thus, just as we have seen throughout this paper, it is possible that those features may be improving the predictive accuracy of the model and yet may be totally or at least conditionally independent of the response.

Finally, we close this section with a brief discussion on the role of model selection and tuning. In developing their framework for intrinsic testing, Williamson et al. [2021] note the importance of considering a sufficiently rich class of predictive models and carefully tuning across that model class in order to find the optimal predictive model before one should consider moving forward with formal inference. On this point we certainly agree. Indeed, our overarching point in this paper was to show that the predictive accuracy of some supervised learning models could be improved by merely including additional irrelevant noise features. While we focused primarily on bagging to demonstrate this point, it is likely that

these troubling effects would have been less severe had we, for example, considered an entire class of random forests and tuned across the `mtry` parameter and depth of trees. Indeed, as noted in the sections above, the additional noise features here are simply serving as a means of implicit regularization. If sufficient regularization can be accomplished via other means, the additional noise features may no longer be of additional benefit and may in fact start to degrade performance as one would expect. This highlights the crucial importance of carefully tuning a random forest via some form of external or cross validation before undertaking any kind of inference. On the other hand, we also want to stress that tuning across a large class is not necessarily sufficient to guarantee the kind of fast convergence needed for intrinsic testing. In recent work by Hastie et al. [2020], for example, the authors repeatedly demonstrate that at low SNRs, best subset selection (in which every possible linear model is constructed) performs quite poorly even when tuned on a large external validation set.

3.5.3 Bad Tests or Bad Interpretations?

Given the results in Section 3.5.1, one may be tempted to conclude that procedures of this style that assign relevance to features based on the improvement in predictive accuracy seen when they are included are simply "bad" because the outcomes are "wrong" far too often. Indeed, if rejecting the null hypothesis in a test of this sort is taken to mean that the features in question are "important" and "important" is taken to mean that those features possess some unique explanatory power for the response not captured by the other features available, then certainly such tests would appear to be highly problematic as the rejection rates in the above settings very often lie far above the nominal level of $\alpha = 0.05$.

In our view, however, such an understanding is too naive. The demonstrations above do not necessarily imply that anything is wrong with the tests themselves. Rejecting the null hypotheses in such tests means only that there is evidence that the features in question improve model performance when included. The simulations in Section 3.3, however, suggest that even the inclusion of additional noise features can improve model performance, sometimes to a dramatic degree. As discussed in the previous subsection, while intrinsic tests can

theoretically overcome these model-specific defects, it's difficult to say in general when the necessary assumptions would be met in practical settings when flexible learning models are being employed.

This situation highlights the crucial need for precise language in discussions of feature importance. While "predictive improvement" intuitively feels like a natural proxy, it seems quite unlikely that features independent of the response (at least conditionally) ought to ever be considered "important" for most practical purposes. Certainly this is the case whenever scientists argue that particular features must be collected in order to construct the optimal predictive model or when arguing that features generated by a new piece of technology can lead to further improved model performance over those that were previously available.

In situations such as these, it seems that what is really being sought is not a measure of how "important" certain features may be, but rather how "essential" they are. Even when additional variables improve model performance, we really seek to determine whether they do so meaningfully or significantly more than randomized alternatives. Interested readers are also invited to see a similar discussion on *model class reliance* appearing recently in Fisher et al. [2019]. Finally, as alluded to also in Williamson et al. [2021], practitioners should always have in mind a notion of relevant effect size when conducting tests for importance such as these. While small upticks in predictive accuracy may sometimes be sufficient to achieve statistical significance for certain features, in practice those improvements may still not justify the cost of their collection and inclusion in the model.

3.6 DISCUSSION

The work in the preceding sections introduced the idea of augmented bagging (AugBagg), a simple procedure identical to traditional bagging except that additional noise features, conditionally independent of the response, are first added to the feature space. Surprisingly, we showed that this simple modification to bagging can lead to drastic improvements in model performance, sometimes even outperforming well-established alternatives like an optimally-tuned random forest. Performance gains appear most dramatic at low SNRs,

though the introduction of correlated noise features can continue to improve performance even at higher SNRs. The fact that performance can sometimes be dramatically improved by simply adding conditionally-independent features into the model has important implications for variable importance measures and especially in interpreting the results from tests of variable importance.

On one hand, this work fits well within the rapidly expanding collection of work that explores the potential benefits of excess noisy features. While some earlier papers experimented with the presence of additional noise either added to or multiplied across the original features prior to training [Bishop, 1995, Srivastava et al., 2014], a more popular recent trend has been to analyze models built with random features generated from transforms of the original predictors obtained, for example, via Gaussian Processes or the Random Fourier Features model (see, e.g., Rahimi and Recht [2007], Rudi and Rosasco [2017], Belkin et al. [2019], Mei and Montanari [2019], Hastie et al. [2019], Jacot et al. [2020]). Much of this recent work has focused on the idea of the "double descent", demonstrating both empirically and mathematically that purposeful over-parameterization – building models that contain more (random) features than observations – can sometimes be beneficial.

On the other hand, we are not aware of other work specifically defining a procedure by simply augmenting the original data with additional pure noise features to potentially achieve superior predictive accuracy. The fact that models constructed on larger and noisier feature collections are sometimes preferable would seem to run counter to much of traditional statistical thinking. Countless procedures have been proposed in recent decades that assume $\boldsymbol{X} = (\boldsymbol{X}_{\text{Signal}}, \boldsymbol{X}_{\text{Noise}})$ and attempt to uncover the subset of signal features $\boldsymbol{X}_{\text{Signal}}$ with a minimal 'false positive' rate. Indeed, many may intuitively believe that the setting where all available features are signal is something of a 'gold standard' for regression. While there may be good inferential reasons why separating signal and noise is important, this work suggests that such a task is unnecessary and perhaps even detrimental (at least for some models) whenever predictive accuracy is the primary objective.

Along those lines, though AugBagg may sometimes produce predictions substantially more accurate than an alternative baseline like random forests, we stress that the procedure should not be seen as replacing or superseding more efficient procedures like random

forests. As detailed in the introduction, random forests have a long documented history of off-the-shelf success and depending on the size of the data at hand, may be much more computationally feasible to implement in practice. Indeed, while random forests reduce the number of features considered at each node, AugBagg, by construction, explicitly increases this computational burden. Furthermore, while our work in Chapter 2 suggests tuning a random forests can sometimes improve performance, at least moderate success can often be found at default values. In contrast, a generic implementation of AugBagg involves tuning both the number of additional features and their correlation with the original features and we are not able to offer default values of these likely to be successful across a broad range of data settings.

Finally, we end by noting that all of the work above was considered within the context of regression. In tree-based contexts, this simply means that predictions at both the tree and ensemble level are formed by averaging. If one were to consider, for example, a classical 0-1 binary response setting in which these kinds of regression trees were still employed (so as to produce estimates generally interpreted as probabilities), we expect the same kinds of potential benefits of random noise features to be present. If, however, those probabilities are then used to perform classification or one employs a majority vote rather than an average, it is unclear to what extent those noise features may remain beneficial. We suspect that such benefits may depend heavily upon the class imbalance in the original data as well as the decision threshold(s) employed. We leave an in-depth study of these issues in classification settings as an open area for potential future work.

4.0 DEPTH AS REGULARIZATION: TREES, FORESTS, CHICKENS, AND EGGS

This chapter focuses on another inherent but often neglected source of regularization in random forests, namely tree depth, and reexamine the decades-old question of whether individual trees in an ensemble ought to be pruned. Despite the fact that default constructions of random forests use near full depth trees in most popular software packages, here we provide strong evidence that tree depth should be seen as a natural form of regularization across the entire procedure. In particular, our work suggests that random forests with shallow trees are advantageous when the signal-to-noise ratio in the data is low. In building up this argument, we also critique the newly popular notion of "double descent" in random forests by drawing parallels to U-statistics and arguing that the noticeable jumps in random forest accuracy are the result of simple averaging rather than interpolation. The following sections are mainly pulled from Zhou and Mentch [2021].

4.1 INTRODUCTION

Recently, Belkin et al. [2019] put forth the more general and now very popular idea of the "double descent" risk curve in which model error, when plotted against model complexity, exhibits the classical U-shaped curve followed by a second descent beyond the interpolation threshold, indicating that once the model becomes over-parameterized, performance can sometimes be further improved. The authors provide empirical evidence for this kind of effect with several different models including both neural networks and random forests. Since then, much effort has been made to provide the theoretical underpinnings for this kind of

effect with more tractable models such as "ridgeless" least squares regression [Hastie et al., 2019] and random features regression [Mei and Montanari, 2019] to name just a few.

An explanation along these lines, however, begs a number of interesting questions. First, how is model "complexity" best understood? Is an average over multiple models really more "complex" than the individual models themselves? Second, the idea of relating RF performance to interpolation creates a kind of chicken-and-egg problem: are RFs accurate *because* they interpolate, or is the (near) interpolation sometimes seen in RFs an innocuous side effect of accurate models constructed in this greedy manner?

In this chapter, we adopt the regularization framework in the last few chapters to investigate the impact of pruning trees in a random forest. Studies of this sort date back to more than two decades ago when Breiman [1996] first proposed the bagging procedure and showed that aggregating unstable trees built on bootstrap samples can lead to substantial gains in accuracy. Since then, two separate schools of thought seem to have emerged with one group arguing that trees in an ensemble should be grown full-depth and the other maintaining that tree-depth itself should be seen as a tuning parameter with the potential to greatly impact overall performance. Indeed, in the original RF proposal by Breiman [2001] as well as in the more recent investigative articles [Wyner et al., 2017, Belkin et al., 2019, Mentch and Zhou, 2020b], the trees in RFs are deep (at least near-full depth) and unpruned. This also remains the standard recommendation in many textbooks (e.g. James et al. [2013], Izenman [2008]) and is the default setting in many standard packages such as `Scikit-learn` [Pedregosa et al., 2011] in `python` and `randomForest` [Liaw et al., 2002] in `R` [R Core Team, 2017]. On the other hand, there is a wide body of existing work [Segal, 2004, Lin and Jeon, 2006, Duroux, Roxane and Scornet, Erwan, 2018, Probst et al., 2019b] offering strong empirical evidence that proper tuning can sometimes significantly improve performance; a more detailed review is given in Section 4.2.

Rather than merely adding yet another piece to the growing stack of literature on the topic, we try and characterize both *why* and *when* pruning trees within a RF has a significant impact on model fit. In particular, we argue that in addition to the bagging (resampling and averaging) and random-feature-subsetting aspects of RFs, tree depth can best be viewed as merely an additional form of regularization, making RFs with shallow trees preferable in low

signal-to-noise ratio (SNR) settings. Ironically, this characterization actually lends support to both schools of thought described above: at medium to high SNR settings, practitioners are unlikely to notice any meaningful gains in accuracy as a result of pruning individual trees because the bagging and random-feature-subsetting components of the model have already regularized the procedure to a sufficient degree. On the other hand, when the SNR is very low, pruning trees can offer a third additional form of regularization that can indeed result in improved performance.

The remainder of the chapter is laid out as follows. In Section 4.2, we offer a more thorough discussion on the previous work in the tree-pruning literature. A case study on the MNIST dataset designed to replicate the experiments in Belkin et al. [2019] and highlight the shortcomings of the double descent argument is given in Section 4.3. In Section 4.4 we provide both a theoretical motivation and a number of wide-ranging simulations to demonstrate the effect of tree depth on RF performance, properly characterize the settings in which pruning is advantageous, and provide suggestions for practical tuning strategies. We conclude with a discussion in Section 4.5. Throughout this chapter, we use the same notations as formalized in Section 2.2.

4.2 LITERATURE ON TREE DEPTH

When constructing stand-alone decision trees, it has long been understood that the depth of the tree must be carefully chosen to avoid under- or over-fitting. This is typically accomplished via a cost-complexity parameter that serves to trade off bias and variance by successively pruning away splits in a fully-grown tree whenever the gain in accuracy realized by including them fails to exceed some predefined threshold. In the case of tree-based ensembles, however, best practices are far less clear and the issue of whether tree depth should be tuned has been the subject of some debate for the past two decades.

Interestingly, Leo Breiman himself – who proposed both bagging [Breiman, 1996] and random forests [Breiman, 2001] – seemed to flip-flop on this issue. In the original paper on bagging Breiman [1996], Breiman proposed the idea of best pruned classification and

regression trees to be used in the ensemble. In proposing random forests, however, his advice switched: "*Grow the tree using CART methodology to maximum size and do not prune*" [Breiman, 2001].

Many textbooks agree with Breiman's latter advice of constructing RFs with full-depth trees. For example, Izenman [2008] states "*there are only two tuning parameters for a random forest: the number of variables randomly chosen as a subset at each node and the number of bootstrap samples. ... grow the tree to a maximum depth with no pruning*". Similarly, James et al. [2013] say that "*To apply bagging to regression trees ... These trees are grown deep, and are not pruned*". Likewise, the default settings of many widely used statistical computing packages also follow this suggestion of constructing trees to (at least near) full depth. The `randomForest` package in `R` constructs trees to the maximum possible depth subject to the constraint of `nodesize` = 5 for regression and `nodesize` = 1 for classification trees. Similarly, with `Scikit-learn` in `python`, cells are split until all leaves are pure or contain fewer observations than `min_sample_split`, which is set equal to 2 by default.

Despite the fact that building trees in RFs to full-depth has largely become the standard advice offered, the fact remains that numerous studies have provided strong empirical evidence that tuning tree depth can substantially impact performance. Segal [2004] performed a number of experiments on both synthetic and real-world data utilizing various tree depths and feature subsampling rates and found significant changes in accuracy, but no clear trends in terms of which settings were optimal in which settings. Lin and Jeon [2006] put forth the idea of adaptive nearest neighbors to better characterize the class of models to which RFs belong and, as related to the idea of tree depth, argued that "*growing the largest tree was not optimal in general*". Duroux, Roxane and Scornet, Erwan [2018] carried out simulations directly comparing the performance of Breiman's original RFs to RFs constructed with shallow trees and reached much the same conclusion. The popular graduate-level textbook "The Elements of Statistical Learning" [Hastie et al., 2009] takes more of a neutral, measured approach, saying only that "*the average of fully grown trees can result in too rich a model*" but that "*Our experience is that using full-grown trees seldom costs much, and results in one less tuning parameter*".

Despite the substantial amount of previous literature on the topic, to the best of our

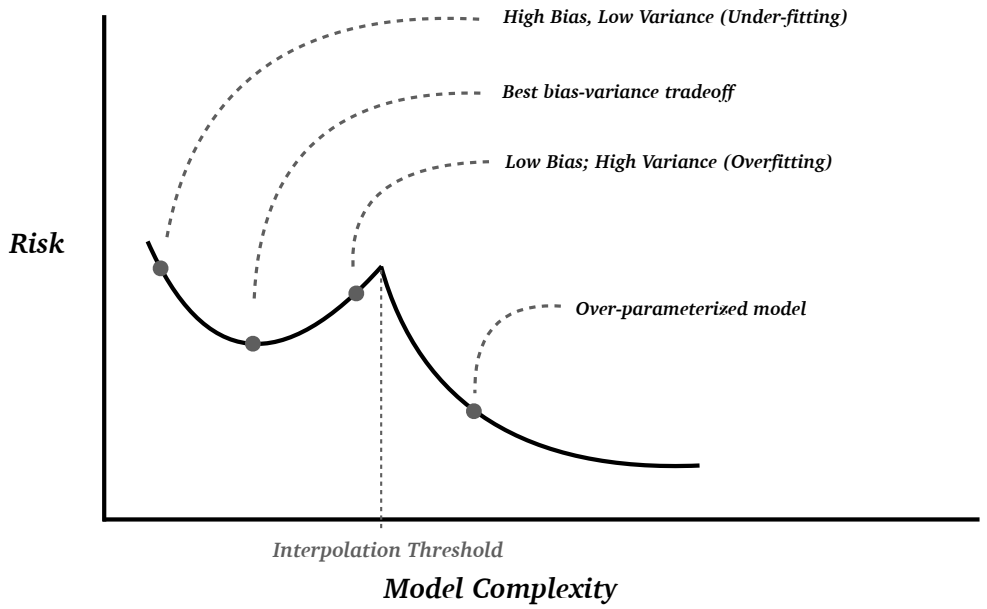

Figure 4.1: Graphical visualization of the "double descent" proposed in Belkin et al. [2019]. Before the interpolation threshold we see the classical U-shaped bias-variance tradeoff, followed by a second descent when considering over-parameterized models. Note that if the model complexity of a tree is measured by the number of terminal nodes, then a tree cannot be over-parameterized.

knowledge, none of this work has offered a principled explanation as to *why* limiting tree-depth in RFs could be helpful or *when* such an alternative construction might be expected to outperform Breiman's original proposal. In the following sections, we aim to take a step forward in this direction by building upon the regularization framework in the last chapters. To begin, we first revisit the double descent argument put forth in Belkin et al. [2019] and demonstrate that in the context of random forests, this second drop in risk is not only achievable, but expected, even when shallow decision trees are employed and interpolation is exceedingly unlikely.

4.3 RANDOM FORESTS AND DOUBLE DESCENT

In work that has since gained a lot of attention, Belkin et al. [2019] studied the the relationship between performance (risk) and complexity for a variety of supervised learning models. Classical statistical theory suggests that such a curve should exhibit a natural U shape: models with complexity too low will have too much bias and under-fit, models with too much complexity will have too much variance and over-fit, and thus the optimal model that minimizes this curve should be that which optimally trades off bias and variance. Much to the surprise of many in the statistics community, however, Belkin et al. [2019] noticed that for a variety of such models, the risk curve can begin a second descent once the complexity crosses beyond the interpolation threshold and becomes over-parameterized – see Figure 4.1. In some cases, this second descent achieves minimum values that exceed even those obtained by the model believed to be optimal by classical theory – that which minimizes the first descent. Put simply, the work suggested that the poor performance often seen in parameter-rich, highly-complex models could be mitigated not only by removing parameters, but also by adding them. Belkin et al. [2019] attributed this second descent to the argument that despite the large model complexity of interpolating functions, their function space norm is smaller and thus they can be seen as "simpler" through an alternative lens.

Before continuing, it is worth pausing to more carefully define the notions of interpolation and model complexity as these are clearly of fundamental importance to the argument put forth in Belkin et al. [2019]. Interpolating functions are simply those whose training error is zero, as defined in Definition 1. The notion of interpolation is straightforward, easily defined, and can pertain to both classification and regression problems. The interpolation threshold is then identified as the minimum complexity at which the model begins to interpolate.

This notion of model *complexity*, on the other hand, is far more nuanced. Belkin et al. [2019] define the complexity of a model as "*the number of parameters needed to specify a function within the class*". Such a definition is natural and intuitive for simple models like ordinary least squares (OLS) linear regression or individual decision trees and is consistent with notion of *the effective number of parameters* discussed in Hastie et al. [2009]. Note that if "model complexity" of tree-based models is taken as the number of leaves, then by con-

struction this cannot exceed the training size and thus a tree cannot be over-parameterized. Indeed, this notion of complexity becomes less straightforward for black-box models. With random forests, for example, Belkin et al. [2019] applies this definition in something of a hybrid fashion: for random forests containing only a single tree, complexity is taken as the number of leaves so that the original (training) sample size n serves as an upper bound. Beyond this point, however, the complexity of ensembles of full-depth trees seems to be measured as (at least proportional to) the number of trees in the ensemble. This is where the natural intuition behind such a definition begins to break down. While averaging B trees can potentially partition the feature space in a finer way (and thus utilize more "parameters") relative to a forest with $B-1$ trees, it seems a bit unorthodox to assume that the former estimator is inherently more "complex".

A natural analogy can be drawn here to classical U-statistics [Hoeffding, 1948]. Recall that the standard motivation for such estimators is as follows. Given a sample $Z_1, ..., Z_n$ of size n and a parameter of interest θ, we assume there exists some unbiased estimator $h(Z_1, ..., Z_k)$ utilizing only $k \leq n$ arguments and that h is permutation symmetric in those k arguments. This base estimator h is generally referred to as a kernel of rank k. While any subsample of size k from the original sample will suffice to produce an unbiased estimate of θ, not surprisingly, a better estimator, and indeed, that which has the minimum variance, is the U-statistic

$$U_n = \frac{1}{\binom{n}{k}} \sum_{(n,k)} h(Z_1^*, ..., Z_k^*)$$

formed by evaluating h over all $\binom{n}{k}$ possible subsamples and averaging. When $B < \binom{n}{k}$ subsamples are utilized, the resulting estimator is referred to as an *incomplete* U-statistic.

Note that when the individual kernels h are seen as (possibly randomized) decision trees, there is an immediate connection between U-statistics and random forests. In fact, it was this connection that was exploited by Mentch and Hooker [2016] to demonstrate that RF predictions are asymptotically normal when trees are constructed via subsampling instead of traditional bootstrap samples.

This connection also helps make clear the shortcomings of the interpolation-based argument for why a second descent is observed when random forests begin consisting of more

than one tree. Decades-old statistical theory tells us that U-statistic-style estimators have the same bias (zero) as the original estimator h but with smaller variance and would thus be preferred according to the classic bias-variance tradeoff. By near-identical reasoning, RFs containing many trees built with resamples of the original data ought to be preferred to individual decision trees. Seen in this fashion, it makes little intuitive sense to define such an estimator as more "complex" merely because it takes an average over a larger collection.

Even more importantly for our purposes here though, estimators constructed by taking a larger average should be preferred *regardless* of the subsample size (rank) k. Just as a U-statistic formulation would be expected to result in an improved estimator regardless of the rank of the kernel, random forests with many trees should be preferable to individual decision trees regardless of the subsample size on which they're built. That is, even when relatively shallow trees are constructed so that complexity falls well short of the interpolation threshold, a second drop in random forest risk should be expected as trees are added to the ensemble. In some classification settings or in settings where trees are each constructed on the same original sample it may be possible to interpolate (at least nearly), but interpolation is not the *cause* of this.

In the following subsection, we elaborate on this point by replicating the analysis in Belkin et al. [2019] before arguing that this kind of shallow-tree construction is actually preferable in noisy data settings in Section 4.4.

4.3.1 A Case Study on the MNIST Dataset

Some of the data employed by Belkin et al. [2019] to demonstrate the double descent curve was from the MNIST database, which contains data (60,000 training samples and 10,000 test samples) extracted from handwritten numeric digits. In order to replicate these kinds of experiments in a computationally efficient manner, we begin by randomly sampling a training set $\mathcal{D}_{\text{Train}}$ from the original training set, as well as test and validation sets $\mathcal{D}_{\text{Test}}$ and $\mathcal{D}_{\text{Val}}$ from the original test set, each of size $n = 2000$. As the response, we use a binary indicator corresponding to whether or not the handwritten digit is a '1'.

In this subsection, we refer to the model complexity of RFs as tree depth/number of

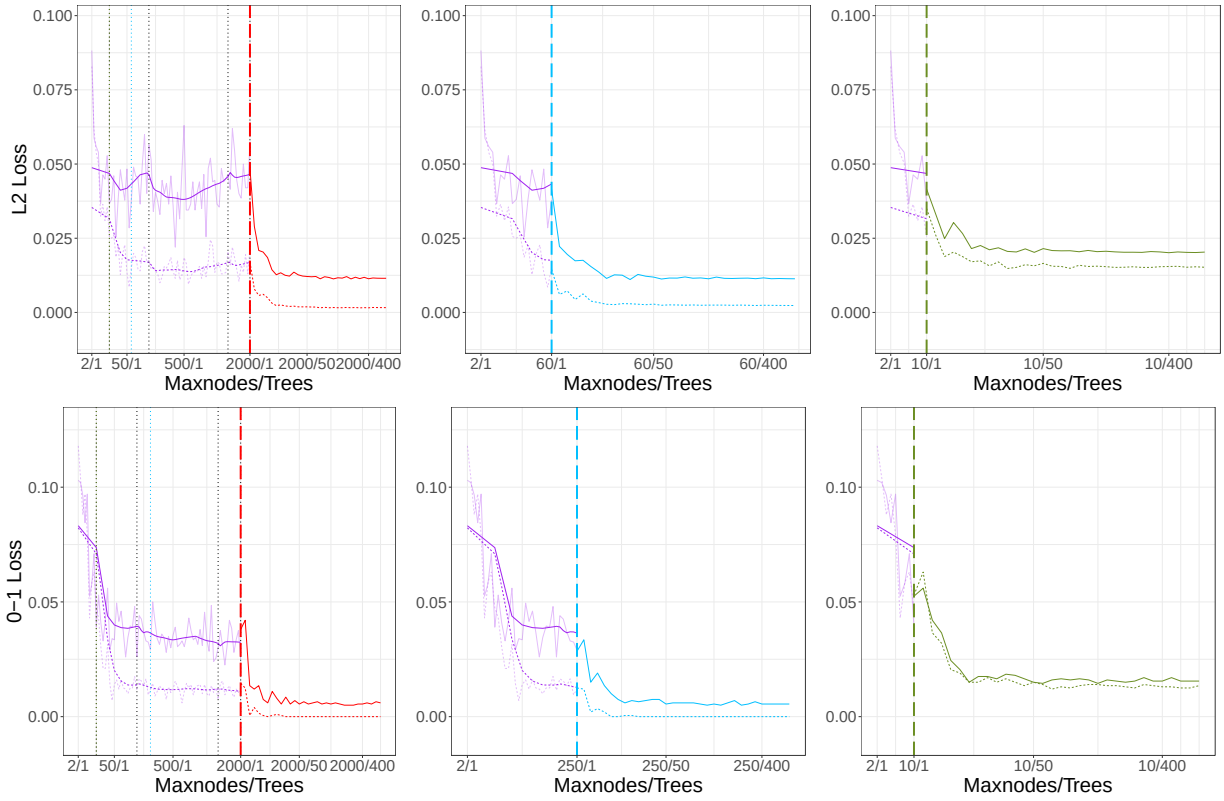

Figure 4.2: Performance of a single randomized tree (purple curves), full-depth RFs (left column, red curves), tuned RFs (middle column, blue curves) and shallow RFs (right column, green curves) using **bootstrap samples** in regression (top row) and classification (bottom row) settings. The transparent purple background curve is the raw result of a single randomized tree; the bold purple line corresponds to a lowess smooth. Dashed and solid curves correspond to the performance on training and test sets, respectively.

trees so as to be consistent with definitions and plots in Belkin et al. [2019]. Tree depth is controlled by two parameters, `maxnodes` and `nodesize`. The `maxnodes` parameter can range from 2 (a stump) to 2000 (one observation in each leaf) and we set the `nodesize` parameter to 1 so that an internal node will be continue to be split until the number of leaves is equal to `maxnodes`.

We consider a total of four different model setups:

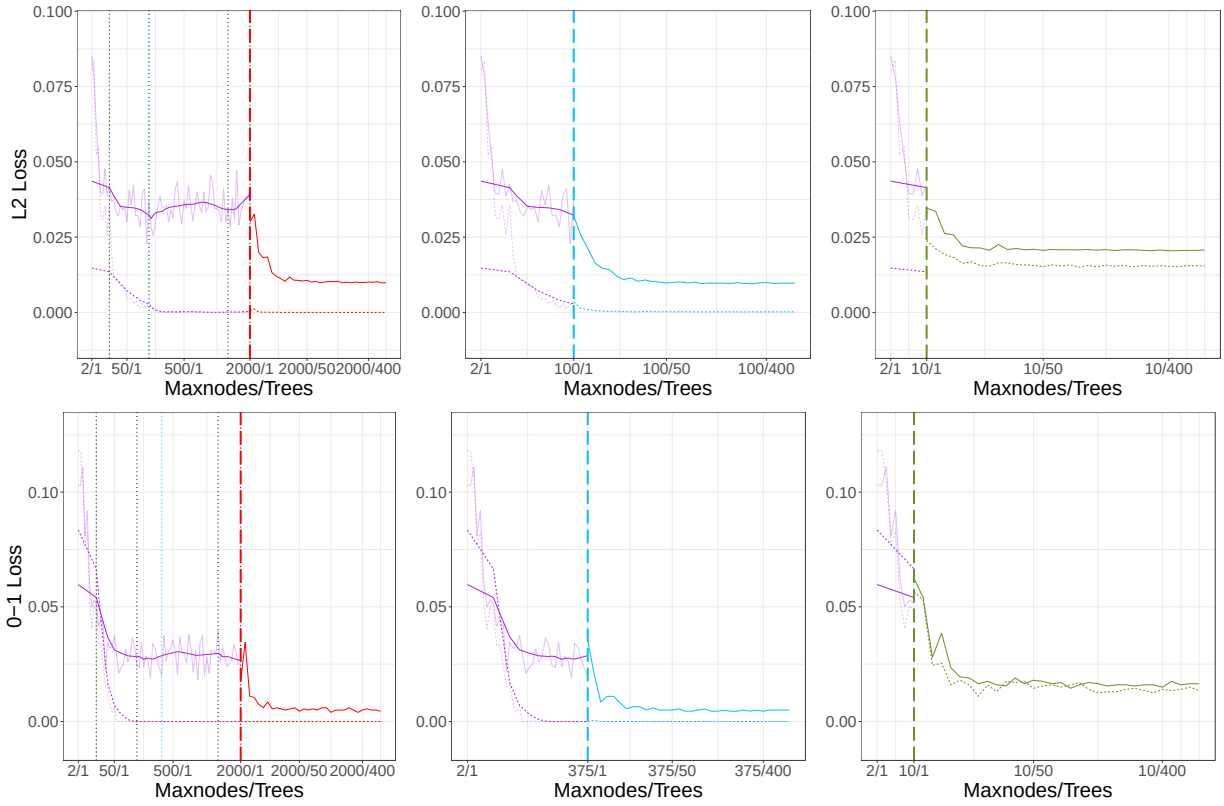

Figure 4.3: Performance of a single randomized tree (purple curves), full-depth RFs (left column, red curves), tuned RFs (middle column, blue curves) and shallow RFs (right column, green curves) using **original samples** in regression (top row) and classification (bottom row) settings. The transparent purple background curve is the raw result of a single randomized tree; the bold purple line corresponds to a lowess smooth. Dashed and solid curves correspond to the performance on training and test sets, respectively.

- **Individual Trees:** A single randomized tree with the number of `maxnodes` ranging from 2 to 2000.
- **Full-depth Forests:** 500 trees constructed with the `maxnodes` set equal to 2000 so that all trees are full depth.
- **Tuned Forests:** 500 trees with `maxnodes` set equal to the depth $\texttt{maxnodes}_{opt}$ that minimizes the error of a single (randomized) tree on the validation set.

- **Shallow Forests:** 500 trees constructed with `maxnodes` = 10.

Note that each setup utilizes randomized trees as in a typical random forest. For each setup, we consider constructing trees via bootstrap samples (as traditionally done) and also by using the same original training set each time (as was done in Belkin et al. [2019]). All remaining parameters are set to the default values in the `R` package `randomForest`. We consider both classification and regression setups. In the classification setup, both tree- and forest-level predictions are made via majority vote and performance on the test set is measured by 0-1 loss. In the regression setting, we utilize regression trees so that the binary responses are averaged at both the tree and forest level and performance is measured via the standard L_2 (squared error) loss.

Results from forests constructed with bootstrap samples and original samples are shown in Figures 4.2 and 4.3 respectively. In each figure, the top row corresponds to the regression setup and the bottom to the classification setup. As in Belkin et al. [2019], model complexity is shown on the horizontal axis and model performance on the vertical axis with the training loss and test loss represented by dashed and solid lines, respectively. The four colors in the plots represent the performance of the four setups under consideration: purple for individual trees, red for full-depth RFs (left column), blue for tuned RFs (middle column), and green for shallow RFs (right column). Randomization in the single tree leads to a great deal of variation in the performance so at the level of individual trees, we also include a bolder purple line corresponding to a lowess smooth. The bold vertical long-dashed lines in each plot denote the transition points from a single tree to a RF with multiple trees. In the left-most column of each figure, there are additional black vertical dotted lines in each plot that are included to indicate that model complexity between these lines is equally spaced.

Plots in the left-most column of each figure are designed to replicate the kind of analysis done in Belkin et al. [2019]. Here we see that once the interpolation threshold is reached and more trees are added to the forest (red dashed vertical line), the performance of full-depth RFs improves and eventually levels off indicating what Belkin et al. [2019] describe as the second descent. A close inspection of the upper-left-hand plots in Figures 4.2 and 4.3 is also very revealing. Note that in Figure 4.2, where bootstrapping is employed, the training error never reaches 0 (the RFs do not interpolate) and yet, once more trees are

added, a near identical drop in test error is observed compared to the no-bootstrap case (Figure 4.3) where trees are constructed on the same original data each time and 0 training error is achieved quite quickly. Furthermore, note that tuned RFs (middle column) perform almost identically to their full depth counterparts. That is, even though the training loss is not 0 at the transition point from single trees to ensembles of trees, a rapid performance improvement is still obvious once more trees added. Once again, in the right-most plots, the shallow RFs again exhibit a dramatic drop in loss when more trees are added despite the fact that the training error remains well above 0.

These demonstrations make clear that while we do indeed observe a dramatic drop in loss with full-depth trees once multiple trees are considered – what Belkin et al. [2019] refers to as the "interpolation threshold" – near identical performance improvement jumps are seen regardless of the training error whenever averaging over many trees as opposed to considering only a single tree. Indeed, the idea that averaging (or otherwise aggregating) unstable estimators to reduce the variance of the procedure is not a new idea (see e.g. Breiman [1996], Bühlmann et al. [2002]) nor is it an idea that is in any way dependent on interpolation of the individual base models. Nonetheless, when comparing the test performance of shallow RFs (right-most column) to their counterparts in Figures 4.2 and 4.3, it's clear that these ensembles of shallow learners are not performing as well as the "deeper" ensembles in the left and middle columns. One may thus be tempted to argue, as continues to be claimed in many textbooks, that ensembles of deeper trees are still preferable nonetheless. In the following section, we dispel this idea and demonstrate that in noisy data settings, ensembles of shallow trees are indeed preferable.

4.4 RANDOM FORESTS AND TREE DEPTH

Our work in Chapter 2 provides theoretical background for the regularization effect of random feature-subsetting in ensembles of base models constructed in a greedy fashion. With the same setup, the regularizing effect of model size (depth) for such randomized ensembles can also be demonstrated. In particular, in the context of RandFS, given an orthogonal design

matrix, the process of averaging across randomized linear models of depth d is equivalent to constructing a linear model that includes all p of the original covariates, but where each OLS coefficient estimate $\hat{\beta}_{j,\text{OLS}}$ is shrunk by some amount $\alpha_{j,d}$. Even more importantly for the conversation on model depth, note that the magnitude by which we shrink the coefficient estimates is directly related to the model size (depth) d: as d becomes smaller, each model must include fewer covariates and thus some covariates must necessarily appear in less models, leading to more shrinkage on their corresponding coefficient estimates. Note also that this shrinkage is not applied uniformly across all covariates; covariates more correlated with the response will have a higher chance of being selected once made available and thus, on average, will be shrunk less than covariates with a weaker relationship to the response. In these ways, this kind of randomized linear model ensemble behaves similarly to procedures like ridge regression and lasso that take an explicitly penalized approach to fitting.

Finally, note that in noisy data settings where the signal is relatively low, this kind of shrinkage can be advantageous by preventing overfitting to noise. Since RFs are constructed in the same kind of fashion by using trees instead of linear models, we ought to expect the same kind of outcome: ensembles of shallow trees ought to perform better in noisier settings. We now explore this empirically via a number of simulations and experiments.

4.4.1 Tree Depth as Regularization

Here we follow closely the simulation setups utilized in recent empirical studies, including those in Hastie et al. [2020] and Chapter 2. We assume the standard linear model relationship $Y_i = \boldsymbol{X}_i^\top \beta + \epsilon_i$ where the data arrive as n i.i.d. ordered pairs $(\boldsymbol{X}_i, Y_i)$ where $\boldsymbol{X}_i \in \mathbb{R}^p$ is a vector of p covariates sampled from $N_p(0, \Sigma)$ and $Y_i \in \mathbb{R}$ is the response. Here, the $(i,j)^{th}$ entry of Σ is of the form $\rho^{|i-j|}$ and ρ is set to 0.35. The first s terms of the coefficient vector β are set equal to 1 and the rest to 0, corresponding to the "beta-type 2" setup in Hastie et al. [2020]. The noise terms ϵ_i are independent of the covariates and are sampled i.i.d. from $N(0, \sigma^2)$ where σ^2 is chosen to create the desired signal-to-noise ratio (SNR) given by

$$\text{SNR} = \frac{\beta^\top \Sigma \beta}{\sigma^2}.$$

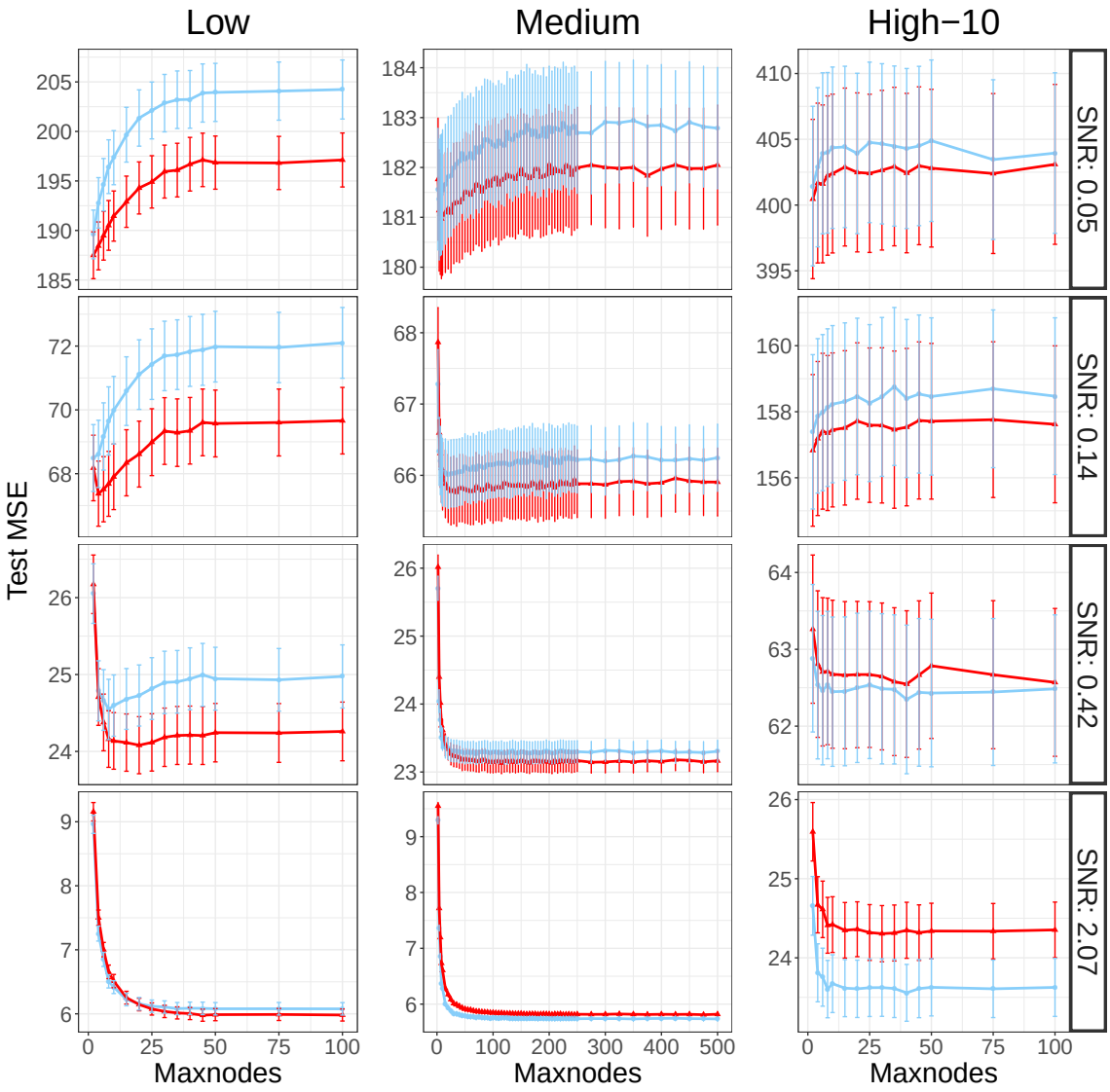

Figure 4.4: Performance of random forests (red) and bagging (blue) in the low (left column), medium (middle column) and high-10 settings (right column) at different SNR levels. The horizontal axis is `maxnodes`, the maximum number of nodes in each tree and vertical axis is the test MSE. Vertical bars denote one standard error.

The use of linear models here allows us to easily calculate the SNR explicitly but more importantly, previous works have shown that the relative performance of RFs with respect to SNR is quite similar across regression relationships, even when they contain interactions

and nonlinearities. Thus, utilizing more complex underlying relationships here is unnecessary for investigating the desired properties.

As in previous studies, we consider 10 SNR values ranging from 0.05 to 6, equally spaced on log scale. Under these conditions, we then consider the following four setups:

- **Low**: $n = 100$, $p = 10$, $s = 5$
- **Medium**: $n = 500$, $p = 100$, $s = 5$
- **High-5**: $n = 50$, $p = 1000$, $s = 5$
- **High-10**: $n = 100$, $p = 1000$, $s = 10$

The RFs are constructed using the `randomForest` package in `R` with the `nodesize` parameter set equal to 1. We use the `maxnodes` parameter as a proxy for tree depth and allow it to take values $\{2, 4, 6, 8, 10, 15, \ldots, n/2, n/2+25, n/2+50, \ldots, n\}$. We consider both bagging and traditional RF setups so that the `mtry` parameter is set to either p or $p/3$, respectively. Each ensemble consists of 500 trees and all remaining parameters are set to their default values. Model performance is measured as the mean squared error (MSE) evaluated on an independently generated test set the same size as the training set. At each individual setting, the entire process is repeated 100 times and the mean and standard error of the test MSEs across these repetitions are reported.

Results for the low, medium and high-10 setups at SNR levels of 0.05, 0.14, 0.42, and 2.07 are shown in Figure 4.4 where the vertical bars represent one standard error; full results across all setups and SNR levels are given in the Appendix C. In both sets of Figures, the blue curve corresponds to the results from bagging and the red to RFs. If we look at the two higher SNR levels in the low setting (Figure 4.4, left-most column, bottom two plots), we see the pattern most would expect for both bagging and random forests – as the trees in the ensemble grow deeper (`maxnodes` increases), the accuracy improves and eventually begins to level off. Indeed, if this were the case across all settings, the conventional wisdom of simply growing trees in an RF to full depth would seem accurate; there would be no need to waste additional computational resources trying to optimize tree depth. As predicted by the theory presented in the beginning of this section, however, this is not the case. At the two lowest SNR levels in the low setting, we see the opposite pattern – shallow trees

perform very well and as the depth of the trees *increases*, the error increases as well before beginning to level off. Much the same story can be seen in the medium setup (Figure 4.4, middle column) except that the switch to a preference for deeper trees appears to occur at a lower SNR. In the high-10 setup (Figure 4.4, right-most column), a similar pattern is discernible, though as would be expected in high-dimensional settings, there appears to be a higher variance as well. Overall across all settings, we see the same general behavior in both bagging and random forests.

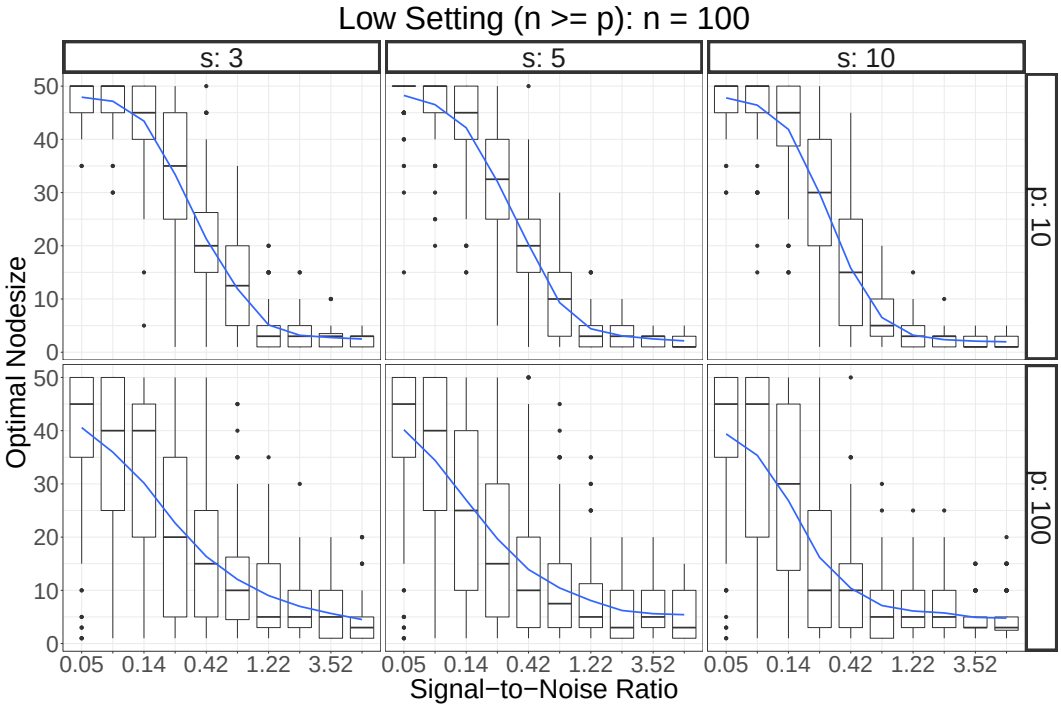

Figure 4.5: Boxplots of optimal `nodesize` of RFs in the Low setting ($n = 100$) over 100 repetitions.

The results in Figure 4.4 demonstrate how RF performance changes as a function of tree depth across a variety of regression setups and SNR levels. We now formulate the question in a slightly different fashion: given a particular data setup, what tree depth minimizes the resulting error of a RF at various SNR levels? Here we use `nodesize` as the proxy for tree depth and allow trees to grow to the maximal possible depth subject to the `nodesize` constraint. Specifically, we consider the following three setups:

- **Low** ($n \geq p$)**:** $n = 100$: $p = 10$ or 100, and $s = 3$, 5 or 10; `nodesize` takes values 1, 3, and 10 additional values equally spaced between 5 and $n/2$.
- **Medium** ($n \geq p$)**:** $n = 500$, $p = 10$ or 100, and $s = 3$, 5 or 10; `nodesize` takes values 1, 3, and 25 additional values equally spaced between 5 and $n/2$.
- **High** ($n \leq p$)**:** $n = 50$ or 100, $p = 1000$, and $s = 5$, 10 or 20; `nodesize` takes values 1, 3, and 10 additional values equally spaced between 5 and $n/2$.

For each setting, we consider the same linear model setups and SNR values as above. At each combination of settings, we obtain and record the optimal `nodesize`, defined as that which minimizes the MSE on an independent test set with 1000 observations. The process is repeated 100 times and boxplots of the optimal node sizes at each SNR level in the low, medium, and high settings are shown in Figures 4.5, C1, and C2, respectively, with the latter two figures appearing in the Appendix C.

The results here are very much in-line with what would be expected based on the theory and experiments above. In particular, in the low and medium settings, regardless of the number of signal covariates s, the same general pattern is clear. At lower SNRs, the optimal `nodesize` is relatively large, meaning that terminal nodes in each tree often contain many observations and the trees are therefore more shallow. As the SNR levels increase, there is an obvious transition to a preference for a smaller `nodesize` (deeper trees). In the high setting (Figure C2), once again we see substantially more variance in the results, though in the larger n case ($n = 100$), a downward trend is still evident.

As alluded to above, this finding is quite intuitive: as the quality of the data improves (SNR levels rise), it makes sense that we would want to extract as much information from it as possible by growing deeper trees. On the other hand, when the data is of poor quality (low SNR levels), this kind of overfitting can be dangerous and we might prefer to keep only the strongest, most evident patterns seen in early splits of the tree.

4.4.2 Tunability

Much of the work above examines the relationship between tree depth and RF accuracy – ensembles of shallow trees are generally preferred with noisy data while those with deeper

trees tend to be more advantageous with high quality data. We now begin to more carefully examine and quantify the potential benefits of tuning – how much improvement can result from tuning the tree depth compared with simply growing full depth trees and how does the benefit arising from tuning tree depth compare to that realized from tuning the `mtry` parameter.

To investigate this, we apply the "tunability" framework proposed by Probst et al. [2019a]. Let $\hat{f}(\boldsymbol{X}, \theta)$ denote the prediction of a model $\hat{f}$ with a $K-$dimensional tuning parameter $\theta = (\theta_1, \ldots, \theta_K) \in \Theta$ at $\boldsymbol{X}$. Let $L(Y, \hat{f}(\boldsymbol{X}, \theta))$ denote the loss function. Given a new test point $(\boldsymbol{X}_0, Y_0)$ so that the risk of the model with respect to θ is defined by $R(\theta) = \mathbb{E}(L(Y_0, \hat{f}(\boldsymbol{X}_0, \theta))$ with the expectation taken over the unknown distribution of $(\boldsymbol{X}_0, Y_0)$. Finally, let $\theta_0 = (\theta_{0,1}, \ldots, \theta_{0,K})$ denote default values of θ – these may correspond, for example, to those specified as defaults in software packages or based on empirical results from previous experiments.

Define

$$\theta^\sharp = \text{argmin}_{\theta \in \Theta} R(\theta),$$

as the optimal value of θ corresponding to the minimum risk. The *tunability* of the model $\hat{f}$ was defined in Probst et al. [2019a] as the difference in performance between the model built with default tuning parameter values and that constructed with the optimal values

$$d = R(\theta_0) - R(\theta^\sharp).$$

Further, for any $S \subset \{1, \ldots, K\}$, we can define

$$\theta^\star_S = \text{argmin}_{\theta \in \Theta, \theta_l = \theta_{0,l} \ \forall l \notin S} R(\theta)$$

as the optimal tuning parameter values whenever those parameters with indices in S are optimally tuned and the remainder are set to their default values so that the resulting tunability value is given by

$$d_S = R(\theta_0) - R(\theta^\star_S).$$

Finally, for any $T \subset S$, the additional performance gain in tuning all parameters with indices in S instead of tuning only those with indices in either T or $S - T$ is given by

$$g_{T,S-T} = \min\left(R(\theta_T^\star), R(\theta_{S-T}^\star)\right) - R(\theta_S^\star).$$

In the following, we focus on the RF context and investigate the tunability of tree depth (controlled by `maxnodes`) and the tunability of the additional randomness `mtry` so that the tuning space is given by $\Theta = (\texttt{mtry}, \texttt{maxnodes})$. Trees are grown using bootstrap samples and the number of trees is large enough (500 in each setting) to ensure that the performance of the ensemble has stabilized. We carry out simulations using the same linear model setup as above with the same four size setups under consideration: low, medium, high-5 and high-10. In each setting, we construct RFs using the `R` package `randomForest` with `nodesize` $= 1$ and the entire procedure is repeated 100 times at each SNR level. The `mtry` parameter is tuned over $0.1p, 0.2p, \ldots, p$, and the `maxnodes` parameter is tuned over $\{2, 4, 6, 8, 10, 15, \ldots, n/2, n/2 + 25, n/2 + 50, \ldots, n\}$. At each $(\texttt{mtry}, \texttt{maxnodes})$ combination, we measure the mean square error (MSE) of RF on an independent test set the same size as dataset used in training and define the relative test error (RTE) as

$$\text{RTE}(\texttt{mtry}, \texttt{maxnodes}) = \frac{1}{\sigma^2} \text{ Test MSE}(\texttt{mtry}, \texttt{maxnodes}).$$

We set the default tuning parameter values to align with those in most softwares as $\texttt{mtry}_0 = 0.3p$ and $\texttt{maxnodes}_0 = n$. In keeping with the notation above, define

$$\begin{aligned}
(\texttt{mtry}^\sharp, \texttt{maxnodes}^\sharp) &= \text{argmin}_{(\texttt{mtry},\texttt{maxnodes})\in\Theta} \text{RTE}(\texttt{mtry}, \texttt{maxnodes}) \\
(\texttt{mtry}^\star, \texttt{maxnodes}_0) &= \text{argmin}_{\substack{(\texttt{mtry},\texttt{maxnodes})\in\Theta; \\ \texttt{maxnodes}=\texttt{maxnodes}_0}} \text{RTE}(\texttt{mtry}, \texttt{maxnodes}) \\
(\texttt{mtry}_0, \texttt{maxnodes}^\star) &= \text{argmin}_{\substack{(\texttt{mtry},\texttt{maxnodes})\in\Theta; \\ \texttt{mtry}=\texttt{mtry}_0}} \text{RTE}(\texttt{mtry}, \texttt{maxnodes}).
\end{aligned}$$

The tunability of the combination of *both* `mtry` and `maxnodes` at the same time, which we refer to as the tunability of RFs, is then given by

$$d_{\text{RF}} = \text{RTE}(\texttt{mtry}_0, \texttt{maxnodes}_0) - \text{RTE}(\texttt{mtry}^\sharp, \texttt{maxnodes}^\sharp)$$

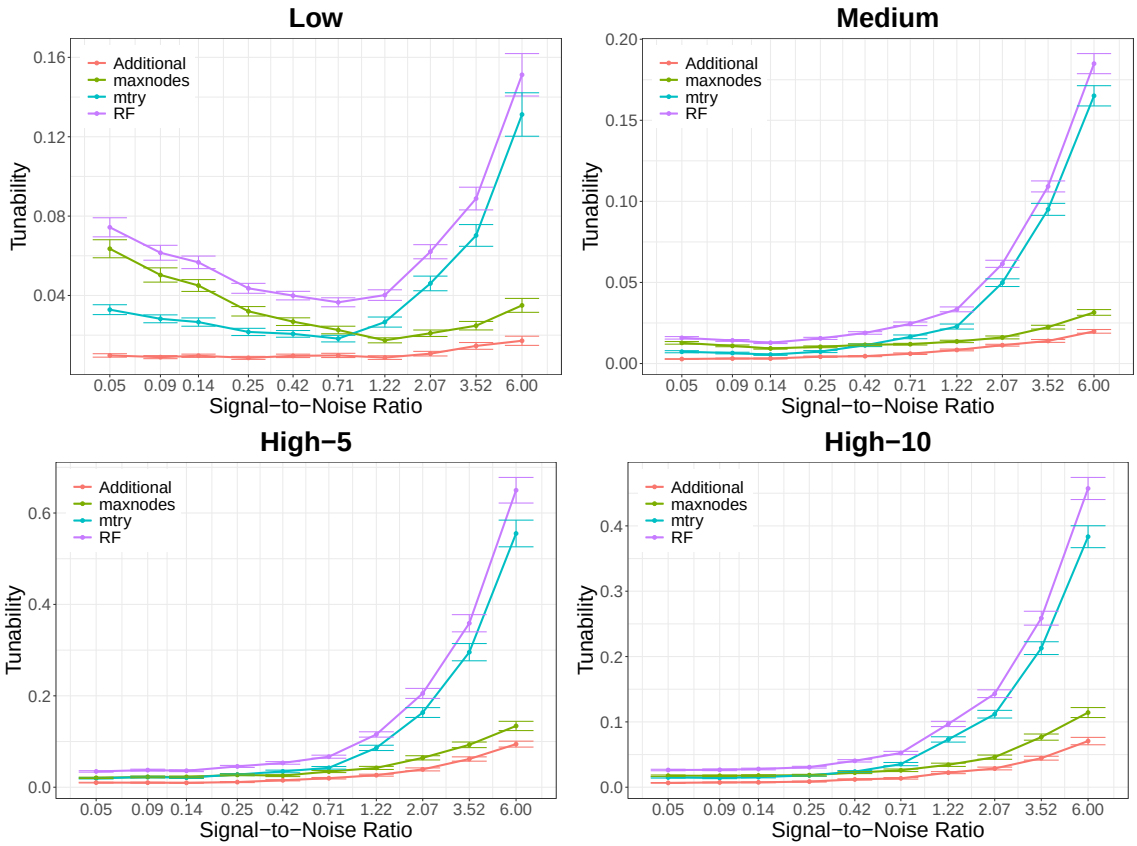

Figure 4.6: Average tunability of random forests d_{RF} (purple), tunability of `maxnodes` $d_{\texttt{maxnodes}}$ (green), tunability of `mtry` $d_{\texttt{mtry}}$ (blue) and additional improvement of tuning both instead of either one $g_{\texttt{mtry,maxnodes}}$ (red) at the 10 SNR levels across the 100 repetitions. Default `mtry` $= 0.3p$ and default `maxnodes` $= n$. Error bars represent one standard error.

and the (marginal) tunability of `mtry` and `maxnodes` individually is given by

$$d_{\texttt{mtry}} = \text{RTE}(\texttt{mtry}_0, \texttt{maxnodes}_0) - \text{RTE}(\texttt{mtry}^\star, \texttt{maxnodes}_0)$$
$$d_{\texttt{maxnodes}} = \text{RTE}(\texttt{mtry}_0, \texttt{maxnodes}_0) - \text{RTE}(\texttt{mtry}_0, \texttt{maxnodes}^\star),$$

respectively. Finally, the additional performance gain seen by tuning both `maxnodes` and

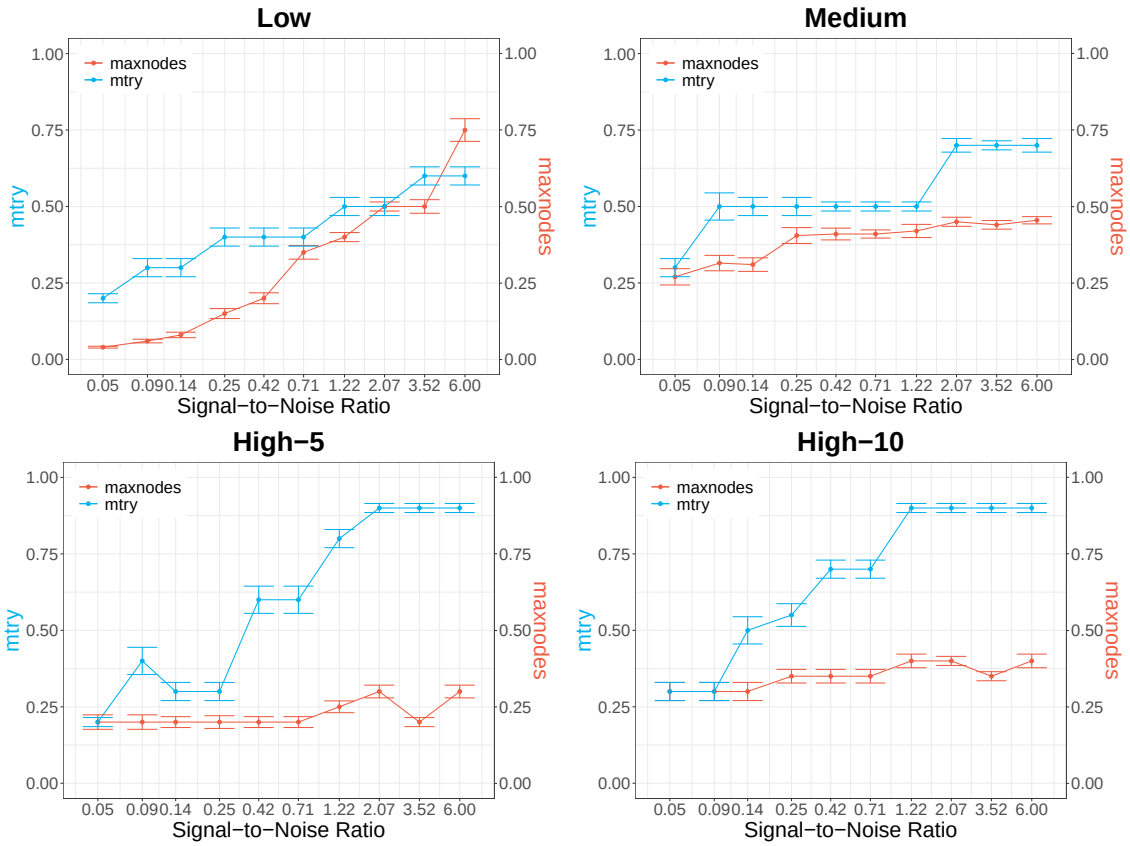

Figure 4.7: Median of optimal combination of `mtry` (blue) and `maxnodes` (red) in 100 repetitions at different SNR levels. Error bars represent median absolute deviation.

`mtry` rather than only one of them is given by

$$g_{\texttt{mtry},\texttt{maxnodes}} = \min\left(\mathrm{RTE}(\texttt{mtry}^\star, \texttt{maxnodes}_0), \mathrm{RTE}(\texttt{mtry}_0, \texttt{maxnodes}^\star)\right) - \mathrm{RTE}(\texttt{mtry}^\sharp, \texttt{maxnodes}^\sharp).$$

Figure 4.6 shows plots of average tunability across each set of 100 repetitions with error bars representing one standard error – corresponding boxplots are shown in Figure C7 in the Appendix C.

Looking at the Low setting (Figure 4.6, upper left), we see a nice transition in the benefits of tuning depth (`maxnodes`) vs `mtry`. If we were to tune only one of these parameters, tuning `maxnodes` would be preferred at low SNRs and tuning `mtry` would yield more substantial

gains at high SNRs. A few other trends are evident in each of the four plots in Figure 4.6. First, the benefits to tuning `mtry` and `maxnodes` together always yields a better result than tuning only one of them individually. Further, the increasing red line in each plot indicates that the additional benefit to tuning both parameters together tends to increase with the SNR.

We close out this subsection by looking at how the optimal combination of `mtry` (as a proportion of p) and `maxnodes` (as a proportion of n) changes with SNR. For each of the 100 repetitions at each setting, we record the optimal combination of tuning parameter values – the medians of these are shown in Figure 4.7 where error bars represent median absolute deviation. In all settings, the optimal combination of `mtry` and `maxnodes` increases as SNR increases, though notice that in the medium and high settings with larger p, the optimal tree depth remains only a relatively small fraction of n even at the highest SNRs.

4.4.2.1 Real Datasets We now investigate the tunability of `mtry` and `maxnodes` on real-world datasets, which presumably contain more complex underlying relationships between the features. In total, we include 8 low dimensional datasets from the UCI Machine Learning Repository [Dua and Graff, 2017] and 4 high dimensional datasets, 3 from openml.org [Vanschoren et al., 2013] and 1 (AquaticTox) from `R` package `QSARdata`. Observations with missing values were removed. Table 4.1 provides a summary of each dataset.

Tunability is obtained based on 10-fold cross validation (CV) error scaled by the sample variance of the corresponding response. As above, RFs are constructed using the package `randomForest` with 500 trees grown on bootstrap samples with `nodesize` set equal to 1. For each dataset, the `mtry` parameter is tuned over $\lceil 0.1p \rceil, \lceil 0.2p \rceil, \ldots, \lceil p \rceil$ whenever $p \geq 10$; for datasets with $p < 10$, `mtry` is tuned over $\{1, \ldots, p\}$. Since depth (`maxnodes`) is related to the available training size and 10-fold CV is applied, `maxnodes` is tuned over 25 values equally spaced between $\lfloor \frac{1}{25} 0.9n \rfloor$ and $\lfloor 0.9n \rfloor$. For the `phen` dataset, `maxnodes` is instead tuned over $\{1, \ldots, \lfloor 0.9n \rfloor\}$. Default values of `mtry` and `maxnodes` are set to $\lceil 0.3p \rceil$ and $\lfloor 0.9n \rfloor$. For each dataset, the procedure is repeated for 100 times.

Figure 4.8 shows boxplots of $d_{\text{RF}}, d_{\texttt{mtry}}, d_{\texttt{maxnodes}}$ and $g_{\texttt{mtry,maxnodes}}$ for the 12 real-world datasets. The first two rows correspond to low dimensional datasets and the last row to

Table 4.1: Summary of the real-world datasets utilized. The text in brackets following each dataset name serves as an abbreviation used in future discussions and plots. Numeric references for each dataset are given either to the work where the data first appeared, or to early work where it was utilized. The optimal values of (`maxnodes`$^\sharp$, `mtry`$^\sharp$) are given in the final column as proportions of the number of observations and features (n and p), respectively.

Dataset	p	n	(`maxnodes`$^\sharp$, `mtry`$^\sharp$)
Bike Sharing [`bike`] [Fanaee-T and Gama, 2014]	11	731	(0.76, 0.55)
Boston Housing [`boston`] [Harrison Jr and Rubinfeld, 1978]	13	506	(0.76, 0.54)
Concrete Compressive Strength [`concrete`] [Yeh, 1998]	8	1030	(0.76, 0.88)
CPU Performance [`cpu`] [Ein-Dor and Feldmesser, 1987]	7	209	(0.67, 0.86)
Conventional and Social Movie [`csm`] [Ahmed et al., 2015]	10	187	(0.72, 0.4)
Facebook Metrics [`fb`] [Moro et al., 2016]	7	499	(0.72, 0.86)
Servo System [`servo`] [Quinlan, 1993]	4	167	(0.72, 1)
Solar Flare [`solar`] [Li et al., 2000]	10	1066	(0.61, 1)
Aquatic Toxicity [`AquaticTox`] [He and Jurs, 2005]	468	322	(0.72, 0.60)
Weighted Holistic Invariant Molecular Descriptor [`pah`] [Todeschini et al., 1995]	112	80	(0.73, 0.71)
Adrenergic Blocking Potencies [`phen`] [Cammarata, 1972]	110	22	(0.68, 0.65)
PDGFR Inhibitor [`pdgfr`] [Guha and Jurs, 2004]	320	79	(0.68, 0.1)

high dimensional ones. For the low dimensional datasets, the tunability of `mtry` (blue) is lower than that of `maxnodes` (green) in only one dataset (`csm`), but higher in the remaining seven. In three of these, the tunability of `mtry` is comparable to d_{RF} (purple), the joint tunability of `mtry` and `maxnodes`, suggesting that there is little more to gain by tuning both `mtry` and `maxnodes` as opposed to tuning only `mtry`. With high dimensional datasets, the tunability of `mtry` and `maxnodes` is more similar across datasets. As expected, tuning both `mtry` and `maxnodes` together leads to improved performance in every case, though in most cases, that additional improvement is relatively small in comparison to the gains seen by tuning only one. What is consistent across all 12 datasets is that the additional improvement $g_{\texttt{mtry,maxnodes}}$ is much smaller than $d_{\texttt{mtry}}$ or $d_{\texttt{maxnodes}}$, the respective improvement from tuning `mtry` or `maxnodes` alone, and is a small fraction of d_{RF}, improvement from tuning `mtry` and `maxnodes` together. Thus, as suggested with the above simulations, in practice, one could often tune `mtry` with full depth trees and tune `maxnodes` with `mtry` set to the default value, and choose the model with minimum generalization error among these two candidates. Such an approach will likely be far more computationally efficient than a full

two-dimensional grid-search and yield a model with similarly small error. Finally, Table 4.1 shows the optimal combination of `mtry` as a proportion of p and `maxnodes` as a proportion of n for each dataset. Note that the optimal depth (`maxnodes`) does not generally correspond to full-depth trees.

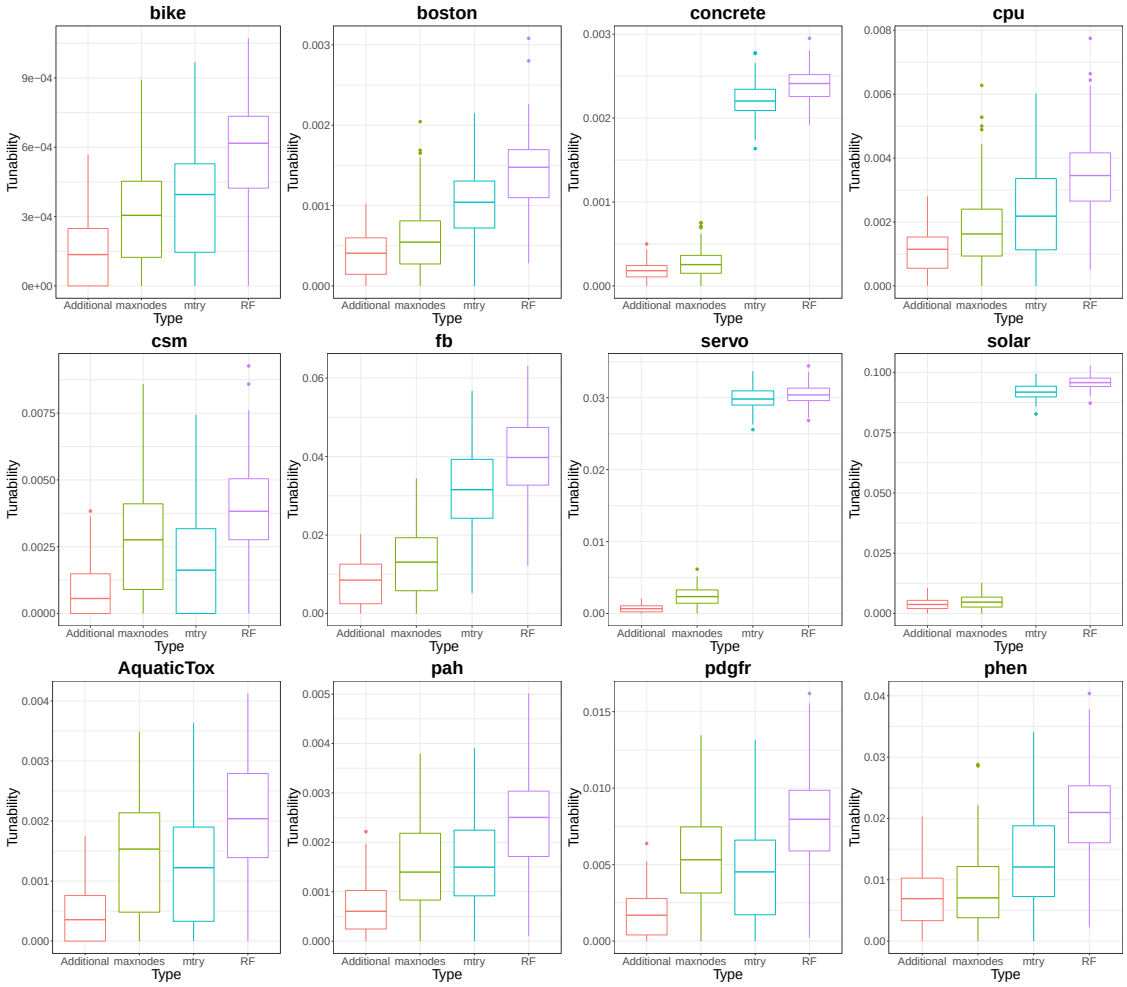

Figure 4.8: Boxplots of $d_{\text{RF}}, d_{\texttt{mtry}}, d_{\texttt{maxnodes}}$ and $g_{\texttt{mtry,maxnodes}}$ in the 100 repetitions for 12 real datasets. The first two rows correspond to low dimensional datasets and the last row to high dimensional datasets. Default `mtry` $= \lceil 0.3p \rceil$ and default `maxnodes` $= \lfloor 0.9n \rfloor$. The color coding remains the same as in Figure 4.6.

4.5 DISCUSSION

In this chapter, we have sought to provide a principled and in-depth study into the relationship between tree depth and RF performance. We argued that the substantial performance gains seen when transitioning from individual trees to ensembles of trees is a natural and expected by-product of averaging high-variance base models and showed that such improvements are seen even when significant training error exists and models are far from the interpolation threshold. We then gave both theoretical and empirical evidence that optimal tree depth is ultimately a function of of the level of noise in the data. In that sense, limiting tree depth can be seen simply as a natural form of regularization obtained by limiting model complexity.

It is important to stress, however, that bagging and random feature subsetting – both hallmarks of classical RFs – also perform a kind of regularization. Thus, in practice, these "built-in" RF regularizers may be sufficient to achieve optimal performance without needing to result to the more computationally intensive task of tuning tree depth. This, we suspect, is why performance gains are not always seen in practice and, as a result, is likely why many textbooks continue to suggest constructing RFs with full-depth trees. Nonetheless, this work makes clear that this is not universally the best strategy and indeed, significant performance gains *can* be had by limiting tree depth, especially when the `mtry` parameter is not tuned. Regardless of how tuning is ultimately performed, because it can sometimes have a substantial impact, whenever one is interested in studying variable importance as measured by changes in predictive accuracy when features are removed or randomized, one must be sure to carefully tune the model first before examining any measures of feature importance.

5.0 CONCLUDING REMARKS

Random forests have a well-earned reputation as an excellent off-the-shelf statistical learning method with outstanding success across almost all scientific domains. Numerous efforts and studies have been devoted to their mathematical and statistical properties such as consistency, asymptotic normality, variance of predictions and so on. Despite this, reasons for random forests' success remained largely mysterious.

When proposing random forests, Breiman [2001] provided an upper bound for the generalization error of random forests which involves accuracy of individual trees and between-tree correlations, the tradeoff between which is similar to the classical bias-variance tradeoff from the statistical perspective. However, this is more a motivation for why random forests can *potentially* work well than an explanation for why they *do*. Another informal but popular intuition given in Biau and Scornet [2016] is that aggregating tree models is able to estimate complex patterns beyond those specified by smoothness or sparsity conditions, but it's not clear when random forests are expected to perform well.

Another attention-drawing idea for explaining success of modern learning models is interpolation. Wyner et al. [2017] attributed the success of random forests to their being "self-averaging interpolators" and isolating noisy observations. Belkin et al. [2019] put forth the more general idea of "double descent" with empirical evidence for modern models including neural networks and random forests. Specifically, the curve of model error against complexity is in a U shape in the under-parameterized regime, but it decreases again in the over-parameterized regime, i.e., when model complexity goes beyond the interpolation threshold.

However, the relationship between interpolation and success of random forests is like a chicken-egg problem: do random forests succeed *because* of interpolation or is interpolation a

side-effect of the success of random forests? If the former is true, then random forests ought to be expected to perform poorly on noisy datasets, which, however, is the opposite to what is observed in practice. Apart from this, in the regression setting, random forests can not interpolate unless individual trees are all constructed using the original sample. Moreover, by drawing parallel to U-statistics, the second descent in the random forests error curve results from simple averaging, and is expected even with an ensemble of shallow trees which is exceedingly unlikely to interpolate. Last but not least, arguments around interpolation ignores the randomness in random forests and applies equally well to bagging.

What makes Breiman's random forests special is the randomness injected in each tree split, without which the procedure reduces to bagging that is commonly taken as inferior to random forests. Instead of naively assuming random forests always outperform bagging, we take a step back to try and fully characterize when the inclusion of such randomness leads to improved performance and why. We have produced strong evidence that the amount of randomness utilized is negatively correlated to the degrees of freedom of the model. In another word, this random-feature-selection provides an implicit form of regularization, making random forests more advantageous in noisy settings. Moreover, this is not a tree-specific phenomenon, but applies equally well to ensembles with base learners constructed in a greedy fashion, such as linear models selected by forward selection but where only a randomly selected subset of features are available as candidates.

Along this line, we continue to explore other forms of regularization that can be introduced to ensemble of trees to produce similarly competitive procedures. Surprisingly and counterintuitively, this can be achieved by inclusion of additional noisy features conditionally independent of the response. Moreover, this has a crucial implication on testing variable importance. One commonly used proxy of variable importance is to compare the performance of model when features of interest are included v.s. excluded from the model. The intuition is that if dropping a set of features leads to significantly decreased performance, then those features must contain some information about the response. However, our work clearly demonstrates that at least with some procedures, additional features – even when purely noise and completely unrelated to the response – can lead to substantially more accurate models, particularly in noisy settings, and be registered as statistically significant

features. Thus, we advocate for alternative forms of testing that focus instead on determining whether the original features produce models that are substantially more accurate than those produced via random substitutes that preserve the between-feature relationship.

Following the regularization framework above, we further investigate the effect of pruning trees in random forests. Although full depth trees are recommended in many textbooks [James et al., 2013, Izenman, 2008] and set as default settings in standard packages such as Scikit-learn [Pedregosa et al., 2011] in `python` and `randomForest` [Liaw et al., 2002] in `R`, we argue that tree depth can be viewed as an additional form of regularization on top of bagging and random-feature-subsetting, with shallow trees more preferred in noisy settings. With enough computational power, it is best to tune both randomness and tree depth. If we were to tune only one of these parameters, our empirical experiments suggest tuning tree depth would be preferred at low SNRs and tuning randomness would yield more substantial gains at high SNRs. In practice, to avoid high computational cost, one could tune randomness with full depth trees and tune depth with default randomness and choose the model with best performance. Despite this, regardless of how tuning is ultimately performed, whenever one is interested in studying variable importance as measured by changes in predictive accuracy when features are removed or randomized, one must be sure to carefully tune the model first before examining any measures of variable importance.

APPENDIX A

ADDITIONAL SIMULATIONS IN CHAPTER 2

In Section 2.3, Figure 2.3 shows the estimated degrees of freedom for random forests across various values of `mtry` in four different regression setups at a fixed SNR of 3.52. In each case we see that the dof increases with both `maxnodes` and `mtry` and we note that the same relationships were seen to also emerge in alternative setups. Figure A1 below shows the same experiments carried out with the SNR set to the much smaller level of 0.09 and indeed the findings remain the same.

In Section 2.4.2 we considered the problem of estimating the optimal value of `mtry` for random forests for both the linear and MARS models at various SNR levels. In the main text, Figure 2.5 shows plots of the optimal `mtry` measured as that which obtains the lowest average test error over 500 iterations. In contrast, Figure A2 here calculates the optimal `mtry` value on each of the 500 iterations and then takes the empirical mean. The results in Figure A2 show the same general pattern as seen in Figure 2.5. As the SNR increases, so does the optimal value of `mtry`.

Figure A3 shows exactly the same plots as in Figure 2.8 but here we add error bars at each point corresponding to ± 1 standard deviation across the 100 replications. These are reserved for the appendix only for readability as the error bars can make the plots a bit more muddled and difficult to make out.

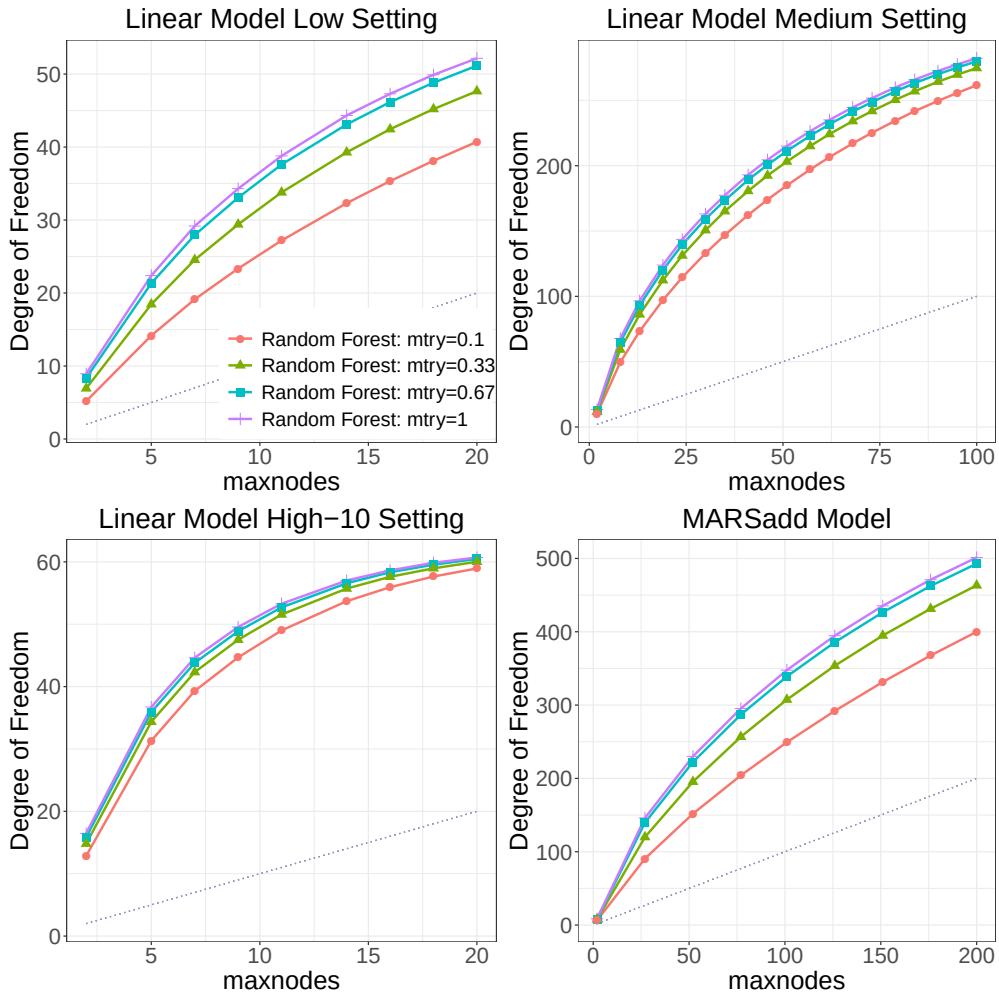

Figure A1: Degrees of freedom for random forests at different levels of `mtry` with a SNR of 0.09.

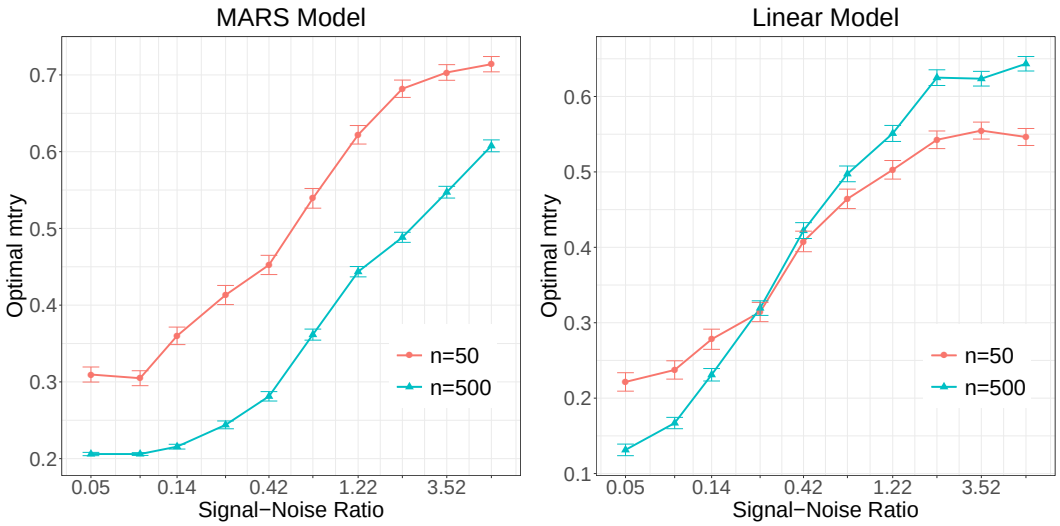

Figure A2: Optimal value of `mtry` vs SNR for the MARS and linear model.

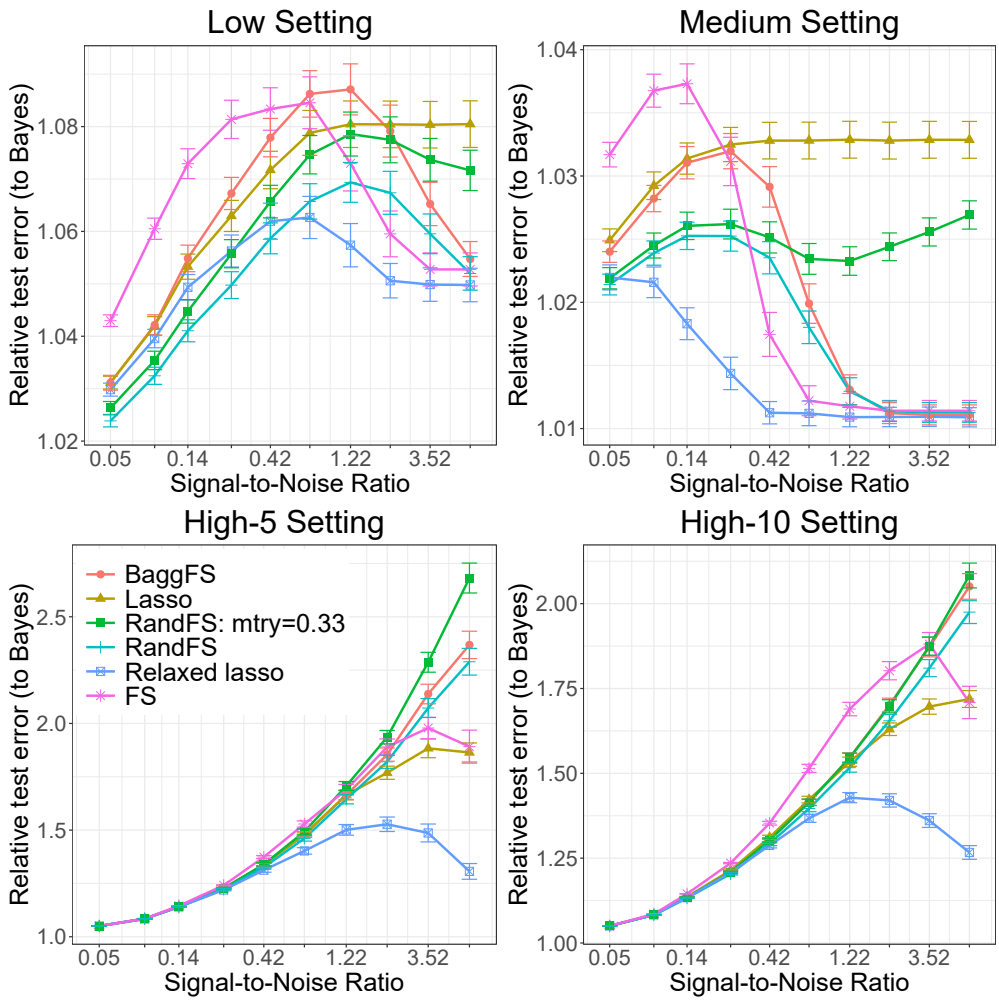

Figure A3: Performance of FS, BaggFS, RandFS, lasso and relaxed lasso across SNR levels for linear models in the low, medium, high-5, and high-10 settings. Error bars at each point correspond to ± 1 standard deviation across the 100 replications.

APPENDIX B

INTRINSIC TESTING AND EXTRINSIC ANALOGUES IN CHAPTER 3

In this appendix, we review the testing procedure developed in Williamson et al. [2021]. For readability, we begin by looking at the extrinsic analogue to the authors' intrinsic test and show that it produces test results similar to those seen in Section 3.5.1. We then go on to discuss the updates necessary in order to consider that test intrinsic.

To being, suppose that we have 3 independent datasets: a training set $\mathcal{D}_n$ of size n, a test set $\mathcal{D}_{Test,1}$ of size n_1 and another test set $\mathcal{D}_{Test,2}$ of size n_2, containing i.i.d. observations as a generic random vector $\boldsymbol{Z} = (Y, \boldsymbol{X})$ where $\boldsymbol{X} = (X_1, \ldots, X_{p+q})$ and the response $Y \in \mathbb{R}$. For simplicity, we further assume that $n_1 = n_2 = n'$. Let $\boldsymbol{X}^*$ denote the modified random vector where the last q features $(X_{p+1}, \ldots, X_{p+q})$ in $\boldsymbol{X}$ are either dropped or replaced with a random substitutes. Thus, as in the main text, for "drop tests", $\boldsymbol{X}^*$ will be of length p whereas for "replacement tests", $\boldsymbol{X}^*$ will be of length $p + q$.

Let $\hat{f}$ and $\hat{f}^*$ be the ensemble (bagged) estimates on the original data $(Y, \boldsymbol{X})$ and on the modified data $(Y, \boldsymbol{X}^*)$, respectively. Let $f = \mathbb{E}(\hat{f})$ and $f^* = \mathbb{E}(\hat{f}^*)$ where the expectation is over the training set $\mathcal{D}_n$. Define

$$MSE(\hat{f}) = \frac{1}{n'} \sum_{i \in \mathcal{D}_{Test,1}} (Y_i - \hat{f}(\boldsymbol{X}_i))^2,$$

$$MSE(\hat{f}^*) = \frac{1}{n'} \sum_{i \in \mathcal{D}_{Test,2}} (Y_i - \hat{f}^*(\boldsymbol{X}_i^*))^2,$$

$$T = MSE(\hat{f}) - MSE(\hat{f}^*)$$

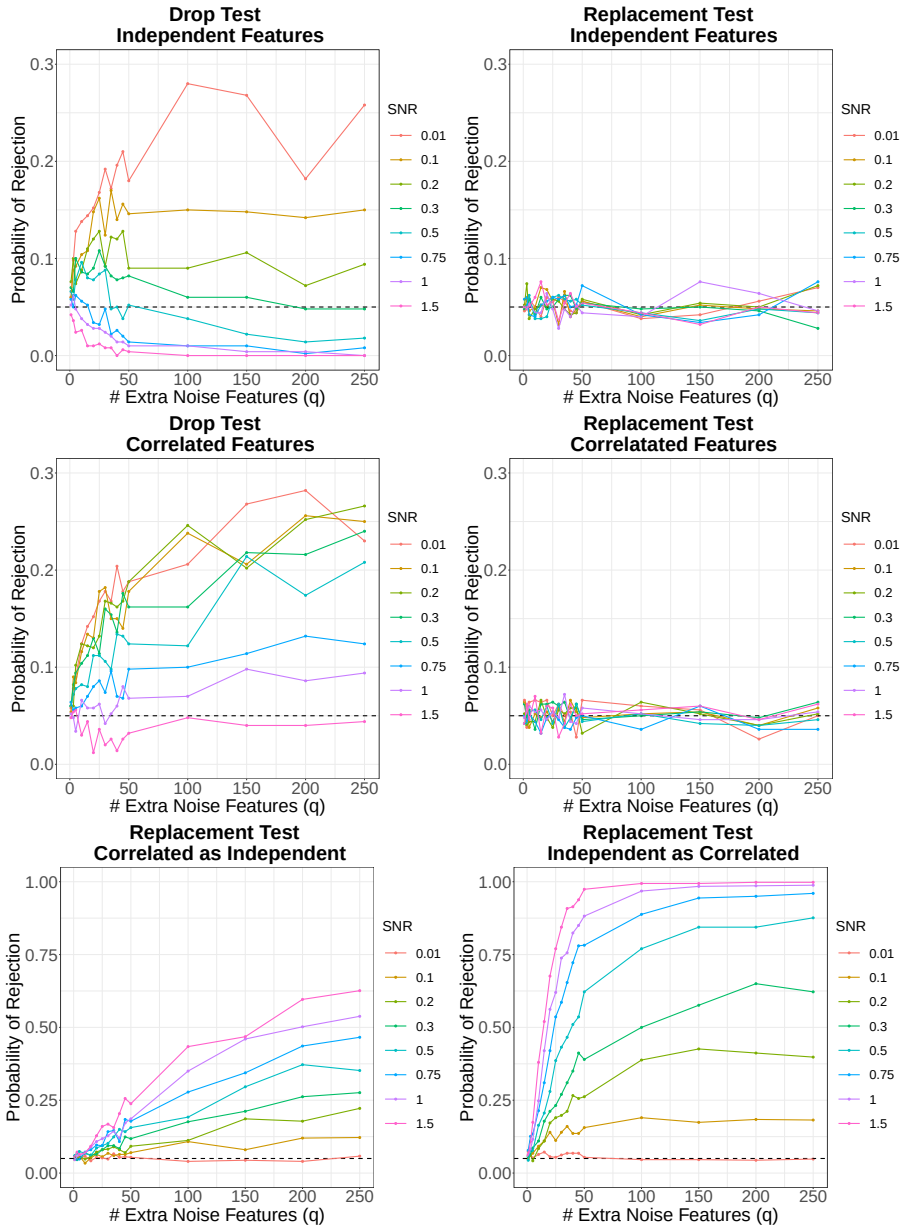

Figure B1: Probability of rejecting the null hypothesis and concluding an additional independent set of noise features are important when dropping the features in question (left column, top two rows) vs replacing the features in question (right column, top two rows) when those features are independent (top row) vs correlated (middle row) and replacement features are sampled from the original underlying distribution (top two rows) In the bottom row, features are replaced by samples from a different distribution.

where $MSE(\hat{f})$ and $MSE(\hat{f}^*)$ are independent since $\mathcal{D}_{Test,1}$ and $\mathcal{D}_{Test,2}$ are independent.

By the central limit theorem,

$$\sqrt{n'}\left(MSE(\hat{f}) - \mathbb{E}\left[(Y - f(\boldsymbol{X}))^2\right]\right) \to_d N\left(0, \text{Var}\left((Y - f(\boldsymbol{X}))^2\right)\right),$$
$$\sqrt{n'}\left(MSE(\hat{f}^*) - \mathbb{E}\left[(Y - f^*(\boldsymbol{X}^*))^2\right]\right) \to_d N\left(0, \text{Var}\left((Y - f^*(\boldsymbol{X}^*))^2\right)\right)$$

where the expectations are with respect to the corresponding test set. Let

$$\Delta = \mathbb{E}\left[(Y - f(\boldsymbol{X}))^2\right] - \mathbb{E}\left[(Y - f^*(\boldsymbol{X}^*))^2\right], \tag{B.1}$$
$$\sigma^2 = \text{Var}\left((Y - f(\boldsymbol{X}))^2\right) + \text{Var}\left((Y - f^*(\boldsymbol{X}^*))^2\right). \tag{B.2}$$

Then by independence of $MSE(\hat{f})$ and $MSE(\hat{f}^*)$,

$$\sqrt{n'}\left(T - \Delta\right) \to_d N\left(0, \sigma^2\right)$$

and σ^2 can be estimated with

$$\hat{\sigma}^2 = s_1^2 + s_2^2 \tag{B.3}$$

where s_1^2 and s_2^2 are the empirical variances of $(Y_1 - \hat{f}(\boldsymbol{X}_i))^2$, $i \in \mathcal{D}_{Test,1}$ and $(Y_1 - \hat{f}^*(\boldsymbol{X}_i^*))^2$, $i \in \mathcal{D}_{Test,2}$, respectively. Here we want to classify the features of interest as important whenever the test MSE is larger in the second ensemble where those features are either dropped or replaced, and thus the null and alternative hypotheses of interest are $H_0 : \Delta = 0$ and $H_1 : \Delta < 0$, respectively.

Figure B1 shows the results of applying this extrinsic tests under identical setups to those described in Section 3.5.1. Note that the top two rows in Figure B1 correspond to the plots in Figure 3.5 and the two plots in the bottom row of Figure B1 correspond to those in Figure 3.6. As noted above, the patterns here are qualitatively quite similar. In Figure B1, we see that these rejection rates are not quite as large in the correlated case with the drop test (middle row, left) compared with those in Figure 3.5 (bottom left). On the other hand, the rates here are larger across a wider range of SNRs in the independent feature setting (top row, left) as compared with those in Figure 3.5 (top left). Rejection rates in the replacement tests do not appear to be inflated above the nominal level of 0.05 (Figure B1 top and middle

row, right), and as before, rejection rates are far above the nominal level when features are replaced with those from a different distribution (Figure B1 bottom row).

We stress again that the test applied here is extrinsic, and thus it should come as no surprise that these results are much in keeping with those seen in Section 3.5.1. To turn this into an intrinsic test, we need to find the influence function to estimate the asymptotic variance.

Let P_0 denote the population distribution of $\boldsymbol{Z} = (Y, \boldsymbol{X})$ and for simplicity, denote $\mathbb{E}_{P_0} = \mathbb{E}_0$. Suppose our measure of predictiveness is negative MSE so that we can define $V(f, P_0) = -\mathbb{E}_0(Y - f(\boldsymbol{X}))^2$ and we have population maximizers $f_0(\boldsymbol{X}) = \mathbb{E}_0(Y|\boldsymbol{X}) := \mu_0(\boldsymbol{X})$ and $f_{0,s}(\boldsymbol{X}) = \mathbb{E}_0(Y|\boldsymbol{X}_{-s}) := \mu_{0,s}(\boldsymbol{X}_{-s})$.

Let $\delta_{\boldsymbol{z}}$ denote the degenerate distribution on $\{\boldsymbol{z}\}$. Denote by $\dot{V}(f, P_0; h)$ the Gâteaux derivative of $P \mapsto V(f, P)$ at P_0 in the direction of h. By definition,

$$\dot{V}(f, P_0; h) = \lim_{\tau \to 0} \frac{1}{\tau} \left[V(f, P_0 + \tau h) - V(f, P_0) \right].$$

The influence function corresponding to f_0 is defined to be $\phi_0 : \boldsymbol{z} \mapsto \dot{V}(f_0, P_0; \delta_{\boldsymbol{z}} - P_0)$ and $\phi_{0,s}(\boldsymbol{z})$ can be defined similarly for $f_{0,s}$. Following the discussion in page 6 in Williamson et al. [2021], with equal-size splitting, the asymptotic variance will be of the form of $\eta_0^2 + \eta_{0,s}^2$ where $\eta_0^2 := \mathbb{E}_0(\phi_0(\boldsymbol{Z}))^2$ and $\eta_{0,s}^2 := \mathbb{E}_0(\phi_{0,s}(\boldsymbol{Z}))^2$ and these terms can be estimated separately on the two test sets $\mathcal{D}_{Test,1}$ and $\mathcal{D}_{Test,2}$. Let $P_\tau(\boldsymbol{z}) := P_0 + \tau(\delta_{\boldsymbol{z}} - P_0) = \tau\delta_{\boldsymbol{z}} + (1 - \tau)P_0$. Then we have

$$\begin{aligned} V(f_0, P_\tau(\boldsymbol{z})) &= -\mathbb{E}_{P_\tau(\boldsymbol{z})}(Y - f_0(\boldsymbol{X}))^2 = -\left[\tau\, \mathbb{E}_{\delta_{\boldsymbol{z}}}(Y - f_0(\boldsymbol{X}))^2 + (1 - \tau)\, \mathbb{E}_0(Y - f_0(\boldsymbol{X}))^2\right] \\ &= -\left[\tau(y - f_0(\boldsymbol{x}))^2 + (1 - \tau)\, \mathbb{E}_0(Y - f_0(\boldsymbol{X}))^2\right] \end{aligned}$$

and so

$$\begin{aligned} \phi_0(\boldsymbol{z}) &= \lim_{\tau \to 0} \frac{1}{\tau} \left[V(f_0, P_\tau(\boldsymbol{z})) - V(f_0, P_0) \right] \\ &= \lim_{\tau \to 0} \frac{1}{\tau} \left[-\left[\tau(y - f_0(\boldsymbol{x}))^2 + (1 - \tau)\, \mathbb{E}_0(Y - f_0(\boldsymbol{X}))^2\right] + \mathbb{E}_0(Y - f_0(\boldsymbol{X}))^2 \right] \\ &= \mathbb{E}_0(Y - f_0(\boldsymbol{X}))^2 - (y - f_0(\boldsymbol{x}))^2. \end{aligned}$$

Thus, $\eta_0^2 = \mathbb{E}_0(\phi_0(\boldsymbol{Z}))^2 = \text{Var}((Y - f_0(\boldsymbol{X}))^2)$ can be estimated by the empirical variance s_1^2 on the test set $\mathcal{D}_{Test,1}$. Similarly, $\phi_{0,s}(\boldsymbol{z}) = \mathbb{E}_{0,s}(Y - f_{0,s}(\boldsymbol{X}))^2 - (y - f_{0,s}(\boldsymbol{x}))^2$, and $\eta_{0,s}^2 = \mathbb{E}_0(\phi_{0,s}(\boldsymbol{Z}))^2 = \text{Var}((Y - f_{0,s}(\boldsymbol{X}))^2)$ can be estimated by the empirical variance s_2^2 on the test set $\mathcal{D}_{Test,2}$. Thus, estimated variance of the difference in test MSE based on the influence function is the same as equation (B.3) in the extrinsic version, thus showing the direct correspondence between the intrinsic and extrinsic versions of these tests.

APPENDIX C

ADDITIONAL SIMULATIONS IN CHAPTER 4

Figures C1 and C2 show the relationship between optimal `nodesize` and SNR in the medium and high settings, respectively, described in Section 4.4.1. Figures C3–C6 show RF performance vs `maxnodes` across the full range of all 10 SNRs under investigation in the low, medium, high-5, and high-10 settings also described in Section 4.4.1. Note that Figure 4.4 was designed to show a representative subset of these and was included for the purpose of preserving space in the main text. Figure C7 show boxplots of tunability.

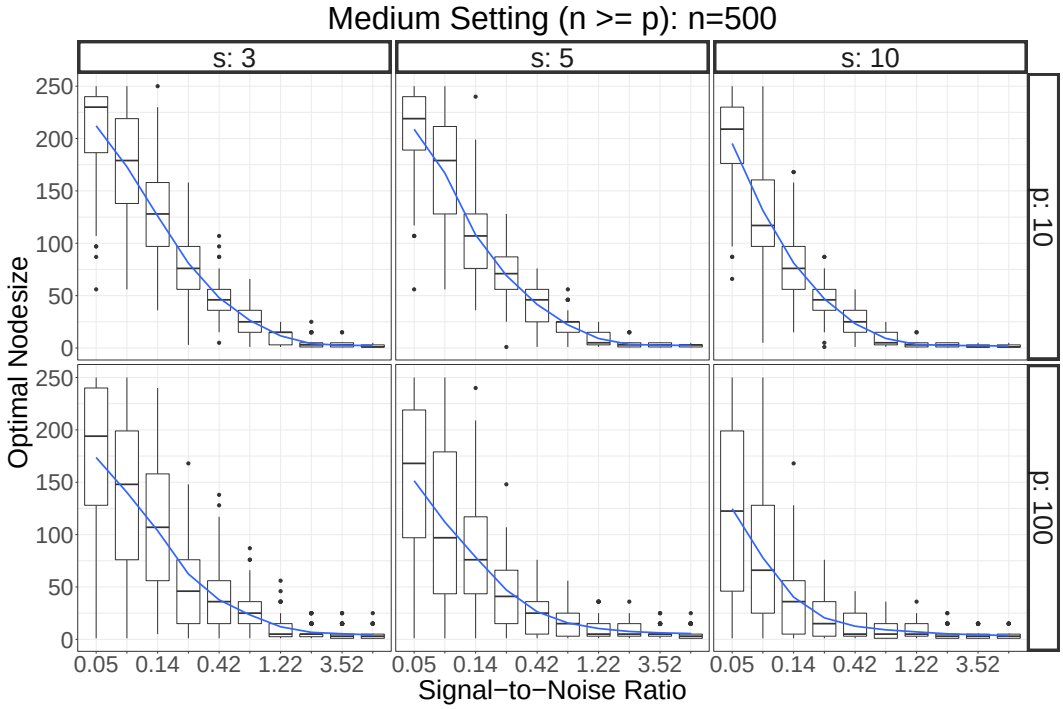

Figure C1: Boxplots of optimal `nodesize` of RFs in the Medium setting ($n = 500$) over 100 repetitions.

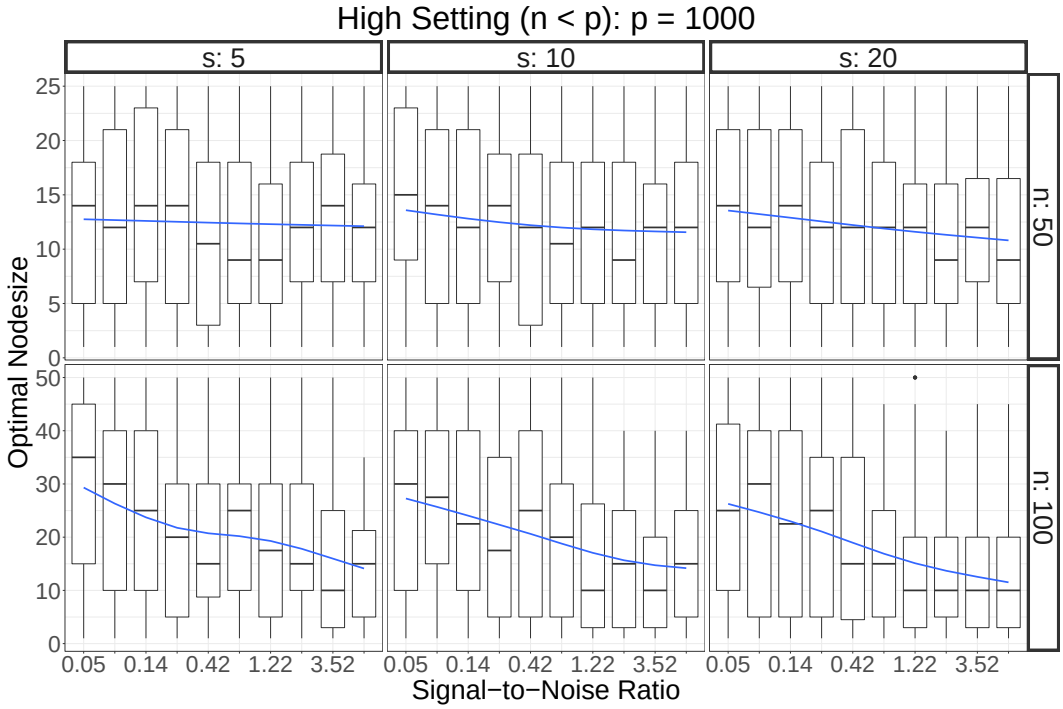

Figure C2: Boxplots of optimal `nodesize` of RFs in the High setting ($p = 1000$) over 100 repetitions.

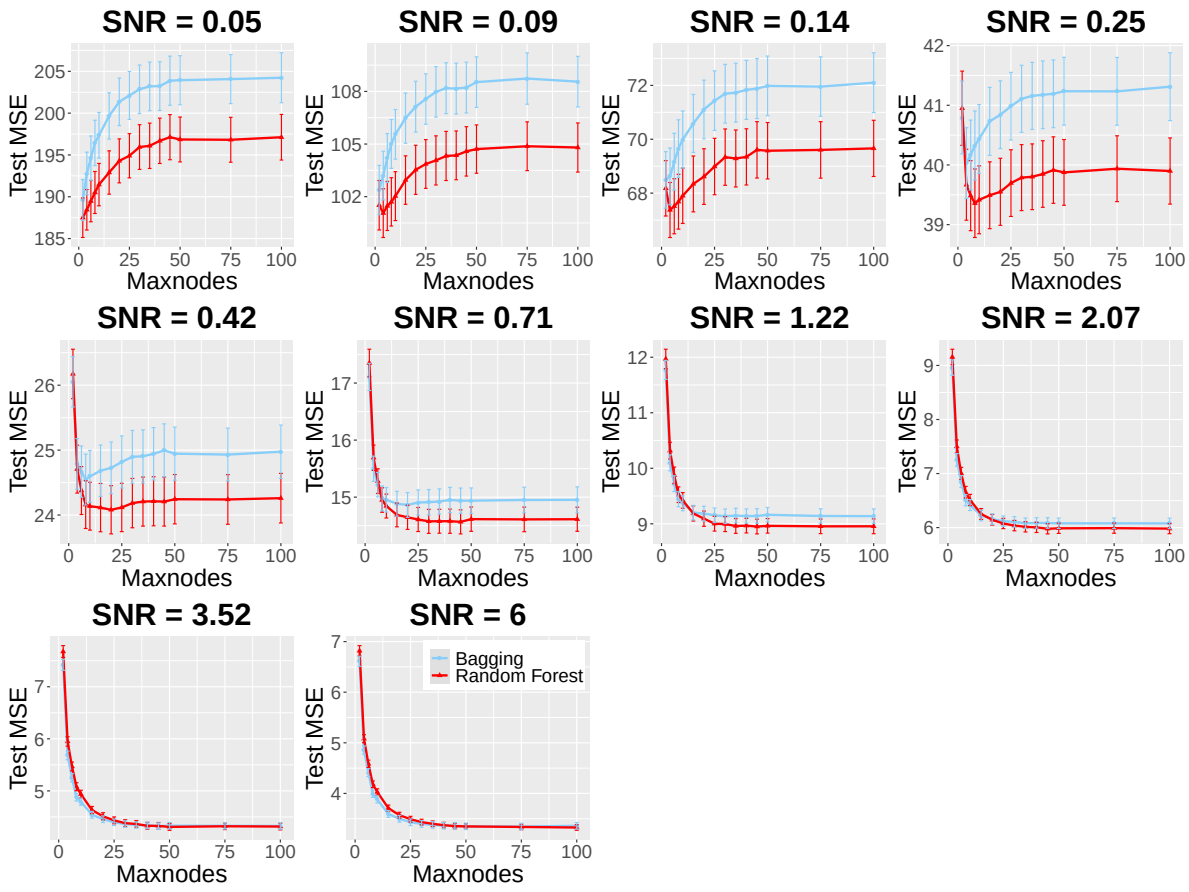

Figure C3: Performance of RFs with `mtry` equal to $p/3$ (the default value) and p (bagging) in the **low** setting. Vertical bars denote one standard error.

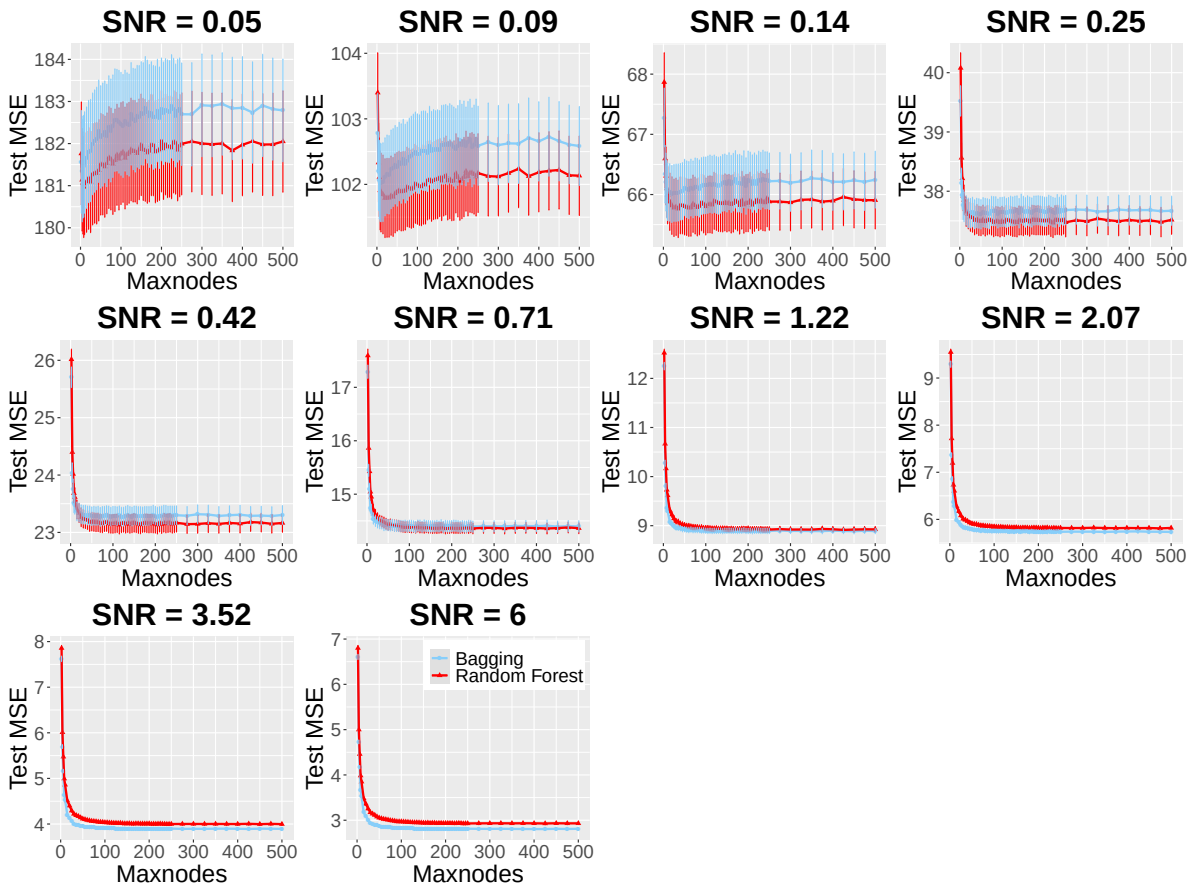

Figure C4: Performance of RFs with `mtry` equal to $p/3$ (the default value) and p (bagging) in the **medium** setting. Vertical bars denote one standard error.

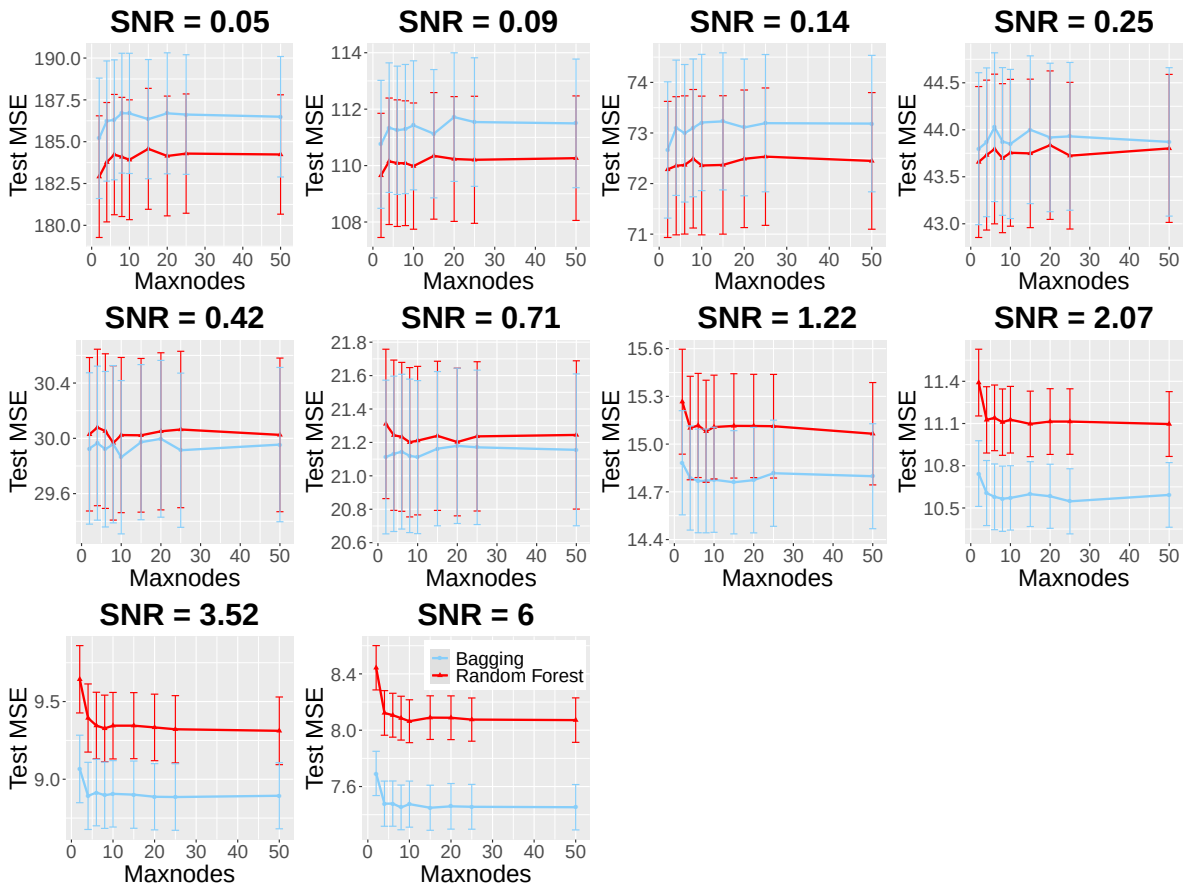

Figure C5: Performance of RFs with `mtry` equal to $p/3$ (the default value) and p (bagging) in the **high-5** setting. Vertical bars denote one standard error.

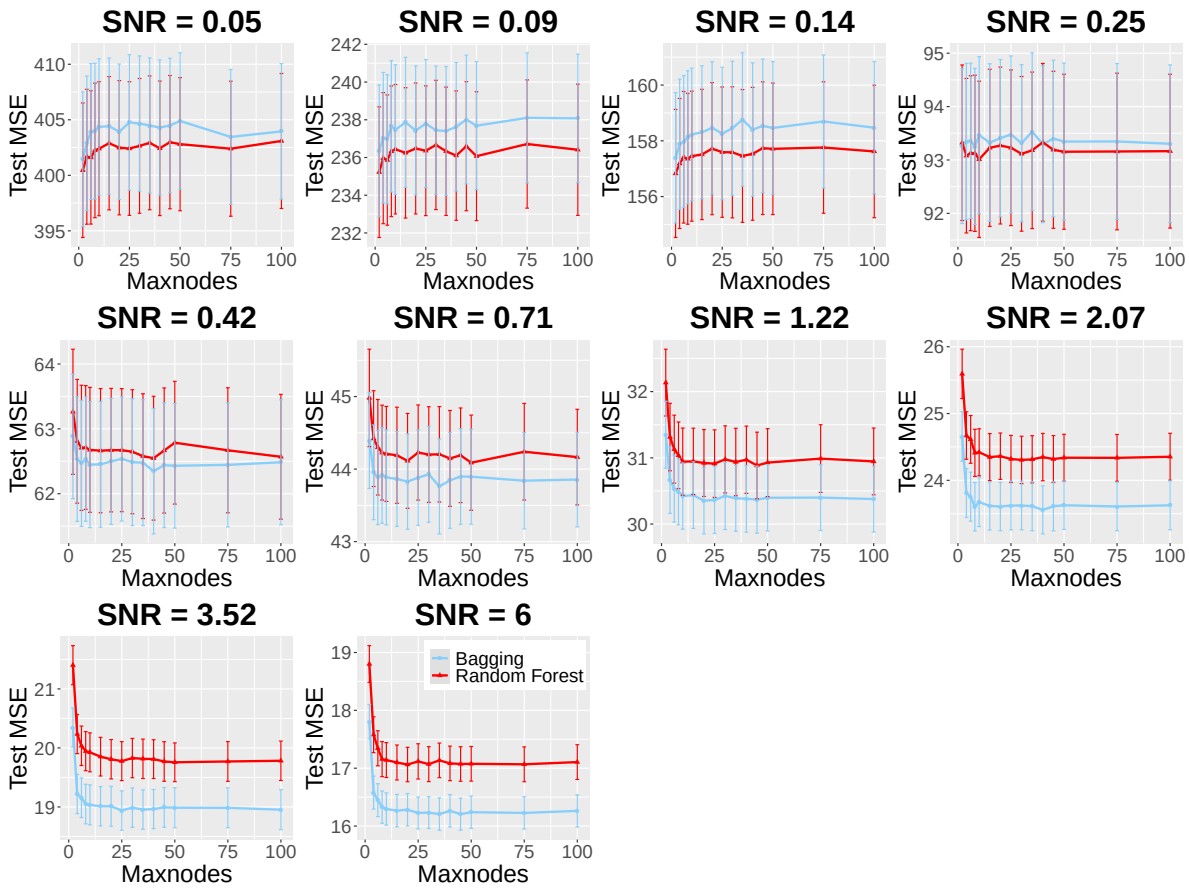

Figure C6: Performance of RFs with `mtry` equal to $p/3$ (the default value) and p (bagging) in the **high-10** setting. Vertical bars denote one standard error.

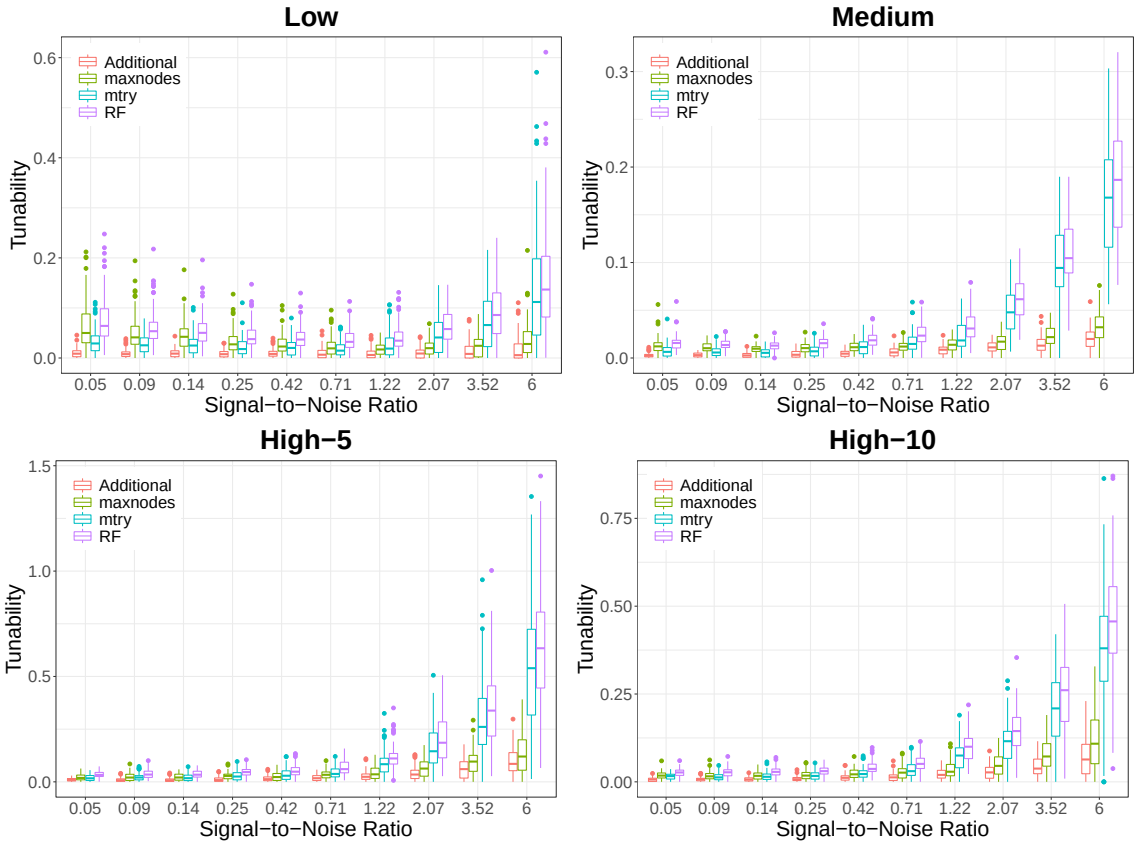

Figure C7: Boxplots of the tunability of random forests d_{RF} (purple), tunability of `maxnodes` $d_{\texttt{maxnodes}}$ (green), tunability of `mtry` $d_{\texttt{mtry}}$ (blue) and additional improvement of tuning both instead of either one $g_{\texttt{mtry,maxnodes}}$ (red) at the 10 SNR levels across the 100 repetitions. Default `mtry` $= 0.3p$ and default `maxnodes` $= n$.

BIBLIOGRAPHY

Mehreen Ahmed, Maham Jahangir, Hammad Afzal, Awais Majeed, and Imran Siddiqi. Using crowd-source based features from social media and conventional features to predict the movies popularity. In *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, pages 273–278. IEEE, 2015.

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

Sylvain Arlot and Robin Genuer. Analysis of purely random forests bias. *arXiv preprint arXiv:1407.3939*, 2014.

Rina Foygel Barber, Emmanuel J Candès, et al. Controlling the false discovery rate via knockoffs. *The Annals of Statistics*, 43(5):2055–2085, 2015.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Christel AS Bergström, Ulf Norinder, Kristina Luthman, and Per Artursson. Molecular descriptors influencing melting point and their role in classification of solid drugs. *Journal of chemical information and computer sciences*, 43(4):1177–1185, 2003.

Simon Bernard, Sébastien Adam, and Laurent Heutte. Using random forests for handwritten digit recognition. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 1043–1047. IEEE, 2007.

Simon Bernard, Laurent Heutte, and Sébastien Adam. Influence of hyperparameters on random forest accuracy. In *International Workshop on Multiple Classifier Systems*, pages 171–180. Springer, 2009.

Gérard Biau. Analysis of a Random Forests Model. *The Journal of Machine Learning Research*, 98888:1063–1095, 2012.

Gérard Biau and Luc Devroye. On the layered nearest neighbour estimate, the bagged nearest neighbour estimate and the random forest method in regression and classification. *Journal of Multivariate Analysis*, 101(10):2499–2518, 2010.

Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25(2):197–227, 2016.

Gérard Biau, Luc Devroye, and Gábor Lugosi. Consistency of Random Forests and Other Averaging Classifiers. *The Journal of Machine Learning Research*, 9:2015–2033, 2008.

Gérard Biau, Erwan Scornet, and Johannes Welbl. Neural random forests. *Sankhya A*, 81 (2):347–386, 2019.

Chris M Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

Leo Breiman. Randomizing outputs to increase prediction accuracy. *Machine Learning*, 40 (3):229–242, 2000a.

Leo Breiman. Some infinity theory for predictor ensembles. Technical report, Technical Report 579, Statistics Dept. UCB, 2000b.

Leo Breiman. Random Forests. *Machine Learning*, 45:5–32, 2001.

Leo Breiman. Consistency for a simple model of random forests. statistical department. *University of California at Berkeley. Technical Report,(670)*, 2004.

Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1st edition, 1984.

Peter Bühlmann, Bin Yu, et al. Analyzing bagging. *The Annals of Statistics*, 30(4):927–961, 2002.

Arthur Cammarata. Interrelation of the regression models used for structure-activity analyses. *Journal of medicinal chemistry*, 15(6):573–577, 1972.

Emmanuel Candes, Yingying Fan, Lucas Janson, and Jinchi Lv. Panning for gold:model-xknockoffs for high dimensional controlled variable selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(3):551–577, 2018.

Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

Tim Coleman, Wei Peng, and Lucas Mentch. Scalable and efficient hypothesis testing with random forests. *arXiv preprint arXiv:1904.07830*, 2019.

Yifan Cui, Ruoqing Zhu, Mai Zhou, and Michael Kosorok. Some asymptotic results of survival tree and forest models. *arXiv preprint arXiv:1707.09631*, 2017.

D Richard Cutler, Thomas C Edwards Jr, Karen H Beard, Adele Cutler, Kyle T Hess, Jacob Gibson, and Joshua J Lawler. Random forests for classification in ecology. *Ecology*, 88 (11):2783–2792, 2007.

Stéphane d'Ascoli, Maria Refinetti, Giulio Biroli, and Florent Krzakala. Double trouble in double descent: Bias and variance (s) in the lazy regime. *arXiv preprint arXiv:2003.01054*, 2020.

Ramón Díaz-Uriarte and Sara Alvarez De Andres. Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, 7(1):3, 2006.

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

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

Duroux, Roxane and Scornet, Erwan. Impact of subsampling and tree depth on random forests. *ESAIM: PS*, 22:96–128, 2018. doi: 10.1051/ps/2018008. URL https://doi.org/10.1051/ps/2018008.

Bradley Efron. How biased is the apparent error rate of a prediction rule? *Journal of the American statistical Association*, 81(394):461–470, 1986.

Bradley Efron and Robert Tibshirani. *Generalized Additive Models*. Chapman and Hall, London, 1990.

Phillip Ein-Dor and Jacob Feldmesser. Attributes of the performance of central processing units: A relative performance prediction model. *Communications of the ACM*, 30(4): 308–318, 1987.

Hadi Fanaee-T and Joao Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2(2-3):113–127, 2014.

Gabriele Fanelli, Matthias Dantone, Juergen Gall, Andrea Fossati, and Luc Van Gool. Random forests for real time 3d face analysis. *International Journal of Computer Vision*, 101 (3):437–458, 2013.

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

Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, 20(177):1–81, 2019.

Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.

Jerome H Friedman. Multivariate Adaptive Regression Splines. *The annals of statistics*, pages 1–67, 1991.

Robin Genuer, Jean-Michel Poggi, and Christine Tuleau. Random forests: some methodological insights. *arXiv preprint arXiv:0811.3619*, 2008.

Robin Genuer, Jean-Michel Poggi, and Christine Tuleau-Malot. Variable selection using random forests. *Pattern Recognition Letters*, 31(14):2225–2236, 2010.

Rajarshi Guha and Peter C Jurs. Development of linear, ensemble, and nonlinear models for the prediction and interpretation of the biological activity of a set of pdgfr inhibitors. *Journal of Chemical Information and Computer Sciences*, 44(6):2179–2189, 2004.

Li Guo, Nesrine Chehata, Clément Mallet, and Samia Boukir. Relevance of airborne lidar and multispectral image data for urban scene classification using random forests. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(1):56–66, 2011.

David Harrison Jr and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management*, 5(1):81–102, 1978.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, 2nd edition, 2009.

Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *arXiv preprint arXiv:1903.08560*, 2019.

Trevor Hastie, Robert Tibshirani, Ryan Tibshirani, et al. Best subset, forward stepwise or lasso? analysis and recommendations based on extensive comparisons. *Statistical Science*, 35(4):579–592, 2020.

Linnan He and Peter C Jurs. Assessing the reliability of a qsar model's predictions. *Journal of Molecular Graphics and Modelling*, 23(6):503–523, 2005.

Wassily Hoeffding. A Class of Statistics with Asymptotically Normal Distribution. *The Annals of Mathematical Statistics*, 19(3):293–325, 1948.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Giles Hooker, Lucas Mentch, and Siyu Zhou. Unrestricted permutation forces extrapolation: variable importance requires at least one more model, or there is no free variable importance. *Statistics and Computing*, 31(6):1–16, 2021.

Torsten Hothorn, Peter Bühlmann, Sandrine Dudoit, Annette Molinaro, and Mark J Van Der Laan. Survival ensembles. *Biostatistics*, 7(3):355–373, 2005.

Chen Huang, Xiaoqing Ding, and Chi Fang. Head pose estimation based on random forests for multiclass classification. In *2010 20th International Conference on Pattern Recognition*, pages 934–937. IEEE, 2010.

Hemant Ishwaran, Udaya B Kogalur, Eugene H Blackstone, Michael S Lauer, et al. Random survival forests. *The annals of applied statistics*, 2(3):841–860, 2008.

Alan Julian Izenman. *Modern multivariate statistical techniques*, volume 10. Springer, 2008.

Arthur Jacot, Berfin Simsek, Francesco Spadaro, Clément Hongler, and Franck Gabriel. Implicit regularization of random feature models. In *International Conference on Machine Learning*, pages 4631–4640. PMLR, 2020.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

Jason M. Klusowski. Sharp analysis of a simple model for random forests. *arXiv preprint 1805.02587v6*, 2019.

Dmitry Kobak, Jonathan Lomond, and Benoit Sanchez. The optimal ridge penalty for real-world high-dimensional data can be zero or negative due to the implicit ridge regularization. *Journal of Machine Learning Research*, 21(169):1–16, 2020.

Balaji Lakshminarayanan, Daniel M Roy, and Yee Whye Teh. Mondrian forests: Efficient online random forests. In *Advances in neural information processing systems*, pages 3140–3148, 2014.

Jing Lei, Max GSell, Alessandro Rinaldo, Ryan J Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523):1094–1111, 2018.

Daniel LeJeune, Hamid Javadi, and Richard Baraniuk. The implicit regularization of ordinary least squares ensembles. In *International Conference on Artificial Intelligence and Statistics*, pages 3525–3535. PMLR, 2020.

Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. Instance-based classification by emerging patterns. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 191–200. Springer, 2000.

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

Yi Lin and Yongho Jeon. Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101(474):578–590, 2006.

Miles E Lopes, Suofei Wu, and Thomas Lee. Measuring the algorithmic convergence of randomized ensembles: The regression setting. *arXiv preprint arXiv:1908.01251*, 2019a.

Miles E Lopes et al. Estimating the algorithmic variance of randomized ensembles via the bootstrap. *The Annals of Statistics*, 47(2):1088–1112, 2019b.

Mahya Mehrmohamadi, Lucas K Mentch, Andrew G Clark, and Jason W Locasale. Integrative modelling of tumour dna methylation quantifies the contribution of metabolism. *Nature communications*, 7:13666, 2016.

Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and double descent curve. *arXiv preprint arXiv:1908.05355*, 2019.

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

Nicolai Meinshausen. Relaxed lasso. *Computational Statistics & Data Analysis*, 52(1):374–393, 2007.

Lucas Mentch and Giles Hooker. Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *The Journal of Machine Learning Research*, 17(1):841–881, 2016.

Lucas Mentch and Giles Hooker. Formal hypothesis tests for additive structure in random forests. *Journal of Computational and Graphical Statistics*, 26(3):589–597, 2017.

Lucas Mentch and Siyu Zhou. Getting better from worse: Augmented bagging and a cautionary tale of variable importance. *arXiv preprint arXiv:2003.03629*, 2020a.

Lucas Mentch and Siyu Zhou. Randomization as regularization: A degrees of freedom explanation for random forest success. *Journal of Machine Learning Research*, 21(171): 1–36, 2020b.

Sérgio Moro, Paulo Rita, and Bernardo Vala. Predicting social media performance metrics and evaluation of the impact on brand building: A data mining approach. *Journal of Business Research*, 69(9):3341–3351, 2016.

Kristin K Nicodemus, James D Malley, Carolin Strobl, and Andreas Ziegler. The behaviour of random forest permutation-based variable importance measures under predictor correlation. *BMC bioinformatics*, 11(1):110, 2010.

Matthew A Olson and Abraham J Wyner. Making sense of random forest probabilities: a kernel perspective. *arXiv preprint arXiv:1812.05792*, 2018.

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

Wei Peng, Tim Coleman, and Lucas Mentch. Asymptotic distributions and rates of convergence for random forests and other resampled ensemble learners. *arXiv preprint arXiv:1905.10651*, 2019.

Andrea Peters, Torsten Hothorn, BD Ripley, T Therneau, and B Atkinson. ipred: Improved predictors, 2019. *URL https://cran.r-project.org/web/packages/ipred/ R package version 0.9-9*.

Anantha M Prasad, Louis R Iverson, and Andy Liaw. Newer classification and regression tree techniques: bagging and random forests for ecological prediction. *Ecosystems*, 9(2): 181–199, 2006.

Philipp Probst and Anne-Laure Boulesteix. To tune or not to tune the number of trees in random forest. *Journal of Machine Learning Research*, 18:181–1, 2017.

Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. Tunability: Importance of hyperparameters of machine learning algorithms. *J. Mach. Learn. Res.*, 20(53):1–32, 2019a.

Philipp Probst, Marvin N Wright, and Anne-Laure Boulesteix. Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3):e1301, 2019b.

J Ross Quinlan. Combining instance-based and model-based learning. In *Proceedings of the tenth international conference on machine learning*, pages 236–243, 1993.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017. URL https://www.R-project.org/.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

Alessandro Rudi and Lorenzo Rosasco. Generalization properties of learning with random features. *Advances in neural information processing systems*, 30, 2017.

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

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

Erwan Scornet, Gérard Biau, Jean-Philippe Vert, et al. Consistency of random forests. *The Annals of Statistics*, 43(4):1716–1741, 2015.

Mark R Segal. Machine learning benchmarks and random forest regression. 2004.

Joseph Sexton and Petter Laake. Standard errors for bagged and random forest estimators. *Computational Statistics & Data Analysis*, 53(3):801–811, 2009.

Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *CVPR 2011*, pages 1297–1304. Ieee, 2011.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC bioinformatics*, 8(1):25, 2007.

Vladimir Svetnik, Andy Liaw, Christopher Tong, J Christopher Culberson, Robert P Sheridan, and Bradley P Feuston. Random forest: a classification and regression tool for compound classification and qsar modeling. *Journal of chemical information and computer sciences*, 43(6):1947–1958, 2003.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Ryan J Tibshirani. Degrees of freedom and model search. *Statistica Sinica*, pages 1265–1296, 2015.

R Todeschini, P Gramatica, R Provenzani, and E Marengo. Weighted holistic invariant molecular descriptors. part 2. theory development and applications on modeling physicochemical properties of polyaromatic hydrocarbons. *Chemometrics and Intelligent Laboratory Systems*, 27(2):221–229, 1995.

Laura Toloşi and Thomas Lengauer. Classification with correlated features: unreliability of feature ranking and solutions. *Bioinformatics*, 27(14):1986–1994, 2011.

Athanasios Tsanas, Max A Little, Patrick E McSharry, and Lorraine O Ramig. Accurate telemonitoring of parkinson's disease progression by noninvasive speech tests. *IEEE transactions on Biomedical Engineering*, 57(4):884–893, 2009.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013. doi: 10.1145/2641190.2641198. URL http://doi.acm.org/10.1145/2641190.2641198.

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

Stefan Wager, Sida Wang, and Percy S Liang. Dropout training as adaptive regularization. In *Advances in neural information processing systems*, pages 351–359, 2013.

Stefan Wager, Trevor Hastie, and Bradley Efron. Confidence intervals for random forests: The jackknife and the infinitesimal jackknife. *The Journal of Machine Learning Research*, 15(1):1625–1651, 2014.

Samuel George Waugh. *Extending and benchmarking Cascade-Correlation: extensions to the Cascade-Correlation architecture and benchmarking of feed-forward supervised artificial neural networks*. PhD thesis, University of Tasmania, 1995.

Johannes Welbl. Casting random forests as artificial neural networks (and profiting from it). In *German Conference on Pattern Recognition*, pages 765–771. Springer, 2014.

Brian D Williamson, Peter B Gilbert, Noah R Simon, and Marco Carone. A general framework for inference on algorithm-agnostic variable importance. *Journal of the American Statistical Association*, (just-accepted):1–38, 2021.

Abraham J Wyner, Matthew Olson, Justin Bleich, and David Mease. Explaining the success of adaboost and random forests as interpolating classifiers. *The Journal of Machine Learning Research*, 18(1):1558–1590, 2017.

Donghui Yan, Aiyou Chen, and Michael I Jordan. Cluster forests. *Computational Statistics & Data Analysis*, 66:178–192, 2013.

I-C Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 28(12):1797–1808, 1998.

Zhao Yi, Stefano Soatto, Maneesh Dewan, and Yiqiang Zhan. Information forests. In *2012 information theory and applications workshop*, pages 143–146. IEEE, 2012.

Faisal Zaman and Hideo Hirose. Effect of subsampling rate on subbagging and related ensembles of stable classifiers. In *International Conference on Pattern Recognition and Machine Intelligence*, pages 44–49. Springer, 2009.

Siyu Zhou and Lucas Mentch. Trees, forests, chickens, and eggs: when and why to prune trees in a random forest. *arXiv preprint arXiv:2103.16700*, 2021.

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