



1-1-2015

Extensions and Applications of Ensemble-of-trees Methods in Machine Learning

Justin Bleich

University of Pennsylvania, jbleich89@gmail.com

Follow this and additional works at: <http://repository.upenn.edu/edissertations>



Part of the [Statistics and Probability Commons](#)

Recommended Citation

Bleich, Justin, "Extensions and Applications of Ensemble-of-trees Methods in Machine Learning" (2015). *Publicly Accessible Penn Dissertations*. 1016.

<http://repository.upenn.edu/edissertations/1016>

This paper is posted at ScholarlyCommons. <http://repository.upenn.edu/edissertations/1016>

For more information, please contact libraryrepository@pobox.upenn.edu.

Extensions and Applications of Ensemble-of-trees Methods in Machine Learning

Abstract

Ensemble-of-trees algorithms have emerged to the forefront of machine learning due to their ability to generate high forecasting accuracy for a wide array of regression and classification problems. Classic ensemble methodologies such as random forests (RF) and stochastic gradient boosting (SGB) rely on algorithmic procedures to generate fits to data. In contrast, more recent ensemble techniques such as Bayesian Additive Regression Trees (BART) and Dynamic Trees (DT) focus on an underlying Bayesian probability model to generate the fits.

These new probability model-based approaches show much promise versus their algorithmic counterparts, but also offer substantial room for improvement. The first part of this thesis focuses on methodological advances for ensemble-of-trees techniques with an emphasis on the more recent Bayesian approaches. In particular, we focus on extensions of BART in four distinct ways. First, we develop a more robust implementation of BART for both research and application. We then develop a principled approach to variable selection for BART as well as the ability to naturally incorporate prior information on important covariates into the algorithm. Next, we propose a method for handling missing data that relies on the recursive structure of decision trees and does not require imputation. Last, we relax the assumption of homoskedasticity in the BART model to allow for parametric modeling of heteroskedasticity.

The second part of this thesis returns to the classic algorithmic approaches in the context of classification problems with asymmetric costs of forecasting errors. First we consider the performance of RF and SGB more broadly and demonstrate its superiority to logistic regression for applications in criminology with asymmetric costs. Next, we use RF to forecast unplanned hospital readmissions upon patient discharge with asymmetric costs taken into account. Finally, we explore the construction of stable decision trees for forecasts of violence during probation hearings in court systems.

Degree Type

Dissertation

Degree Name

Doctor of Philosophy (PhD)

Graduate Group

Statistics

First Advisor

Richard Berk

Keywords

bayesian, ensemble, machine learning, predictive, random forests, statistical learning

Subject Categories

Statistics and Probability

EXTENSIONS AND APPLICATIONS OF ENSEMBLE-OF-TREES METHODS IN MACHINE LEARNING

Justin Bleich

A DISSERTATION

in

Statistics

For the Graduate Group in
Managerial Science and Applied Economics

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy

2015

Supervisor of Dissertation

Signature _____

Richard A. Berk
Professor, Criminology and
Statistics

Graduate Group Chairperson

Signature _____

Eric T. Bradlow, K. P. Chao Professor,
Marketing, Statistics, and Education

Dissertation Committee

Richard A. Berk, Professor,
Criminology and Statistics
Ed George, Universal Furniture
Professor, Statistics

Abba M. Krieger, Robert Steinberg
Professor, Statistics

EXTENSIONS AND APPLICATIONS OF ENSEMBLE-OF-TREES METHODS
IN MACHINE LEARNING

COPYRIGHT ©
2015

Justin Bleich

Acknowledgments

I would like to first and foremost thank my family for their never-ending love and support. I can assuredly say I would not have been able to reach this milestone in my life without you beside me every step of the way.

I would like to thank my advisor, Professor Richard Berk, for his guidance and direction throughout both my graduate and undergraduate studies. Additionally, I would like to thank Professors Ed George and Abba Krieger for supervising my thesis. I would also like to thank Professor Adi Wyner for his support over the past few years.

I would like to thank Adam Kapelner, my primary research associate, for all the hard work and long hours that helped make much of the research herein a reality. I would like to thank Emil Pitkin and Alex Goldstein who have been both great friends and research partners during my graduate studies. Finally, I would like to thank all of my friends who have been there for me as well.

ABSTRACT

EXTENSIONS AND APPLICATIONS OF ENSEMBLE-OF-TREES METHODS IN MACHINE LEARNING

Justin Bleich

Richard Berk

Ensemble-of-trees algorithms have emerged to the forefront of machine learning due to their ability to generate high forecasting accuracy for a wide array of regression and classification problems. Classic ensemble methodologies such as random forests (RF) and stochastic gradient boosting (SGB) rely on algorithmic procedures to generate fits to data. In contrast, more recent ensemble techniques such as Bayesian Additive Regression Trees (BART) and Dynamic Trees (DT) focus on an underlying Bayesian probability model to generate the fits.

These new probability model-based approaches show much promise versus their algorithmic counterparts, but also offer substantial room for improvement. The first part of this thesis focuses on methodological advances for ensemble-of-trees techniques with an emphasis on the more recent Bayesian approaches. In particular, we focus on extensions of BART in four distinct ways. First, we develop a more robust implementation of BART for both research and application. We then develop a principled

approach to variable selection for **BART** as well as the ability to naturally incorporate prior information on important covariates into the algorithm. Next, we propose a method for handling missing data that relies on the recursive structure of decision trees and does not require imputation. Last, we relax the assumption of homoskedasticity in the **BART** model to allow for parametric modeling of heteroskedasticity.

The second part of this thesis returns to the classic algorithmic approaches in the context of classification problems with asymmetric costs of forecasting errors. First we consider the performance of **RF** and **SGB** more broadly and demonstrate its superiority to logistic regression for applications in criminology with asymmetric costs. Next, we use **RF** to forecast unplanned hospital readmissions upon patient discharge with asymmetric costs taken into account. Finally, we explore the construction of stable decision trees for forecasts of violence during probation hearings in court systems.

Table of Contents

List of Tables	x
List of Figures	xi
1 Introduction	1
1.1 Overview	1
1.2 Decision Trees	4
1.2.1 Classification Trees	7
1.2.2 Regression Trees	8
1.3 Algorithmic Ensemble-of-trees Techniques	9
1.3.1 Random Forests	9
1.3.2 Stochastic Gradient Boosting	11
1.4 Bayesian Additive Regression Trees	13
1.4.1 Priors and likelihood	14
1.4.2 Posterior distribution and prediction	17
1.4.3 BART for classification	19
1.4.4 BART Implementation	20
2 Bayesian Additive Regression Trees Implementation	21
2.1 Introduction	21
2.1.1 Comparison of BART Implementations	22
2.2 The bartMachine package	23
2.2.1 Speed improvements and parallelization	24
2.2.2 Implementation of Tree Alterations	28
2.2.3 Implementation of Research Features	28
2.3 bartMachine Package Features for Regression	29
2.3.1 Computing parameters	29
2.3.2 Model building	30

2.3.3	Assumption checking	35
2.3.4	Credible intervals and prediction intervals	38
2.3.5	Variable importance	39
2.3.6	Variable effects	39
2.3.7	Partial dependence	44
2.3.8	Incorporating missing data	45
2.3.9	Variable selection	48
2.3.10	Informed prior information on covariates	51
2.3.11	Interaction effect detection	52
2.3.12	bartMachine Model Persistence Across R Sessions	53
2.4	bartMachine Package Features for Classification	55
2.5	Conclusion	60
2.5.1	Forecasting “Bakeoff”	60
2.5.2	Discussion	61
3	Variable Selection for BART	62
3.1	Introduction	63
3.2	Techniques for Variable Selection	65
3.2.1	Linear Methods	65
3.2.2	Tree-Based Methods	66
3.3	Calibrating BART Output for Variable Selection	67
3.3.1	BART Variable Inclusion Proportions	69
3.3.2	Further Exploration of Null Simulation	70
3.3.3	Variable Inclusion Proportions under Permuted Responses	74
3.3.4	Real Prior Information in BART-based Variable Selection	79
3.4	Simulation Evaluation of BART-based Variable Selection	80
3.4.1	Simulation Setting 1: Linear Relationship	84
3.4.2	Simulation Setting 2: Nonlinear Relationship	87
3.4.3	Simulation Setting 3: Linear Model with Informed Priors	90
3.5	Application to Gene Regulation in Yeast	93
3.6	Conclusion	101
4	Incorporating Missingness into BART	104
4.1	Introduction	104
4.2	Background	106
4.2.1	A Framework for Missing Data in Statistical Learning	106
4.2.2	Strategies for Incorporating Missing Data in Statistical Learning	110
4.2.3	Missing data in Binary Decision Trees	112
4.3	Missing Incorporated in Attributes within BART	113
4.4	Generated Data Simulations	116
4.4.1	A Simple pattern-mixture Model	116
4.4.2	Selection Model Performance	120
4.5	Real Data Example	122

4.6	Discussion	126
5	BART with Parametric Models of Heteroskedasticity	130
5.1	Introduction	130
5.2	Bayesian Heteroskedastic Regression	132
5.3	Augmenting BART to Incorporate Heteroskedasticity	133
5.3.1	Priors and Likelihood	135
5.3.2	Sampling from the Posterior	136
5.4	Simulations	139
5.4.1	Univariate Model	139
5.4.2	Multivariate Model	142
5.5	Real Data Examples	145
5.5.1	Lidar Data	145
5.5.2	Motorcycle Data	145
5.6	Discussion	148
6	Ensemble-of-Trees Algorithms in Criminology	150
6.1	Introduction	151
6.2	Proper Criminal Justice Forecasting Comparisons	154
6.2.1	Some Common-Sense Requirements for Fair Forecasting Comparisons	156
6.3	Some Conceptual Fundamentals	159
6.3.1	The Basic Account	159
6.3.2	Building in Differential Forecasting Error Costs	163
6.3.3	Nonlinear Decision Boundaries	165
6.3.4	Enter Machine Learning	169
6.4	The Forecasting Contestants	170
6.4.1	Logistic Regression with Asymmetric Costs	171
6.4.2	Random Forests with Asymmetric Costs	172
6.4.3	Asymmetric Costs in Stochastic Gradient Boosting	173
6.4.4	A Simulation	174
6.5	An Empirical Example	179
6.5.1	Forecasting Arrests for Serious Crimes	179
6.6	Conclusions	182
7	Using Random Forests with Asymmetric Costs to Predict Hospital Readmissions	184
7.1	Background and Significance	185
7.2	Objective	188
7.3	Materials and Methods	188
7.3.1	Setting	188
7.3.2	Outcome Variables	188
7.3.3	Predictor Variables	189

7.3.4	Random Forests with Asymmetric Costs	189
7.3.5	Variable Selection	190
7.3.6	Model Evaluation	191
7.4	Results	192
7.4.1	Descriptive Characteristics	192
7.4.2	30-Day All-cause Readmissions Random Forests Model	193
7.4.3	7-Day Unplanned Readmissions Random Forests Model	196
7.5	Discussion	197
7.6	Conclusion	200
8	Bootstrapping for Stable Classification Trees	202
8.1	Introduction	203
8.2	Modifying Classification Trees for Criminal Justice Settings	205
8.2.1	Asymmetric Costs and Tuning	207
8.2.2	Stability Analysis	209
8.3	Data	210
8.3.1	Variables	211
8.4	Random Forest Results	213
8.5	Classification Tree Results	214
8.5.1	A Classification Tree Visualization	216
8.5.2	Stability Analysis	218
8.6	Conclusions	225
9	Conclusion	227
A	Appendices	229
A.1	Supplement for Chapter 1	229
A.1.1	Sampling New Trees	229
A.1.2	Grow Proposal	230
A.1.3	Prune Proposal	235
A.1.4	Change	236
A.2	Supplement for Chapter 3	240
A.2.1	Pseudo-code for Variable Selection Procedures	240
A.3	Supplement for Chapter 5	244
A.4	Supplement for Chapter 8	253
	Bibliography	254

List of Tables

2.1	BART implementation feature comparison	24
2.2	Bakeoff of BayesTree, BART, and RF	60
2.3	Runtimes for BayesTree, BART, and RF	61
3.1	Nested variance decomposition of inclusion proportions variability . .	72
3.2	Distribution of prior influences values across 6026 genes	99
4.1	Missing data mechanisms in statistical learning framework	110
4.2	Missing data scenarios for Boston Housing data simulations	124
6.1	Logistic regression confusion table using simulated test data.	177
6.2	Logistic regression with interaction confusion table using simulated test data.	178
6.3	Random forests confusion table using simulated test data.	179
6.4	Logistic regression test data confusion table for serious crime.	180
6.5	Random forests test data confusion table for serious crime.	181
6.6	Stochastic gradient boosting test data confusion table for serious crime.	181
7.1	University of Pennsylvania Health System study cohort characteristics	192
7.2	Incidence rates for 30-day all cause and 7-day readmissions	193
8.1	RF confusion table for violent crime prediction	213
8.2	Classification tree confusion table for violent crime prediction	215

List of Figures

1.1	Growing a classification tree	6
2.1	Runtime comparisons for BART implementations	27
2.2	Out-of-sample predictive performance by number of trees.	33
2.3	BART model assumption checking plots	36
2.4	BART convergence diagnostic plots	37
2.5	Fitted versus actual values for automobile dataset	40
2.6	Average variable inclusion proportions for automobile data	41
2.7	Tests of covariate importance	43
2.8	Partial dependence plots for automobile data	46
2.9	Visualization of variable selection procedure	50
2.10	Top interaction counts for Friedman function data	54
2.11	Covariate importance test for Pima Indians data	59
2.12	Partial dependence plot for predictor <code>glu</code> from Pima Indians data . .	59
3.1	Variable inclusion proportions in null setting	71
3.2	Visualization of nested variance decomposition	74
3.3	Top variable inclusion proportions for yeast geen YAL004W with selection	78
3.4	Average F_1 scores by method for linear simulation setting	86
3.5	Average F_1 scores by method for nonlinear simulation setting	89
3.6	Average F_1 scores for different BART prior choices	92
3.7	Distributions of number of predictors selected by method across 6026 genes	97
3.8	Distributions of RMSE reduction per predictor by method across 6026 genes	99
3.9	Number of genes for which each transcription factor was selected . . .	101

4.1	BART estimation of conditional means under different missing data mechanisms	118
4.2	BARTm versus complete-case analysis results	123
4.3	Bakeoff results for Boston Housing Data	127
5.1	BART's and HBART's posterior mean estimates	140
5.2	Estimates of the conditional mean function f for HBART and BART . .	143
5.3	Distribution of out-of-sample RMSE for BART and HBART	144
5.4	Posterior mean estimates for BART and HBART (Lidar data)	146
5.5	Posterior predictive intervals for many methods (motorcycle data) . .	147
6.1	Two linear decision boundaries in 2-dimensional predictor space . . .	160
6.2	Impact of asymmetric costs in 2-dimensional predictor space	164
6.3	Linear and nonlinear 2D decision boundary	165
6.4	Challenging classification surface for parametric models	175
7.1	RF confusion matrix for 30-day readmissions on training data	194
7.2	RF confusion matrix for 30-day readmissions on validation data	195
7.3	Variable importance plot for 30-day all-cause readmissions RF model .	195
7.4	RF confusion matrix for 7-day readmissions on training data	197
7.5	RF confusion matrix for 7-day readmissions on validation data	198
7.6	Variable importance plot for 7-day unplanned readmissions RF model	198
8.1	Classification tree for arrest for violent crime	217
8.2	Histogram of proportions of common classifications between tree pairs.	222
8.3	Histogram of proportions of common classifications between tree pairs with stability condition	223

1.1 Overview

Leo Breiman eloquently captures the motivation for and aim of statistical machine learning in his 2001 *Statistical Science* paper:

“The approach is that nature produces data in a black box whose insides are complex, mysterious, and, at least, partly unknowable. What is observed is a set of \mathbf{x} ’s that go in and a subsequent set of \mathbf{y} ’s that come out. The problem is to find an algorithm $f(\mathbf{x})$ such that for future \mathbf{x} in a test set, $f(\mathbf{x})$ will be a good predictor of \mathbf{y} .”

For the past few decades, statisticians and computer scientists have sought to uncover “nature’s black box” across a wide array of domains and applications, leading to many novel algorithms and advancements in a field that today has come to be known as “supervised learning.”

In particular, ensemble methods have gained much popularity in supervised learning problems, where the goal is to estimate this unknown function f from observed data. Ensemble methods take a set of base algorithms, also called “learners”, and

combine them together in some fashion to produce a final model. In both theory and practice, ensemble methods are often able to improve predictive performance relative to the capabilities of the base algorithms that constitute the ensemble. Such findings hold true for both regression and classification settings (Rokach, 2009).

We focus in particular on ensemble-of-trees procedures. These algorithms are machine learning techniques that use decision trees, such as Classification and Regression Trees (CART, Breiman et al., 1984), as the base learner. As a standalone technique, decision trees provide highly interpretable models that can effectively capture interaction effects and nonlinearities (Chipman et al., 2010). Regarding predictive accuracy, decision trees fare reasonably well; Breiman (2001b) states that decision trees rate “a B on prediction.” More recent efforts in machine learning though have studied the predictive performance of ensembles of decision trees. Such techniques include random forests (RF, Breiman, 2001a), stochastic gradient boosting (SGB, Friedman, 2002), Bayesian Additive Regression Trees (BART, Chipman et al., 2010) and Dynamic Trees (DT, Taddy et al., 2011). The aforementioned algorithms have achieved superior predictive accuracy across a wide array of domains and rank among the most competitive techniques for supervised learning to date. In order to achieve strong predictive performance, these models sacrifice a high degree of interpretability associated with the component decision trees. For example, Breiman (2001b) remarks that RF “are A+ predictors...on interpretability, they rate an F.” Much work has been done to improve the interpretability of such “black-box” algorithms, and some new developments will be presented in this work.

The ensemble-of-trees methods can be divided into two distinct groups. The first group contains the supervised learning procedures that are constructed in a purely algorithmic fashion; there is no statistical model underlying the technique. RF and SGB fall into this group. The second group, developed more recently, contains supervised

learning procedures that posit an underlying probability model. **BART** and **DT**, two members of the second group, are both based on Bayesian probability models. In the first component of this thesis, we develop methodological extensions for the models in the second group, particularly for **BART**. In the second half of this thesis, we return to the algorithmically-motivated procedures and explore applications in criminology and healthcare where there are asymmetric costs associated with forecasting errors.

The remainder of this thesis is organized as follows. In the subsequent sections of this chapter, we first review **CART** as an example of how a decision tree is constructed. We next briefly review **RF** and **SGB**, and then provide a more extensive introduction to **BART**. Each chapter following the introduction is adapted from a specific article (cited at the end of each chapter) and modified for coherence within the overall thesis.

The next four chapters develop extensions for **BART**. Chapter 2 introduces **bartMachine**, a new R package implementing **BART** for both robust research and application, highlighting key performance and visualization features of the package. Using the **bartMachine** implementation as a computational engine, we then develop a number of methodological innovations for **BART**. Chapter 3 develops a method for principled nonparametric variable selection using **BART**. The approach relies on applying frequentist permutation testing ideas to output from a Bayesian model. We additionally develop a means for incorporating informed prior information into the variable selection and demonstrate the promise of our approach via an application to inferring the genetic regulatory network in yeast. The third methodological innovation, discussed in Chapter 4, offers an approach for incorporating missing data into **BART** in both the training and forecasting phases. The approach takes advantage of the structure of decision trees and does not require any imputation. Finally, Chapter 5 relaxes the assumption of homoskedasticity in the original **BART** model and introduces an approach for specifying parametric models of heteroskedasticity in

BART.

Following the methodological innovations, Chapters 6-8 discuss applications of ensemble-of-trees methods in classification problems where forecasting errors associated with each outcome class have asymmetric costs. Chapter 6 advocates for the superiority of RF and SGB over traditional techniques such as logistic regression in criminology applications with asymmetric forecasting costs. The exposition in the chapter is largely didactic and meant to highlight the general merits of machine learning methods over parametric modeling for prediction problems in criminology. Continuing with the theme of asymmetric costs incorporated into ensemble-of-trees models, we next switch the subject matter domain. Chapter 7 proposes RF models for forecasts of unplanned hospital readmissions that rely on a large set of patient covariates. The models herein are designed to be used in real-time upon patient discharge and account for the asymmetric forecasting costs that health system administrators face. Finally, we return to the criminology setting to explore a scenario where traditional ensemble methods may not be available. Chapter 8 develops prototype classification tree models that can be used in sentencing decisions for courts where the technology to access more sophisticated procedures may not be available. We develop an approach to generate trees with stable predictions by examining classification agreement across an ensemble of bootstrapped trees. Finally, Chapter 9 offers directions for future research and concludes.

1.2 Decision Trees

The basic building block for all ensemble-of-trees algorithms is the decision tree. Although multiple versions of decision trees have been proposed in the literature, we limit our focus to CART and cover the key aspects of decision trees required to

understand the ensemble methods developed herein. The ensuing discussion draws from Breiman et al. (1984) and Hastie et al. (2009).

At a high level, decision trees partition the observed data (or training data) into subsets with similar values of the response. This is accomplished by exploiting the predictors to create the greatest homogeneity between the observed outcomes and the outcomes that the tree can project. Let \mathbf{y} denote the response vector, which may be continuous or categorical, and let $\mathbf{x}_1, \dots, \mathbf{x}_p$ denote the set of p predictors as column vectors. With this notation, we can describe the general construction of a **CART** tree. After introducing the algorithm, we will highlight key aspects of regression trees and classification trees separately.

First, all of the observations begin in a single node known as the “root node.” The data is the partitioned into two child nodes by considering “splitting rules” of the form $\mathbf{x}_j < c$. Here, \mathbf{x}_j denotes the “splitting variable” (or “splitting attribute”) and c denotes the “splitting value.” All observations satisfying the “splitting rule” are sent to the left child node and the rest of the observations are sent to the right child node. Using a greedy approach, all possible splitting rules that can be created from the observed data are evaluated and the rule that minimizes some statistical criterion is chosen. Once a split is selected, the observations are moved to their respective nodes and the above process is repeated for each of the two child nodes. Such split construction proceeds in a recursive fashion until a stopping rule is satisfied. Examples of common stopping rules include a maximum depth for which a tree is allowed to grow, a constraint on the minimum number of observations in a node, or a required reduction in the value of the statistical criterion selected. Nodes which no longer split are labelled “terminal nodes.” The construction of a **CART** tree is completed once all nodes are labelled as terminal. Note that in some cases, the nodes are removed from the **CART** tree in a process known as “pruning,” but we omit a discussion of

CART pruning as it is generally not applicable to the understanding of the ensemble methods we discuss.

After tree construction, fitted values are assigned to each terminal node based on the observations that land in that node. As will be shown, the fitted values will be an average of response values for regression trees and the majority class (or plurality class for multi-class problems) in the node for classification trees. Forecasts for observations with an unknown response can be obtained by “dropping” the cases down the tree and following the path implied by the splitting rules. The forecast assigned to the observation is the fitted value associated with the terminal node in which the observation lands.

Figure 1.1 shows two steps in the growth of a classification tree for response y with levels “0” and “1” and predictors x_1 and x_2 . Terminal nodes are assigned the majority class as evidenced by the proportion of outcome classes given in the terminal nodes.

With the general CART algorithm in place, we now highlight the specifics of classification trees and regression trees.

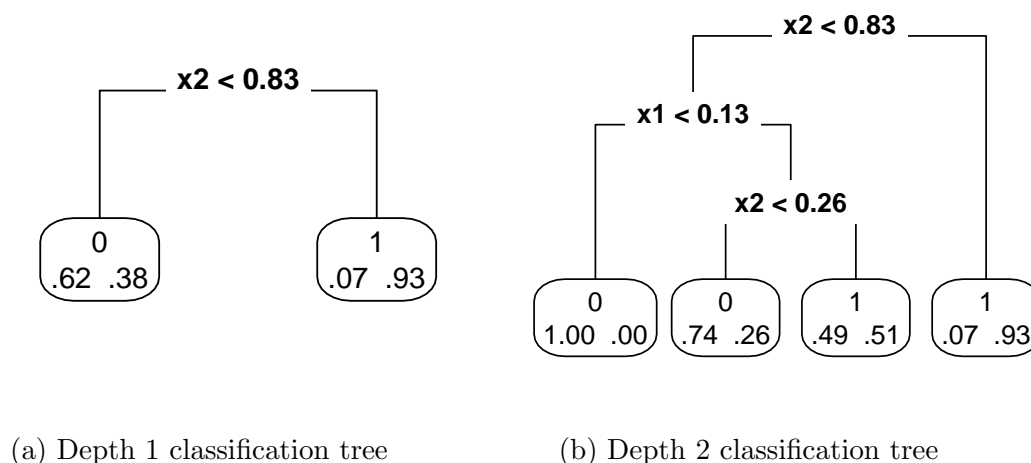


Figure 1.1: Two steps in the growing process of a classification tree.

1.2.1 Classification Trees

Classification trees are employed for data with categorical response variables. Suppose there are K distinct classes for the response in the training data. For the m^{th} node in the tree, let P_m represent the set of observations in node m of size N_m . One can compute

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{i \in P_m} I(y_i = k) \quad (1.1)$$

which is the empirical proportion of observations belonging to class k in node m . \hat{p}_{mk} is then used to define the statistical criterion used to determine the optimal splitting rule at each step in growing classification trees. This criterion acts to minimize within-node homogeneity of class labels and are often called “impurity functions.” Breiman et al. (1984) proposes three commonly impurity functions:

$$\text{Misclassification error} : \frac{1}{N_m} \sum_{i \in P_m} I(y_i \neq \hat{c}_m) \quad (1.2)$$

$$\text{Gini index} : \sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} \quad (1.3)$$

$$\text{Cross-entropy} : - \sum_{k=1}^K \hat{p}_{mk} \log(\hat{p}_{mk}) \quad (1.4)$$

In practice, the Gini index or cross-entropy are more commonly employed as they are smooth functions and hence more amenable to optimization. These criteria are also more sensitive to changes in the node probabilities compared to the misclassification error.

Letting ϕ denote an impurity function evaluated on the observations in node m , for each splitting rule s evaluated, the goodness of split s is given by

$$\Delta(s, m) = \phi(m) - \frac{1}{N_{m_L}}\phi(m_L) - \frac{1}{N_{m_R}}\phi(m_R) \quad (1.5)$$

where m_L and m_R denote the left and right child nodes resulting from splitting rule s on node m . The splitting rule that maximizes $\Delta(s, m)$ across all available rules is selected and the tree is partitioned.

Once the classification tree has been fully grown, fitted values for terminal nodes are often assigned by taking the plurality class in the node. Letting \hat{c}_m denote the fitted value for a (terminal) node m , we have

$$\hat{c}_m = \operatorname{argmax}_k \hat{p}_{mk}. \quad (1.6)$$

Note that the assignment above operates under the assumption of symmetric costs for misclassification errors. We will introduce asymmetric misclassification costs and the appropriate modifications to the CART algorithm in Chapter 8.

1.2.2 Regression Trees

Regression trees are employed for data with continuous response variables. For regression trees, the impurity function used for CART is the sum of square errors $\sum_{i \in P_m} (y_i - \bar{y}_m)^2$ where \bar{y}_m is the average of the observations in node m . The splitting rule s that maximizes $\Delta(s, m)$ for node m is selected at each partitioning step.

Terminal node fitted values in regression trees are given by the average value of all observations in the terminal node:

$$\hat{c}_m = \frac{1}{N_m} \sum_{i \in P_m} y_i. \quad (1.7)$$

1.3 Algorithmic Ensemble-of-trees Techniques

Two popular ensemble-of-trees techniques, **RF** and **SGB** traditionally use **CART** trees as the basic building blocks of their ensembles. The trees serve as the base learners within the algorithm and the output of each tree is combined to create a final fitted value or forecast. **RF** and **SGB** use the **CART** trees in substantially different fashions and we review each algorithm in the subsequent sections. The discussion for **RF** draws from Breiman (2001a) and Hastie et al. (2009, chapter 15), and the discussion for **SGB** draws upon Friedman (2002) and Hastie et al. (2009, chapter 10).

1.3.1 Random Forests

RF builds upon a method known as “bootstrap aggregation,” or “bagging” (Breiman, 1996), in which multiple versions of some base learner are independently constructed on different bootstrapped samples of the data. In bagging, the forecasts from each of the learners is then aggregated, often via averaging for continuous responses or plurality vote for categorical responses. When grown sufficiently deep, trees are especially good candidates for bagging as they are considered to be low-bias, high variance learners. Aggregating across multiple instances can work to lower variance and reduce generalization error.

RF takes bagging one step further in an effort to further reduce variance. The key innovation is to search for an optimal split at each partitioning step by greedily

evaluating a *subset* of predictors rather than all predictors. By choosing $m^* \leq p$ predictors, an additional source of randomness is introduced into the tree-growing process. This randomness decorrelates the decision trees across the ensemble (at the expense of a slight increase in the individual variance of each tree), which can further reduce the variance of the aggregated ensemble.

RF contains two important tuning parameters. The first is the number of decision trees to grow in the ensemble. Typically, no more than 500 decision trees are needed for the performance of RF to stabilize and the procedure does not overfit the data as the number of trees grows larger. The second tuning parameter is m^* , the number of predictors to evaluate at each split. For regression problems, $p/3$ is recommended and for classification problems, \sqrt{p} is the commonly used default. Empirical evidence suggests that RF is fairly robust to choices of these tuning parameters. For certain applications, additionally tuning the minimum number of observations required to split a node may also be useful. Algorithm 1 provides the pseudocode for the default RF algorithm.

Algorithm 1 Random Forests

1. For $b = 1$ to B :
 - (a) Draw a bootstrap sample \mathbf{X}^* of size N from the training data
 - (b) Grow a decision tree T_b to the data \mathbf{X}^* by doing the following recursively until the minimum node size n_{min} is reached:
 - i. Select m^* of the p variables
 - ii. Pick the best split from the m^* variables and partition
2. Output the ensemble $\{T_b\}_b^B$

Classification: Let $\hat{C}_b(\mathbf{x}^*)$ be the predicted class for \mathbf{x}^* tree T_b . Then $\hat{C}_{rf}^B(\mathbf{x}^*) = \text{plurality vote}\{\hat{C}_b(\mathbf{x}^*)\}_1^B$.

Regression: Let $\hat{f}_b(\mathbf{x}^*)$ be the predicted value from tree T_b . Then $\hat{f}_{rf}^B(\mathbf{x}^*) = \frac{1}{B} \sum_{i=1}^B \hat{f}_i(\mathbf{x}^*)$.

When predicting for new observations \mathbf{x}^* , RF computes forecasts by dropping the

observation down each decision tree as described in the above algorithm. However, there is often interest in obtaining fitted values for the training data. Such estimates can be obtained via “out-of-bag” (OOB) estimates. OOB estimates are computed by obtaining the forecasts for observations in the training data on decision trees for which the observations were not used to construct the tree. This implies that such observations were not selected in the bootstrap sample \mathbf{X}^* and hence are referred to as out-of-bag for such trees. For each observation, the set of forecasts from each tree is then aggregated to produce a single fitted value. Given that OOB estimates are obtained from trees where the observation did not contribute to tree construction, the estimates are often a good proxy for true out-of-sample performance. We will make extensive use of this fact in Chapters 6-8 to tune the RF models for asymmetric costs of forecasting errors.

Throughout this work, we use the RF implementation provided in the R package `randomForest` (Liaw and Wiener, 2002).

1.3.2 Stochastic Gradient Boosting

In contrast to RF which aggregates a collection of independent base learners, SGB operates on a set of learners in a stepwise fashion. SGB proceeds by taking a pass through the data, constructing a base learner, and then reweighting observations that were more difficult to correctly classify. While there are numerous flavors of boosting that each result in different reweighting schemes for the data, SGB operates on the negative partial derivatives of the loss function at each training observation. These partial derivatives are often referred to as “pseudo-residuals”. A regression tree is then grown to the pseudo-residuals, thereby using partitions of predictor space to group similar pseudo-residuals together. Fitted values for terminal nodes are found

by computing the addition to the current fit that minimizes the loss function. The current fitted values are then updated by adding the computed terminal node values to the existing fitted values for each observation.

Friedman (2001) originally proposed the above algorithm under the name “gradient boosting.” In this implementation, each regression tree was constructed on the entire set of training data. Friedman (2002) modified the algorithm to select a random bootstrap sample of training observations for growing each regression tree, resulting in the name *stochastic* gradient boosting. This subsampling step results in variance reduction, which improves performance through lower generalization error.

SGB contains a number of important tuning parameters as well. The first is the “learning rate” ν . The update to the current fitted values are shrunk by ν to control how quickly the algorithm descends down the gradient of the loss function. The learning rate provides regularization through shrinkage. The second parameter is the number of boosting iterations T . It is well-known that boosting algorithms overfit the data as the number of iterations grows and too many iterations can result in poor generalization error. There is a tradeoff between ν and T , however. Lower values of ν allow for more iterations without too much overfitting. However, this relationship between ν and T is not direct, and hence Ridgeway (2006) recommends choosing a small value of ν to not descend the gradient too quickly and then select T via cross-validation. The third important tuning parameter is the depth to which the regression trees are grown. Deeper trees allow for higher order interaction effects. Depths between 2-6 work well in practice (Ridgeway, 2006) and can be chosen via cross-validation.

Algorithm 2 provides the pseudocode for **SGB** for general loss functions. Commonly used loss functions include squared error loss or absolute loss for continuous responses, and Bernoulli deviance or Huberized hinge loss for binary classification.

Implementations of loss functions for Poisson-distributed, multinomial, and censored outcomes have been developed as well. The reader is referred to Ridgeway (2006) for a more complete discussion of available loss functions.

Algorithm 2 Stochastic Gradient Boosting

1. Initialize $f_0(\mathbf{x}) = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma)$
 2. For $t = 1$ to T :
 - (a) Draw a bootstrap sample \mathbf{X}^* of size N from the training data:
 - (b) For $i = 1, 2, \dots, N$ compute:

$$r_{im} = - \left\{ \frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)} \right\}_{f=f_{t-1}} \quad i \in \mathbf{X}^*$$
 - (c) Fit a regression tree to the r_{it} yielding terminal nodes R_{jt} , $j = 1, \dots, J_t$
 - (d) For $j = 1, 2, \dots, J_t$ compute:

$$\gamma_{jt} = \underset{\nu}{\operatorname{argmin}} \sum_{\mathbf{x}_i \in R_{jt}} L(y_i, f_{t-1}(\mathbf{x}) + \gamma)$$
 - (e) Update $f_t(\mathbf{x}) = f_{t-1} + \nu \cdot \sum_{j=1}^{J_t} \gamma_{jt} I(\mathbf{x} \in R_{jt})$
 3. Output $\hat{f}(\mathbf{x}) = f_T(\mathbf{x})$
-

Throughout this work, we use the **SGB** implementation provided in the R package **gbm** (Ridgeway, 2006).

1.4 Bayesian Additive Regression Trees

BART differs from **RF** and **SGB** by relying on an underlying probability model to generate estimates of some unknown function f rather than a purely algorithmic approach. **BART** uses a Bayesian probability model to generate a posterior distribution for $f(\mathbf{x})$. In this section, we extensively develop the **BART** model. The ensuing development is adapted from Kapelner and Bleich (2015) and also draws from Chipman et al. (2010).

BART can be considered a sum-of-trees ensemble. While single decision trees are effective for capturing nonlinearities and interaction effects in f , a sum-of-trees model allows for better fitting of additive components in f . Specifically, the **BART** model can be expressed as:

$$\mathbf{Y} = f(\mathbf{X}) + \boldsymbol{\varepsilon} \approx \mathfrak{T}_1^{\mathcal{L}}(\mathbf{X}) + \mathfrak{T}_2^{\mathcal{L}}(\mathbf{X}) + \dots + \mathfrak{T}_m^{\mathcal{L}}(\mathbf{X}) + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}_n(\mathbf{0}, \sigma^2 \mathbf{I}_n) \quad (1.8)$$

where \mathbf{Y} is the $n \times 1$ random vector of responses, \mathbf{X} is the $n \times p$ design matrix (the predictors column-joined) and $\boldsymbol{\varepsilon}$ is the $n \times 1$ vector of noise. Here we have m distinct regression trees, each composed of a tree structure, denoted by \mathfrak{T} , and the parameters at the terminal nodes (also called leaves), denoted by \mathcal{L} . The two together, denoted as $\mathfrak{T}^{\mathcal{L}}$ represents an entire tree with both its structure and set of leaf parameters.

The structure of a given tree \mathfrak{T}_t includes all of the splitting rules, allowing one to specify how any observation traverses down the tree. We denote the parameters for the leaves of the tree as $\mathcal{L}_t = \{\mu_{t,1}, \mu_{t,2}, \dots, \mu_{t,b_t}\}$ where b_t is the number of terminal nodes for a given tree. An observation's predicted value is the sum of the m leaf values arrived at by traversing down all m trees.

As a Bayesian model, **BART** consists of a set of priors for the structure and the leaf parameters and a likelihood for data in the terminal nodes. The aim of the priors is to provide regularization, preventing any single regression tree from dominating the total fit. We first provide an overview of the priors for **BART** and likelihood and then discuss how draws from the posterior distribution are made.

1.4.1 Priors and likelihood

The prior for the **BART** model has three components: (1) the tree structure itself, (2) the leaf parameters given the tree structure, and (3) the error variance σ^2 which is independent of the tree structure and leaf parameters

$$\mathbb{P}\left(\mathfrak{F}_1^{\text{leaf}}, \dots, \mathfrak{F}_m^{\text{leaf}}, \sigma^2\right) = \left[\prod_t \mathbb{P}\left(\mathfrak{F}_t^{\text{leaf}}\right) \right] \mathbb{P}\left(\sigma^2\right) \quad (1.9)$$

$$= \left[\prod_t \mathbb{P}\left(\mathcal{L}_t \mid \mathfrak{F}_t\right) \mathbb{P}\left(\mathfrak{F}_t\right) \right] \mathbb{P}\left(\sigma^2\right) \quad (1.10)$$

$$= \left[\prod_t \prod_{\ell} \mathbb{P}\left(\mu_{t,\ell} \mid \mathfrak{F}_t\right) \mathbb{P}\left(\mathfrak{F}_t\right) \right] \mathbb{P}\left(\sigma^2\right) \quad (1.11)$$

where the last equality follows from an additional assumption of conditional independence of the leaf parameters given the tree’s structure.

We first describe $\mathbb{P}\left(\mathfrak{F}_t\right)$, the component of the prior which affects the locations of nodes within the tree. Node depth is defined as distance from the root. Thus, the root itself has depth 0, its first child node has depth 1, etc. Nodes at depth d are nonterminal with prior probability $\alpha(1+d)^{-\beta}$ where $\alpha \in (0, 1)$ and $\beta \in [0, \infty]$. This component of the tree structure prior has the ability to enforce shallow tree structures, thereby limiting complexity of any single tree and resulting in more model regularization. Default values for these hyperparameters of $\alpha = 0.95$ and $\beta = 2$ are recommended by Chipman et al. (2010).

For nonterminal nodes, splitting rules occur in two parts. First, a predictor is randomly selected to serve as the splitting variable. In the original formulation, each available predictor is equally likely to be chosen from a discrete uniform distribution, and hence each variable is selected with probability $1/p$. This is relaxed in our implementation to allow for a generalized Bernoulli distribution where the user specifies p_1, p_2, \dots, p_p (such that $\sum_{j=1}^p p_j = 1$), where p_j denotes the probability of the j th variable being selected *a priori*. This more general prior will be developed further in Chapter 3. Additionally, note that “structural zeroes,” variables that do not have any valid split values, are assigned probability zero in the implementation of the al-

gorithm (see Appendix A.1.2 for details). Once the splitting variable is chosen, the splitting value is chosen from the multiset (the non-unique set) of available values at the node via the discrete uniform distribution.

We now describe the prior component $\mathbb{P}\left(\mathcal{L}_t \mid \mathfrak{F}_t\right)$ which controls the leaf parameters. Given a tree with a set of terminal nodes, each terminal node (or leaf) has a continuous parameter (the leaf parameter) representing the “best guess” of the response in this partition of predictor space. This parameter is the fitted value assigned to any observation that lands in that node. The prior on each of the leaf parameters is given as: $\mu_\ell \stackrel{iid}{\sim} \mathcal{N}(\mu_\mu/m, \sigma_\mu^2)$. The expectation, μ_μ , is picked to be at the range center, $(y_{\min} + y_{\max})/2$.

The variance hyperparameter σ_μ^2 is empirically chosen so that the range center plus or minus $k = 2$ variances cover 95% of the provided response values in the training set (where $k = 2$ corresponding to 95% coverage is only by default and can be customized). Thus, since there are m trees, we then choose σ_μ such that $m\mu_\mu - k\sqrt{m}\sigma_\mu = y_{\min}$ and $m\mu_\mu + k\sqrt{m}\sigma_\mu = y_{\max}$. The aim of this prior is to provide model regularization by shrinking the leaf parameters towards the center of the distribution of the response. The larger the value of k , the smaller the value of σ_μ^2 , resulting in more model regularization.

The final prior is on the error variance and is $\sigma^2 \sim \text{InvGamma}(\nu/2, \nu\lambda/2)$. λ is determined from the data so that there is a $q = 90\%$ a priori chance (by default) that the BART model will improve upon the root mean square error (RMSE) from an ordinary least squares regression. Therefore, the majority of the prior probability mass lies below the RMSE from least squares regression. Additionally, this prior limits the probability mass placed on small values of σ^2 to prevent overfitting. Thus, the higher the value of q , the larger the values of the sampled σ^2 's, resulting in more model regularization.

Note that the adjustable hyperparameters are α , β , k , ν and q . Additionally, m , the number of trees, must be chosen. Default values generally provide good performance, but optimal tuning can be achieved automatically via cross-validation.

Along with a set of priors, **BART** specifies the likelihood of responses in the terminal nodes. They are assumed a priori normal with the mean being the “best guess” in the leaf at the moment (i.e. in the current Markov chain Monte Carlo (MCMC) iteration) and variance being the best guess of the variance at the moment, $\mathbf{y}_\ell \sim \mathcal{N}(\mu_\ell, \sigma^2)$.

1.4.2 Posterior distribution and prediction

A Metropolis-within-Gibbs sampler (Geman and Geman, 1984; Hastings, 1970) is employed to generate draws from the posterior distribution of $\mathbb{P}(\mathfrak{T}_1^{\mathcal{L}}, \dots, \mathfrak{T}_m^{\mathcal{L}}, \sigma^2 \mid \mathbf{y})$. A key feature of this sampler for **BART** is to employ a form of “Bayesian backfitting” (Hastie and Tibshirani, 2000) where the j th tree is fit iteratively, holding all other $m - 1$ trees constant by exposing only the residual response that remains unfitted:

$$\mathbf{R}_{-j} := \mathbf{y} - \sum_{t \neq j} \mathfrak{T}_t^{\mathcal{L}}(\mathbf{X}). \quad (1.12)$$

The sampler,

$$\begin{aligned} 1 : & \quad \mathfrak{T}_1 \mid \mathbf{R}_{-1}, \sigma^2 \\ 2 : & \quad \mathcal{L}_1 \mid \mathfrak{T}_1, \mathbf{R}_{-1}, \sigma^2 \\ 3 : & \quad \mathfrak{T}_2 \mid \mathbf{R}_{-2}, \sigma^2 \\ 4 : & \quad \mathcal{L}_2 \mid \mathfrak{T}_2, \mathbf{R}_{-2}, \sigma^2 \\ & \quad \vdots \end{aligned} \quad (1.13)$$

$$\begin{aligned}
2m-1 : \quad & \mathfrak{T}_m \mid \mathbf{R}_{-m}, \sigma^2 \\
2m : \quad & \varnothing_m \mid \mathfrak{T}_m, \mathbf{R}_{-m}, \sigma^2 \\
2m+1 : \quad & \sigma^2 \mid \mathfrak{T}_1, \varnothing_1, \dots, \mathfrak{T}_m, \varnothing_m, \boldsymbol{\mathcal{E}},
\end{aligned}$$

proceeds by first proposing a change to the first tree's structure \mathfrak{T} which are accepted or rejected via a Metropolis-Hastings step. Note that sampling from the posterior of the tree structure does not depend on the leaf parameters, as they can be analytically integrated out of the computation (see Appendix A.1.2). Given the tree structure, samples from the posterior of the b leaf parameters $\varnothing_1 := \{\mu_1, \dots, \mu_b\}$ are then drawn. This procedure progresses iteratively for each tree, using the updated set of partial residuals \mathbf{R}_{-j} . Finally, conditional on the updated set of tree structures and leaf parameters, a draw from the posterior of σ^2 is made based on the full model residuals $\boldsymbol{\mathcal{E}} := \mathbf{y} - \sum_{t=1}^m \mathfrak{T}_t^{\varnothing}(\mathbf{X})$.

Within a given terminal node, since both the prior and likelihood are normally distributed, the posterior of each of the leaf parameters in \varnothing is conjugate normal with its mean being a weighted combination of the likelihood and prior parameters (lines 2, 4, \dots , $2m$ in Equation set 1.13). Due to the normal-inverse-gamma conjugacy, the posterior of σ^2 is inverse gamma as well (line $2m+1$ in Equation set 1.13). The complete expressions for these posteriors can be found in Gelman et al. (2004).

Lines 1, 3, \dots , $2m-1$ in Equation set 1.13 rely on Metropolis-Hastings draws from the posterior of the tree distributions. These involve introducing small perturbations to the tree structure: growing a terminal node by adding two child nodes, pruning two child nodes (rendering their parent node terminal), or changing a split rule. We denote the three possible tree alterations as: GROW, PRUNE, and CHANGE.¹ The

¹In the original formulation, Chipman et al. (2010) include an additional alteration called SWAP. In the implementation of BART used herein, this step is omitted and hence we exclude it from the discussion.

mathematics associated with the Metropolis-Hastings step are tedious. Appendix A.1 contains the explicit calculations. Once again, over many MCMC iterations, trees evolve to capture the fit left currently unexplained.

All $2m + 1$ steps represent a *single* Gibbs iteration. Empirical work suggests that no more than 1,000 iterations are needed as “burn-in,” although diagnostics can be used to assess the MCMC chain (see Chapter 2). An additional 1,000 iterations are usually sufficient to serve as draws from the posterior for $f(\mathbf{x})$. A single predicted value $\hat{f}(\mathbf{x})$ can be obtained by taking the average of the posterior values and a quantile estimate can be obtained by computing the appropriate quantile of the posterior values. Additional features of the posterior distribution will be discussed in Chapter 2.

1.4.3 BART for classification

BART can easily be modified to handle classification problems for categorical response variables. In Chipman et al. (2010), only binary outcomes were explored but recent work has extended BART to the multiclass problem (Kindo et al., 2013). All work herein focuses on the binary classification problem and we limit the discussion to that scenario.

For the binary classification problem (coded with outcomes “0” and “1”), we assume a probit model,

$$\mathbb{P}(Y = 1 \mid \mathbf{X}) = \Phi \left(\mathfrak{T}_1^{\circledast}(\mathbf{X}) + \mathfrak{T}_2^{\circledast}(\mathbf{X}) + \dots + \mathfrak{T}_m^{\circledast}(\mathbf{X}) \right), \quad (1.14)$$

where Φ denotes the cumulative distribution function of the standard normal distribution. In this formulation, the sum-of-trees model serves as an estimate of the conditional probit at \mathbf{x} which can be easily transformed into a conditional probability

estimate of $Y = 1$.

In the classification setting, the prior on σ^2 is not needed as the model assumes $\sigma^2 = 1$. The prior on the tree structure remains the same as in the regression setting and a few minor modifications are required for the prior on the leaf parameters.

Sampling from the posterior distribution is again obtained via Gibbs sampling with a Metropolis-Hastings step outlined in Section 1.4.2. Following the data augmentation approach of Albert and Chib (1993), an additional vector of latent variables \mathbf{Z} is introduced into the Gibbs sampler. Then, a new step is created in the Gibbs sampler where draws of $\mathbf{Z} \mid \mathbf{y}$ are obtained by conditioning on the sum-of-trees model:

$$Z_i \mid y_i = 1 \sim \max N \left(\sum_t \mathfrak{f}_t^{\otimes}(\mathbf{X}), 1 \right), 0 \text{ and} \quad (1.15)$$

$$Z_i \mid y_i = 0 \sim \min \left\{ N \left(\sum_t \mathfrak{f}_t^{\otimes}(\mathbf{X}), 1 \right), 0 \right\}. \quad (1.16)$$

Next, \mathbf{Z} is used as the response vector instead of \mathbf{y} in all steps of Equation 1.13.

Upon obtaining a sufficient number of samples from the posterior, inferences can be made using the the posterior distribution of conditional probabilities and classification can be undertaken by applying a threshold to the averages (or another summary) of these posterior probabilities.

1.4.4 BART Implementation

BART was implemented by the algorithm's original authors in the R package `BayesTree`. Chapter 2 develops a new novel R implementation for BART, `bartMachine`. Unless otherwise noted, we exclusively use `bartMachine` throughout this work.

Bayesian Additive Regression Trees Implementation

Abstract

We present a new package in `R` implementing `BART`. The package introduces many new features for data analysis using `BART` such as variable selection, interaction detection, model diagnostic plots, incorporation of missing data and the ability to save trees for future prediction. It is significantly faster than the current `R` implementation, parallelized, and capable of handling both large sample sizes and high-dimensional data.

2.1 Introduction

The initial implementation of `BART` was provided by Chipman et al. (2010) as a supplement to the original work. The algorithm was developed with a `C++` engine for constructing the ensemble with the output linked to `R` to allow for effective data analysis. This implementation is available on CRAN in the package `BayesTree`. `BayesTree`, however, provides limited capabilities for a data analyst and lacks features that allow it to be a robust platform for research and application. In this chapter,

we develop a novel implementation of **BART** that draws inspiration from **BayesTree**, but also attempts to remedy a number of its shortcomings in order to provide a more broadly applicable **BART** toolbox to the statistics and machine learning communities. The R package for our implementation, **bartMachine**, is available from the Comprehensive R Archive Network at <http://CRAN.R-project.org/package=bartMachine>.

We first highlight some key differences across **BART** implementations in Section 2.1.1 and then devote the rest of the chapter to elucidating the features of **bartMachine**. In Section 2.2 we provide a general introduction to the package, highlighting the novel features. Section 2.3 provides step-by-step examples of the regression capabilities and Section 2.4 introduces additional step-by-step examples of features unique to classification problems. We conclude in Section 2.5.

2.1.1 Comparison of BART Implementations

The goal of **bartMachine** is to provide a fast, easy-to-use, visualization-rich machine learning package for R users. In developing **bartMachine**, we explored other **BART** implementations and attempted to understand both the positives and negatives of each implementation of the algorithm.

One of the most critical drawbacks of **BayesTree** is its lack of a standalone **predict** function. Test data must be provided as an argument during the training phase of the model. Hence it is impossible to generate forecasts on future data without re-fitting with the entire model. Since the run time is not trivial, forecasting becomes an arduous exercise. A significantly faster implementation of **BART** that contains master-slave parallelization was introduced in Pratola et al. (2013), but this is only available as standalone C++ source code and not integrated with R. Additionally, a recent package **dbarts** allows updating of **BART** with new predictors and

response values to incorporate **BART** into a larger Bayesian model. **dbarts** relies on **BayesTree** as its **BART** engine.

Our implementation of **BART** is in **Java** and is integrated into **R** via **rJava** (Urbanek, 2011). From a runtime perspective, our algorithm is significantly faster than **BayesTree** and is parallelized, allowing computation on as many cores as desired. Not only is the model construction itself parallelized, but the additional features such as prediction, variable selection, and many others can be divided across cores as well.

We also include a variety of expanded and additional features. We implement the ability to save trees in memory and provide convenience functions for prediction on test data as well as the ability to save models across **R** sessions. We also include plotting functions for both posterior credible and predictive intervals and plots for visually inspecting the convergence of **BART**'s MCMC chain. We expand variable importance exploration to include permutation tests and interaction detection. We implement recently developed features for **BART** including a formal approach to variable selection and the ability to incorporate prior information for covariates (Chapter 3). We also implement a new strategy to incorporate missing data during training and handle missingness during prediction without imputation (Chapter 4). Table 2.1 emphasizes the differences in features between **bartMachine** and **BayesTree**, the two existing **R** implementations of **BART**.

2.2 The **bartMachine** package

The package **bartMachine** provides a novel implementation of **BART** in **R**. The algorithm is substantially faster than the current **R** package **BayesTree** and our implementation is parallelized at the MCMC iteration level during prediction. Additionally, the interface with **rJava** allows for the entire posterior distribution of tree ensembles

Feature	bartMachine	BayesTree
Implementation Language	Java	C++
External Predict Function	Yes	No
Model Persistence Across Sessions	Yes	No
Parallelization	Yes	No
Native Missing Data Mechanism	Yes	No
Built-in Cross-Validation	Yes	No
Variable Importance	Statistical Tests	Exploratory
Tree Proposal Types	3 Types	4 Types
Partial Dependence Plots	Yes	Yes
Convergence Plots	Assess trees and σ^2	Assess σ^2
Model Diagnostics	Yes	No
Incorporation into Larger Model	No	Through dbarts

Table 2.1: Comparison of features between **bartMachine** and **BayesTree**.

to persist throughout the R session, allowing for prediction and other calls to the trees without having to re-run the Gibbs sampler (a limitation in the current **BayesTree** implementation). The **bartMachine** object can be serialized, thereby persisting across R sessions as well (a feature discussed in Section 2.3.12). Since our implementation is different from **BayesTree**, we provide a predictive accuracy “bakeoff” on different datasets in Section 2.5.1 which illustrates that the two exhibit similar performance.

2.2.1 Speed improvements and parallelization

We make a number of significant speed improvements over the original **BayesTree** implementation.

First, **bartMachine** is fully parallelized (with the number of cores customizable) during model creation, prediction, and many of the other features. During model creation, we chose to parallelize by creating one independent Gibbs chain per core. Thus, if we employ the default 250 burn-in samples and 1,000 post burn-in samples and four cores, each core would sample 500 samples: 250 for a burn-in and 250

post burn-in samples. The final model will aggregate the four post burn-in chains for the four cores yielding the desired 1,000 total burn-in samples. This has the drawback of effectively running the burn-in serially (which suffers from Amdahl’s Law), but has the added benefit of reducing auto-correlation of the sum-of-trees samples in the posterior samples since the chains are independent which may provide greater predictive performance. Parallelization at the level of likelihood calculations is left for a future release as we were unable to address the costs of thread overhead. Parallelization for prediction and other features scale linearly in the number of cores without Amdahl’s diminishing returns.

Additionally, we take advantage of a number of additional computational short-cuts:

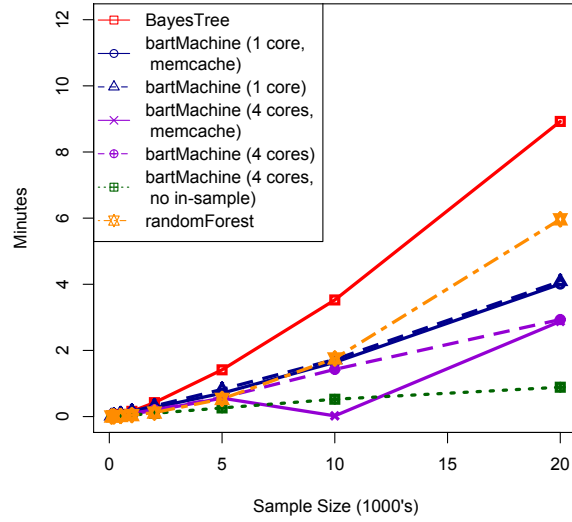
1. Computing the unfitted responses for each tree (Equation 1.12) can be accomplished by keeping a running vector and making entry-wise updates as the Gibbs sampler (Equation 1.13) progresses from step 1 to $2m$. Additionally, during the σ^2 sampling (step $2m + 1$), the residuals do not have to be computed by traversing the data down all the trees.
2. Each node caches its acceptable variables for split rules and the acceptable unique split values so they do not need to be calculated at each tree sampling step. Recall from the discussion concerning uniform splitting rules in Section 1.4.1 that acceptable predictors and values change based on the data available at an arbitrary location in the tree structure. This speed enhancement, which we call *memcache* comes at the expense of memory and may cause issues for large data sets. We include a toggle in our implementation defaulted to “on.”
3. Careful calculations in Appendix A.1 eliminate many unnecessary computations. For instance, the likelihood ratios are only functions of the squared sum of re-

sponses and no longer require computing the sum of the responses squared.

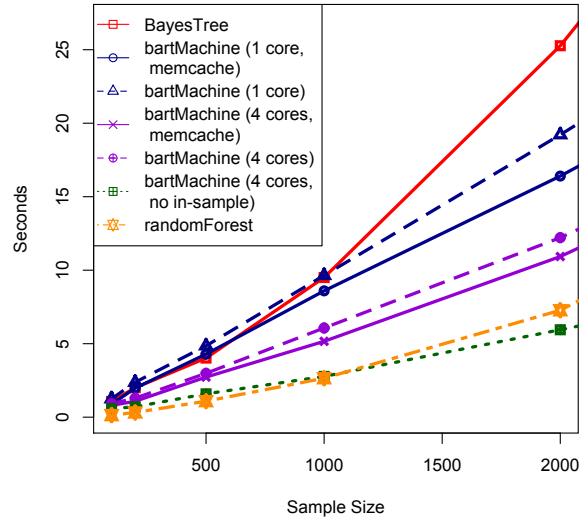
Figure 2.1 displays model creation speeds for different values of n on a linear regression model with $p = 20$, normally distributed covariates, $\beta_1, \dots, \beta_{20} \stackrel{iid}{\sim} U(-1, 1)$, and standard normal noise. Note that we do not vary p as it was already shown in Chipman et al. (2010) that **BART**’s computation time is largely unaffected by the dimensionality of the problem. We include results for **BART** using **BayesTree**, **bartMachine** with one core, **bartMachine** with four cores having the *memcache* option both on and off, and **bartMachine** with four cores, *memcache* off, and computation of in-sample statistics off (all with $m = 50$ trees). The in-sample statistics that are computed by default are in-sample predictions ($\hat{\mathbf{y}}$), residuals ($e := y - \hat{\mathbf{y}}$), $L1$ error which is defined as $\sum_{i=1}^{n_{\text{train}}} |e_i|$, $L2$ error which is defined as $\sum_{i=1}^{n_{\text{train}}} e_i^2$, pseudo- R^2 which is defined as $1 - L2/(\sum_{i=1}^{n_{\text{train}}} (y_i - \bar{y})^2)$ and RMSE which is defined as $\sqrt{L2/n_{\text{train}}}$. We also include random forests model creation times via the package **randomForest** (Liaw and Wiener, 2002) with its default settings.

We first note that Figure 2.1a demonstrates that the **bartMachine** model creation runtime is approximately linear in n (without in-sample statistics computed). There is about a 30% speed-up when using four cores instead of one. The *memcache* enhancement should be turned off only with sample sizes larger than $n = 20,000$ (data unshown). Noteworthy is the 50% reduction in time of constructing the model when not computing in-sample statistics. In-sample statistics are computed by default because the user generally wishes to see them. Also, for the purposes of this comparison, **BayesTree** models compute the in-sample statistics by necessity since the trees are not saved. The **randomForest** implementation becomes slower just after $n = 1,000$ due to its reliance on greedy exhaustive search at each node.

Figure 2.1b displays results for smaller sample sizes ($n \leq 2,000$) that are often encountered in practice. We observe the *memcache* enhancement provides about a



(a) Large sample sizes



(b) Small sample sizes

Figure 2.1: Model creation times as a function of sample size for a number of settings of `bartMachine`, `BayesTree` and `randomForest`. Simulations were run on a quad-core 3.4GHz Intel i5 desktop with 24GB of RAM running the Windows 7 64bit operating system.

10% speed improvement. Thus, if memory is an issue, it can be turned off with little performance degradation.

2.2.2 Implementation of Tree Alterations

Additionally, recall from Section 1.4.2, that there are 4 possible proposals for altering the trees in **BART** originally proposed by Chipman et al. (2010): GROW, PRUNE, CHANGE, and SWAP. **bartMachine** does not implement SWAP due to complexities that arise in implementation. Additionally, Pratola et al. (2013) argue that a CHANGE step is unnecessary for sufficient mixing of the Gibbs sampler. While we too observed this to be true for estimates of the posterior means, we found that omitting CHANGE can negatively impact the variable inclusion proportions (the feature introduced in Section 2.3.5). As a result, we implement a modified CHANGE step where we only propose new splits for nodes that are singly internal (versus the original proposal of changing any splitting rule in a tree). These are nodes where both children nodes are terminal nodes (details are given in Appendix A.1.4). After a singly internal node is selected we (1) select a new split attribute from the set of available predictors and (2) select a new split value from the multiset of available values (these two uniform splitting rules were explained in detail previously). We emphasize that the CHANGE step does not alter tree structure.

2.2.3 Implementation of Research Features

The current stable release of **bartMachine** available on CRAN implements the variable selection and informed prior procedures introduced in Chapter 3 and the method for natively handling missing data proposed in Chapter 4. An experimental version of the package also implements the heteroskedasticity augmentation developed in

Chapter 5. Future work will involve incorporating this last feature into the package available on CRAN.

2.3 bartMachine Package Features for Regression

We illustrate the package features by using both real and simulated data, focusing first on regression problems.

2.3.1 Computing parameters

We first set some computing parameters. In this exploration, we allow up to 5GB of RAM for the Java heap² and we set the number of computing cores available for use to 4.

```
R> options(java.parameters = "-Xmx5000m")
R> library("bartMachine")
R> set_bart_machine_num_cores(4)
bartMachine now using 4 cores.
```

The following Sections 2.3.2 – 2.3.9 use a dataset obtained from UCI repository (Bache and Lichman, 2013). The $n = 201$ observations are automobiles and the goal is to predict each automobile's price from 25 features (15 continuous and 10 nominal), first explored by Kibler et al. (1989).³ This dataset also contains missing data. We omit missing data for now (41 observations that will later be retained in Section 2.3.8)

²Note that the maximum amount of memory can be set only *once* at the beginning of the R session (a limitation of `rJava` since only one Java Virtual Machine can be initiated per R session), but the number of cores can be respecified at any time.

³We first preprocess the data. We first drop one of the nominal predictors (car company) due to too many categories (22). We then coerce two of the of the nominal predictors to be continuous. Further, the response variable, price, was logged to reduce right skew in its distribution.

and we create a variable for the design matrix \mathbf{X} and the response \mathbf{y} . The following code loads the data.

```
R> data(automobile)
R> automobile <- na.omit(automobile)
R> y <- automobile$log_price
R> X <- automobile; X$log_price <- NULL
```

2.3.2 Model building

We are now ready to construct a `bartMachine` model. The default hyperparameters generally follow the recommendations of Chipman et al. (2010) and provide a ready-to-use algorithm for many data problems. Our hyperparameter settings are $m = 50$,⁴ $\alpha = 0.95$, $\beta = 2$, $k = 2$, $q = 0.9$, $\nu = 3$, and probabilities of the GROW / PRUNE / CHANGE steps is 28% / 28% / 44%. We retain the default number of burn-in Gibbs samples (250) as well as the default number of post-burn-in samples (1,000). In the default mode, the covariates are equally important *a priori*. Other parameters and their defaults can be found in the package’s online manual. Below is a default `bartMachine` model. Here, \mathbf{X} denotes automobile attributes and \mathbf{y} denotes the log price of the automobile.

```
R> bart_machine <- bartMachine(X, y)

Building bartMachine for regression ...

evaluating in sample data...done
```

⁴In contrast to Chipman et al. (2010), we recommend this default as a good starting point rather than $m = 200$ due to our experience experimenting with the “RMSE by number of trees” feature found later in this section. Performance is often similar and computational time and memory requirements are dramatically reduced.

If one wishes to see more information about the individual iterations of the Gibbs sampler of Equation 5.5, the flag `verbose` can be set to “TRUE.” One can see more debug information from the Java program by setting the flag `debug_log` to TRUE and the program will print to `unnamed.log` in the current working directory. In the following code segment, we inspect the model object to query its in-sample performance and to be reminded of the input data and model hyperparameters.

```
R> bart_machine
bartMachine v1.1.1 for regression

training data n = 160 and p = 41
built in 1.7 secs on 4 cores, 50 trees, 250 burn-in
and 1000 post. samples

sigsq est for y beforehand: 0.014
avg sigsq estimate after burn-in: 0.00794

in-sample statistics:

L1 = 8.01
L2 = 0.65
rmse = 0.06
Pseudo-Rsq = 0.979
p-val for shapiro-wilk test of normality of residuals: 0.04584
p-val for zero-mean noise: 0.97575
```

The above output provides a summary for a default `bartMachine` model built with the automobile data. Since the response was continuous, `bartMachine` for regression

was employed automatically. First, the output prints the dimensions of the design matrix. Then, it prints the creation time along with other model parameters. Next the output prints the MSE for the OLS model and displays the **bartMachine** model's estimate of σ_e^2 . We are then given in-sample statistics on error. The pseudo- R^2 is calculated via $1 - SSE/SST$. Also provided are outputs from tests of the error distribution being normal and mean centered. Note that the p-value for Shapiro-Wilk test of normality of residuals is marginally less than 5%. Thus we conclude that the noise of Equation 1.8 is not normally distributed. Just as when interpreting the results from a linear model, non-normality implies we should be circumspect concerning **bartMachine** output that relies on this distributional assumption such as the credible and prediction intervals of Section 2.3.4.

We can also obtain out-of-sample statistics to assess level of overfitting by using k-fold cross-validation. Using 10 randomized folds we find:

```
R> k_fold_cv(X, y, k_folds = 10)
.....
$L1_err
[1] 21.64303

$L2_err
[1] 4.742511

$rmse
[1] 0.1721647

$PseudoRsq
[1] 0.8467881
```

The code provides the out of sample statistics for the model built above. This function also returns the $\hat{\mathbf{y}}$ predictions as well as the vector of the fold indices (which are omitted in the output shown above).

The Pseudo- R^2 being lower out-of-sample (above) versus in-sample suggests that `bartMachine` is slightly overfitting (note also that the training sample during cross-validation is 10% smaller).

It may also be of interest to see how the number of trees m affects performance. One can examine how out-of-sample predictions vary by the number of trees via

```
R> rmse_by_num_trees(bart_machine, num_replicates = 20)
```

and the output is shown in Figure 2.2. This illustration suggests that predictive performance levels off around $m = 50$. We observe similar illustrations across a wide variety of datasets and hyperparameter choices which is the reason we have set $m = 50$ as the default value in the package.

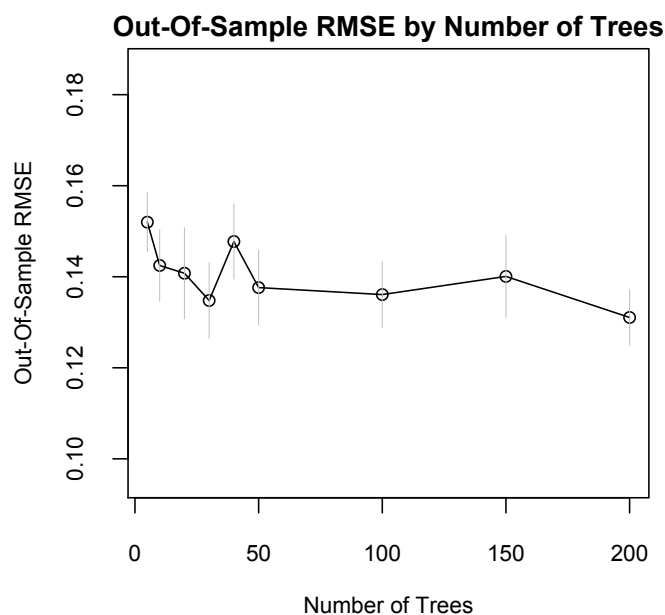


Figure 2.2: Out-of-sample predictive performance by number of trees.

Note that there is nominal improvement at $m = 200$. There may also be improvement if other hyperparameters are varied. We can attempt to build a better `bartMachine` model using the procedure `bartMachineCV` by grid-searching over a set of hyperparameter combinations, including m (for more details, see BART-cv in Chipman et al., 2010). The grid of interest can be customized by the user and defaults to a small grid.

```
R> bart_machine_cv <- bartMachineCV(X, y)
...
bartMachine CV win: k: 2 nu, q: 3, 0.9 m: 200
```

This function returns the “winning” model, which is the one with lowest out-of-sample RMSE over a 5-fold (by default) cross-validation. Here, the cross-validated `bartMachine` model has slightly better in-sample performance ($L1 = 8.18$, $L2 = 0.68$ and $\text{Pseudo-}R^2 = 0.978$) in comparison to the initial BART model as well as slightly better out-of-sample performance ($L1 = 21.05$, $L2 = 4.40$ and $\text{Pseudo-}R^2 = 0.858$) when assessed via:

```
R> k_fold_cv(X, y, k_folds = 10, k = 2, nu = 3, q = 0.9,
num_trees = 200)
```

Predictions are handled with the `predict` function. Below are fits for the first seven observations.

```
R> predict(bart_machine_cv, X[1 : 7, ])
[1] 9.49 9.78 9.79 10.05 9.67 9.70 9.91
```

We also include a convenience method `bart_predict_for_test_data` that will predict and return out-of-sample error metrics when the test outcomes are known.

2.3.3 Assumption checking

The package includes features that assess the plausibility of the **BART** model assumptions. Checking the mean-centeredness of the noise is addressed in the summary output of a constructed model and is simply a one-sample t -test of the average residual value against a null hypothesis of true mean zero. We assess both normality and heteroskedasticity via:

```
R> check_bart_error_assumptions(bart_machine_cv)
```

This will display a plot similar to Figure 2.3 which contains a QQ-plot (to assess normality) as well as a residual-by-predicted plot (to assess homoskedasticity). There is little evidence that the errors violate the assumptions of normality and homoskedasticity.

In addition to the model assumptions, **BART** requires convergence of its Gibbs sampler which can be investigated via:

```
R> plot_convergence_diagnostics(bart_machine_cv)
```

Figure 2.4 displays the plot which features four types of convergence diagnostics (each are detailed in the figure caption). The top left plot shows σ^2 by MCMC iteration. Samples to the left of the first vertical grey line are burn-in from the first computing core's MCMC chain. The four subsequent plots separated by grey lines are the post-burn-in iterations from each of the four computing cores employed during model construction. The top right plot shows the percent acceptance of Metropolis-Hastings proposals across the m trees where each point plots one iteration. Points before the grey vertical line illustrate burn-in iterations and points after illustrate post burn-in iterations. Each computing core is colored differently. The bottom left plot shows the average number of leaves across the m trees by iteration (post burn-in

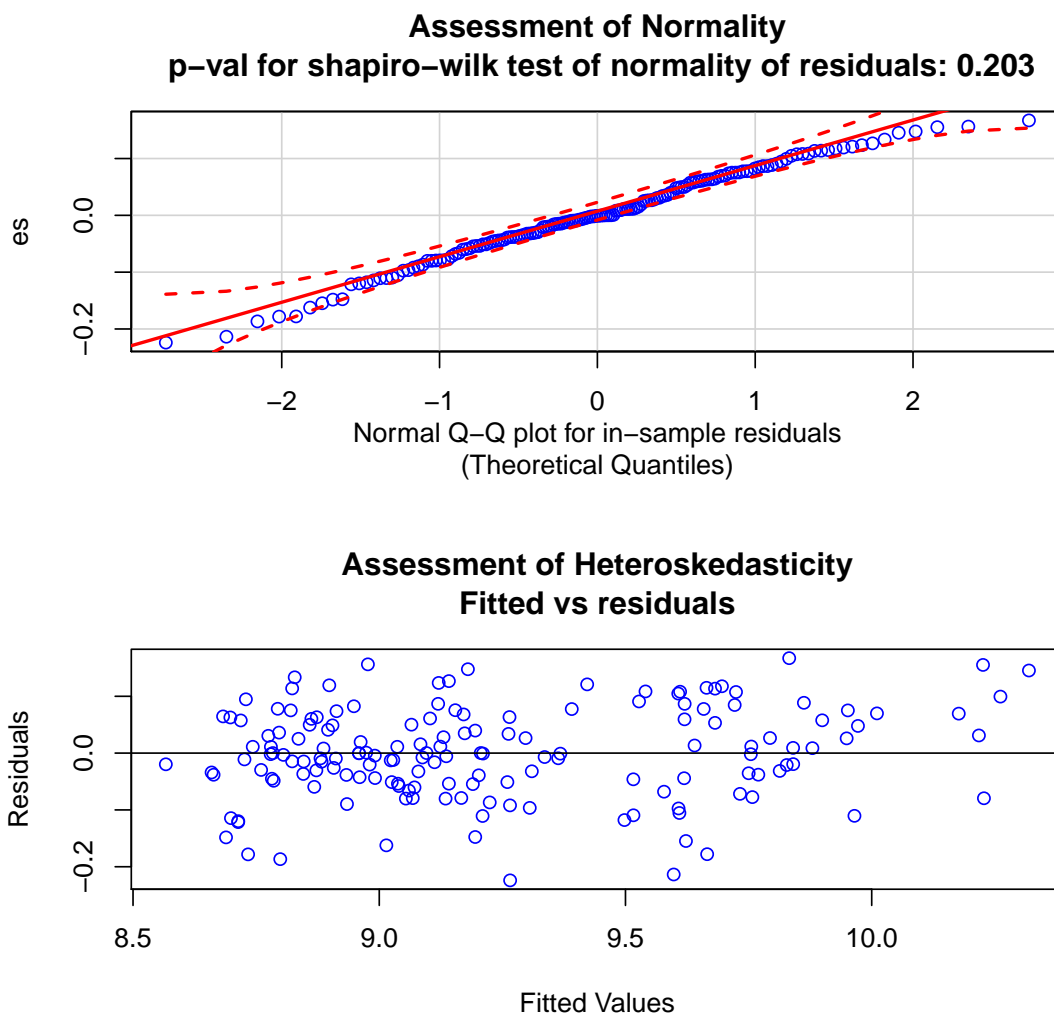


Figure 2.3: Test of normality of errors using QQ-plot and the Shapiro-Wilk test (top), residual plot to assess heteroskedasticity (bottom).

only where computing cores separated by vertical grey lines). Finally, the bottom right plot shows average tree depth across the m trees by iteration (post burn-in only where computing cores separated by vertical grey lines). Overall, visual inspection of the plots suggests that the `bartMachine` model has been sufficiently burned-in as each of the plots seems to exhibit a stationary process.

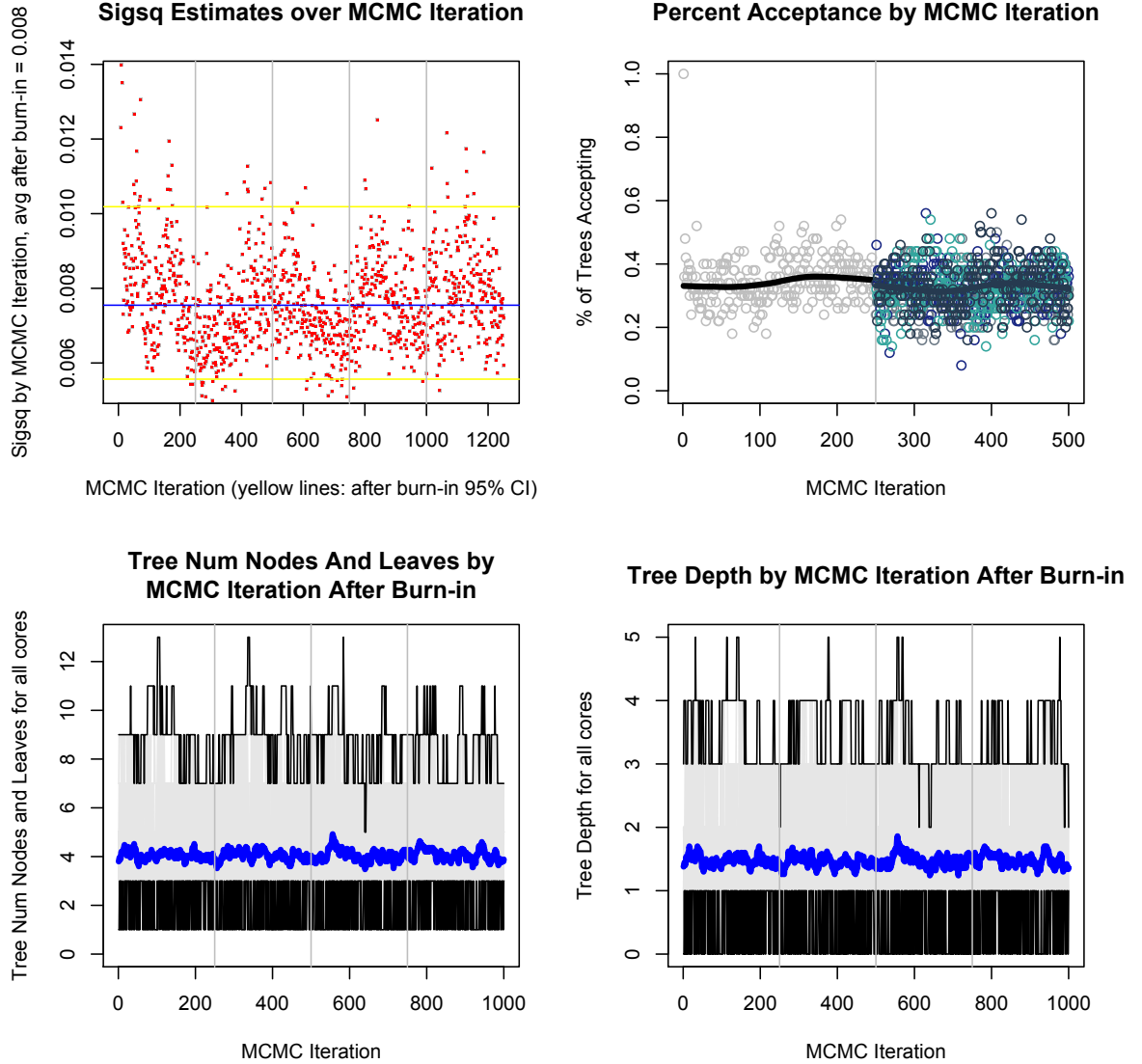


Figure 2.4: Convergence diagnostics for the cross-validated `bartMachine` model.

2.3.4 Credible intervals and prediction intervals

An advantage of **BART** is that if we believe the priors and model assumptions, the Bayesian probability model and corresponding burned-in MCMC iterations provide the approximate posterior distribution of $f(\mathbf{x})$. Thus, one can compute uncertainty estimates via quantiles of the posterior samples. These provide Bayesian “credible intervals” which are intervals for the conditional expectation function, $\mathbb{E}[\mathbf{y} \mid \mathbf{X}]$.

Another useful uncertainty interval can be computed for individual predictions by combining uncertainty from the conditional expectation function with the systematic, homoskedastic normal noise produced by \mathcal{E} . This is accomplished by generating 1,000 samples (by default) from the posterior predictive distribution and then reporting the appropriate quantiles.

Below is an example of how both types of intervals are computed in the package (for the 100th observation of the training data):

```
R> calc_credible_intervals(bart_machine_cv, X[100, ], ci_conf = 0.95)
      ci_lower_bd ci_upper_bd
[1,]      8.725202      8.971687
R> calc_prediction_intervals(bart_machine_cv, X[100, ],
      pi_conf = 0.95)
      pi_lower_bd pi_upper_bd
[1,]      8.631243      9.06353
```

Note that the prediction intervals are wider than the credible intervals because they reflect the uncertainty from the error term.

We can then plot these intervals in sample:

```
R> plot_y_vs_yhat(bart_machine_cv, credible_intervals = TRUE)
R> plot_y_vs_yhat(bart_machine_cv, prediction_intervals = TRUE)
```

Figure 2.5a shows how our prediction fared against the original response (in-sample) with 95% credible intervals. Figure 2.5b shows the same prediction versus the original response plot now with 95% prediction intervals.

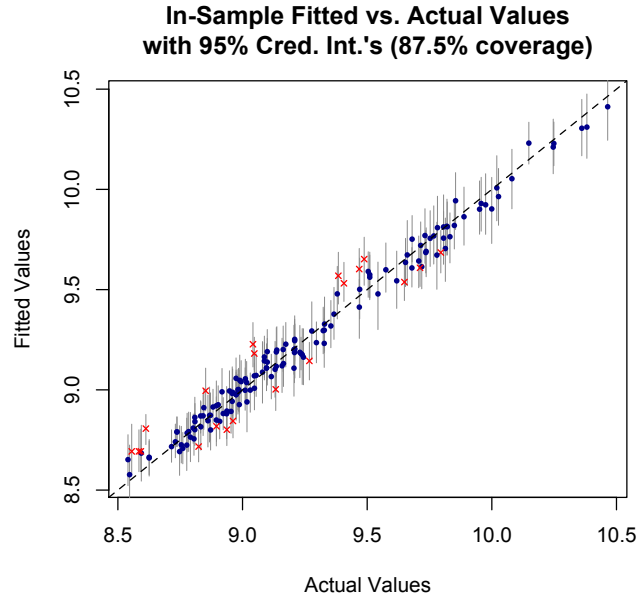
2.3.5 Variable importance

After a `bartMachine` model is built, it is natural to ask the question: which variables are most important? This is assessed by examining the splitting rules in the m trees across the post burn-in MCMC iterations which are known as “inclusion proportions” (Chipman et al., 2010). The inclusion proportion for any given predictor represents the proportion of times that variable is chosen as a splitting rule out of all splitting rules among the posterior draws of the sum-of-trees model. Figure 2.6 illustrates the inclusion proportions for all variables obtained via:

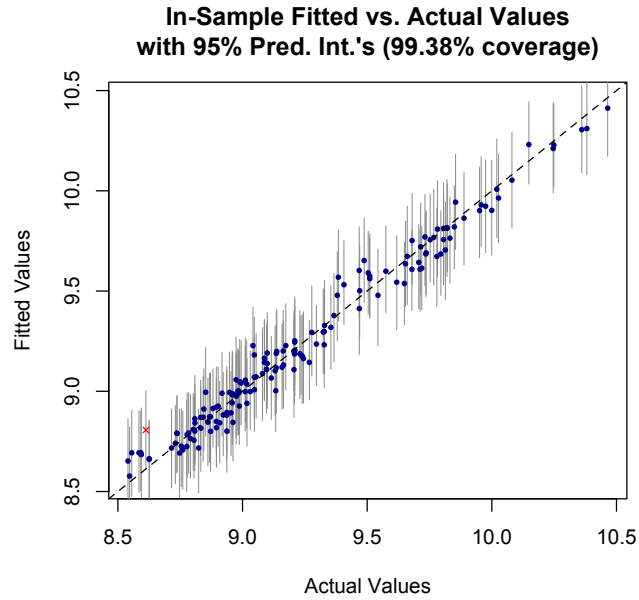
```
R> investigate_var_importance(bart_machine_cv,
  num_replicates_for_avg = 20)
```

2.3.6 Variable effects

It is also natural to ask: does \mathbf{x}_j affect the response, controlling for other variables in the model? This is roughly analogous to the t -test in ordinary least squares regression of no linear effect of \mathbf{x}_j on \mathbf{y} while controlling for all other variables, \mathbf{x}_{-j} . The null hypothesis here is the same but the linearity constraint is relaxed. To test this, we employ a permutation approach where we record the observed Pseudo- R^2 from the `bartMachine` model built with the original data. Then we permute the \mathbf{x}_j th column, thereby destroying any relationship between \mathbf{x}_j and \mathbf{y} (and the other predictors), construct a new duplicate `bartMachine` model from this permuted design matrix and



(a) Segments illustrate credible intervals



(b) Segments illustrate prediction intervals

Figure 2.5: Fitted versus actual response values for the automobile dataset. Segments are 95% credible intervals (a) or 95% prediction intervals (b). Green dots indicate the true response is within the stated interval and red dots indicate otherwise.

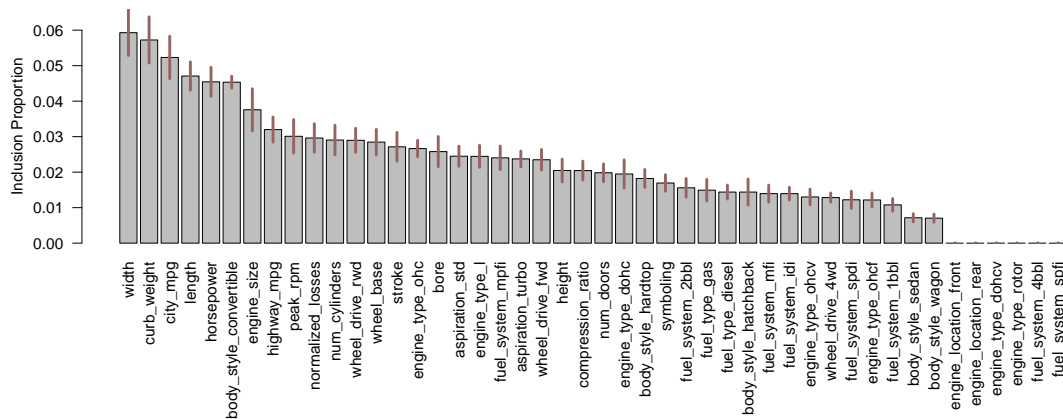


Figure 2.6: Average variable inclusion proportions in the cross-validated `bartMachine` model for the automobile data averaged over 100 model constructions to obtain stable estimates across many posterior modes in the sum-of-trees distribution (as recommended in Bleich et al., 2014). The segments on top of the bars represent 95% confidence intervals. The eight predictors with inclusion proportions of zero are predictors with identically one value (after missing data were dropped).

record a “null” Pseudo- R^2 . We then repeat this process to obtain a null distribution of Pseudo- R^2 ’s. Since the alternative hypothesis is that \mathbf{x}_j has an effect on \mathbf{y} in terms of predictive power, our p value is the proportion of null Pseudo- R^2 ’s greater than the observed Pseudo- R^2 , making our procedure a natural one-sided test. Note, however, that this test is conditional on the BART model and its selected priors being true, similar to the assumptions of the linear model.

If we wish to test if a set of covariates $A \subset \{\mathbf{x}_1, \dots, \mathbf{x}_p\}$ affect the response after controlling for other variables, we repeat the procedure outlined in the above paragraph by permuting the predictors in A in every null sample. We do not permute each column separately, but instead permute as a unit in order to preserve the collinearity structure in A . This is roughly analogous to the partial F -test in ordinary least squares regression.

If we wish to test if *any* of the covariates matter in predicting \mathbf{y} , we simply permute \mathbf{y} during the null sampling. This procedure breaks the relationship between the response and the predictors but does not alter the existing associations between predictors. This is roughly analogous to the omnibus F -test in ordinary least squares regression.

At $\alpha = 0.05$, Figure 2.7a demonstrates an insignificant effect of the variable **width** of car on price. Even though **width** is putatively the “most important” variable as measured by proportions of splits in the posterior sum-of-trees model (Figure 2.6), note that this is largely an easy prediction problem with many collinear predictors. Figure 2.7b shows the results of a test of the putatively most important categorical variable, **body style** (which involves permuting the categories, then dummifying the levels to preserve the structure of the variable). We find a marginally significant effect ($p = 0.0495$). A test of the top ten most important variables is convincingly significant (Figure 2.7c). For the omnibus test, Figure 2.7d illustrates an extremely

statistically significant result, as would be expected. The code to run these tests is shown below (output suppressed).

```
R> cov_importance_test(bart_machine_cv, covariates = c("width"))
R> cov_importance_test(bart_machine_cv, covariates = c("body_style"))
R> cov_importance_test(bart_machine_cv, covariates = c("width",
  "curb_weight", "city_mpg", "length", "horsepower", "body_style",
  "engine_size", "highway_mpg", "peak_rpm", "normalized_losses"))
R> cov_importance_test(bart_machine_cv)
```

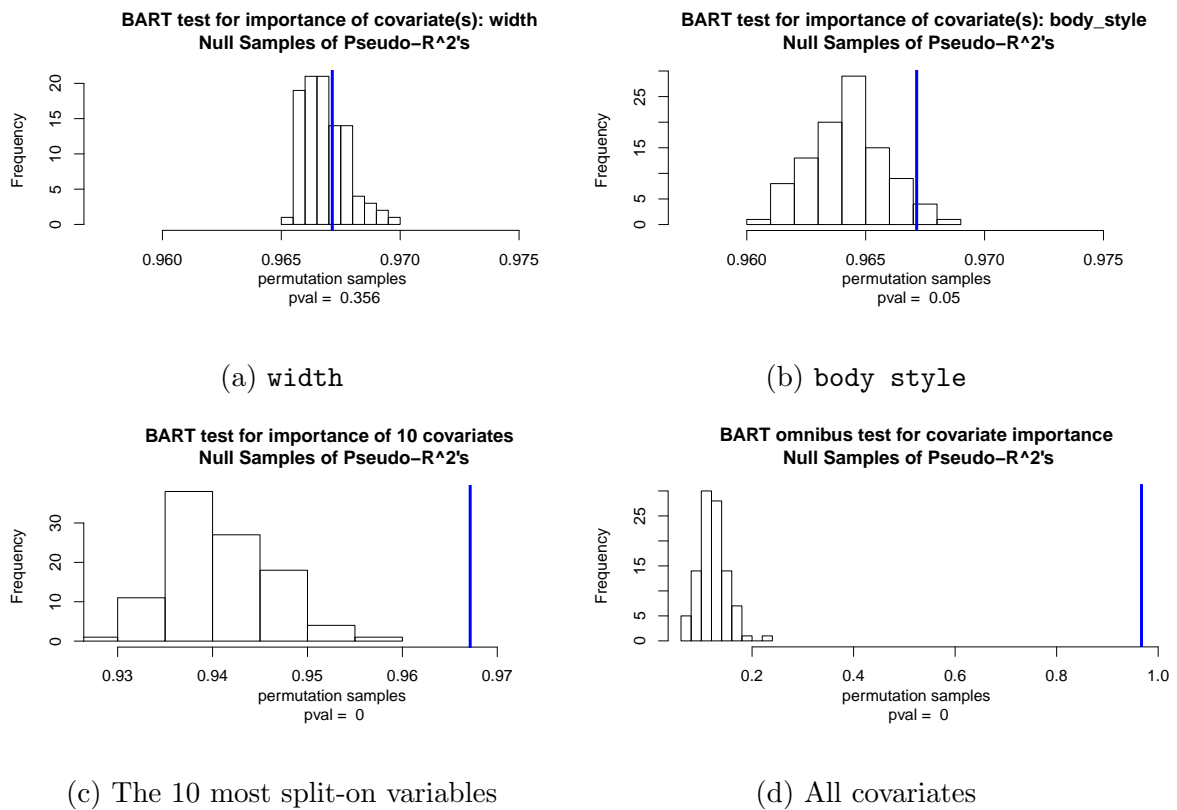


Figure 2.7: Tests of covariate importance conditional on the cross-validated `bartMachine` model. All tests performed with 100 null samples.

2.3.7 Partial dependence

A data analyst may also be interested in understanding how \mathbf{x}_j affects the response on average, after controlling for other predictors. This can be examined using Friedman (2001)'s Partial Dependence Function (PDP),

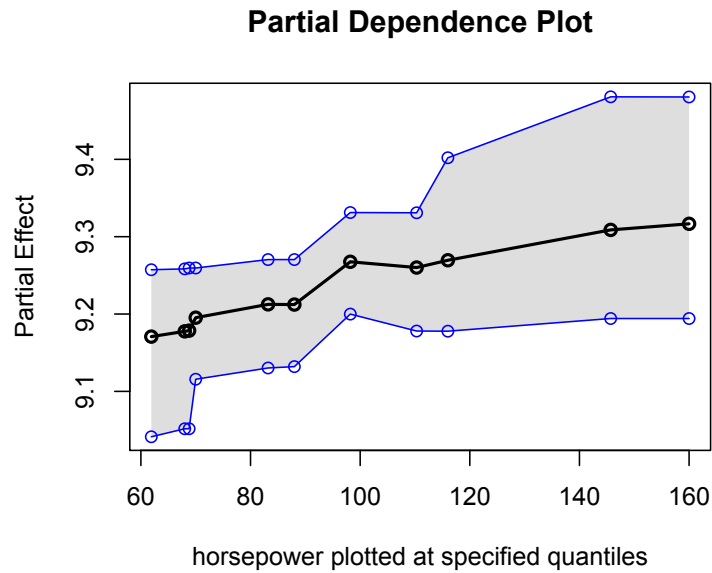
$$f_j(\mathbf{x}_j) = \mathbb{E}_{\mathbf{x}_{-j}} [f(\mathbf{x}_j, \mathbf{x}_{-j})] := \int f(\mathbf{x}_j, \mathbf{x}_{-j}) dP(\mathbf{x}_{-j}). \quad (2.1)$$

The PDP of predictor \mathbf{x}_j gives the average value of f when \mathbf{x}_j is fixed and \mathbf{x}_{-j} varies over its marginal distribution, $dP(\mathbf{x}_{-j})$. As neither the true model f nor the distribution of the predictors $dP(\mathbf{x}_{-j})$ are known, we estimate Equation 2.1 by computing

$$\hat{f}_j(\mathbf{x}_j) = \frac{1}{n} \sum_{i=1}^n \hat{f}(\mathbf{x}_j, \mathbf{x}_{-j,i}) \quad (2.2)$$

where n is the number of observations in the training data and \hat{f} denotes predictions via the `bartMachine` model. Since BART provides an estimated posterior distribution, we can plot credible bands for the PDP function. Credible bands are computed as follows: in Equation 2.2, the \hat{f} can be replaced with a function that calculates the q th quantile of the post-burned-in MCMC iterations for $\hat{\mathbf{y}}$. Figure 2.8a plots the PDP along with the 2.5%ile and the 97.5%ile for the variable `horsepower`. By varying over most of the range of `horsepower`, the price is predicted to increase by about \$1000, holding all else constant. Figure 2.8b plots the PDP along with the 2.5%ile and the 97.5%ile for the variable `stroke`. This predictor seemed to be relatively unimportant according to Figure 2.6 and the PDP confirms this, with a very small, yet nonlinear average partial effect. The code for both plots is below.

```
R> pd_plot(bart_machine_cv, j = "horsepower")
R> pd_plot(bart_machine_cv, j = "stroke")
```

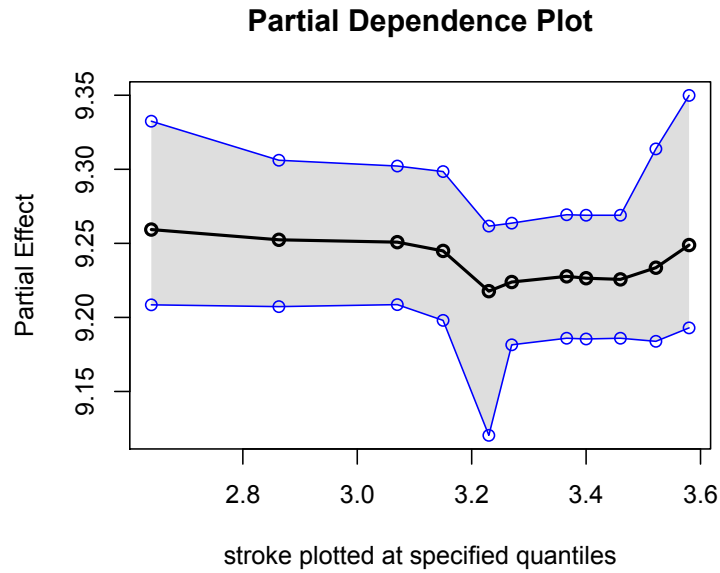


(a) horsepower

2.3.8 Incorporating missing data

The procedure for incorporating missing data will be formally developed in Chapter 4. Here, we briefly introduce how to build a `bartMachine` model using our procedure for incorporating missing data below:

```
R> y <- automobile$log_price
R> X <- automobile; X$log_price <- NULL
R> bart_machine <- bartMachine(X, y, use_missing_data = TRUE,
  use_missing_data_dummies_as_covars = TRUE)
```



(b) `stroke`

Figure 2.8: PDPs plotted in black and 95% credible intervals plotted in blue for variables in the automobile dataset. Points plotted are at the 5%ile, 10%ile, 20%ile, ..., 90%ile and 95%ile of the values of the predictor. Lines plotted between the points approximate the PDP by linear interpolation.

The model output below parallels the model output with the missing rows omitted (Section 2.3.2) with the key difference that the missing data feature has been turned on.

```
R> bart_machine
```

```
bartMachine v1.1.1 for regression
```

```
Missing data feature ON
```

```
training data n = 201 and p = 50
```

```
built in 1.4 secs on 1 core, 50 trees, 250 burn-in
```

```
and 1000 post. samples
```

```
sigsq est for y beforehand: 0.016
```

```
avg sigsq estimate after burn-in: 0.00939
```

```
in-sample statistics:
```

```
L1 = 11.49
```

```
L2 = 1.04
```

```
rmse = 0.07
```

```
3 Pseudo-Rsq = 0.9794
```

```
p-val for shapiro-wilk test of normality of residuals: 0.69814
```

```
p-val for zero-mean noise: 0.96389
```

Note that the output reflects the use of the complete data set. There are 41 observations now included for which there are missing features. Also note that p has now increased from 41 to 50. The nine “new” predictors are:

```
[1] "engine_location_rear" "engine_type_rotor"    "fuel_system_4bbl"
```

```
[4] "fuel_system_spfi"      "M_normalized_losses"  "M_bore"
```

```
[7] "M_stroke"           "M_horsepower"       "M_peak_rpm"
```

The first two predictors are two new levels for the variable `engine_location` which appear in the 41 rows with missingness. The next two predictors are two new levels for the variable `fuel_system` which appear in the 41 rows with missingness as well. The last five new predictors are dummy variables which indicate missingness constructed from the predictors which exhibited missingness (due to the `use_missing_data_dummies_as_covars` parameter being set to true).

The procedure developed in Chapter 4 also incorporates missing data during prediction. As will be shown, missingness in the data will yield larger credible intervals. In the example below, we suppose that the `curb_weight` and `symboling` values were suddenly unavailable for the 20th automobile and we observe their credible intervals widening as a result.

```
R> x_star <- X[20, ]
R> calc_credible_intervals(bart_machine, x_star, ci_conf = 0.95)
      ci_lower_bd ci_upper_bd
[1,]    8.650093    8.824515
R> x_star[c("curb_weight", "symboling")] <- NA
R> calc_credible_intervals(bart_machine, x_star, ci_conf = 0.95)
      ci_lower_bd ci_upper_bd
[1,]    8.622582    8.978313
```

2.3.9 Variable selection

In this section we demonstrate the variable selection procedures formally developed in Chapter 3. The following code will select variables based on the three thresholds

and also displays the plot in Figure 2.9.⁵

```
R> vs <- var_selection_by_permute(bart_machine,
                                bottom_margin = 10, num_permute_samples = 10)
R> vs$important_vars_local_names
"curb_weight" "city_mpg" "engine_size" "horsepower"
"length"      "width"    "num_cylinders" "body_style_convertible"
"wheel_base"  "peak_rpm" "highway_mpg"  "wheel_drive_fwd"
R> vs$important_vars_global_max_names
"curb_weight" "city_mpg" "engine_size" "horsepower" "length"
R> vs$important_vars_global_se_names
"curb_weight" "city_mpg" "engine_size" "horsepower" "length"
"width"       "num_cylinders" "wheel_base" "wheel_drive_fwd"
```

As will be shown, in many situations it will not be clear to the data analyst which threshold is most appropriate. The fourth procedure we will introduce can choose the “best” procedure via cross-validation using RMSE as follows:

```
var_selection_by_permute_response_cv(bart_machine)
$best_method
[1] "important_vars_local_names"

$important_vars_cv
[1] "body_style_convertible" "city_mpg" "curb_weight"
[4] "engine_size"           "engine_type_ohc" "horsepower"
[7] "length"                "num_cylinders"  "peak_rpm"
```

⁵By default, variable selection is performed individually on dummy variables for a factor. The variable selection procedures return the permutation distribution and an aggregation of the dummy variables’ inclusion proportions can allow for variable selection to be performed on an entire factor.

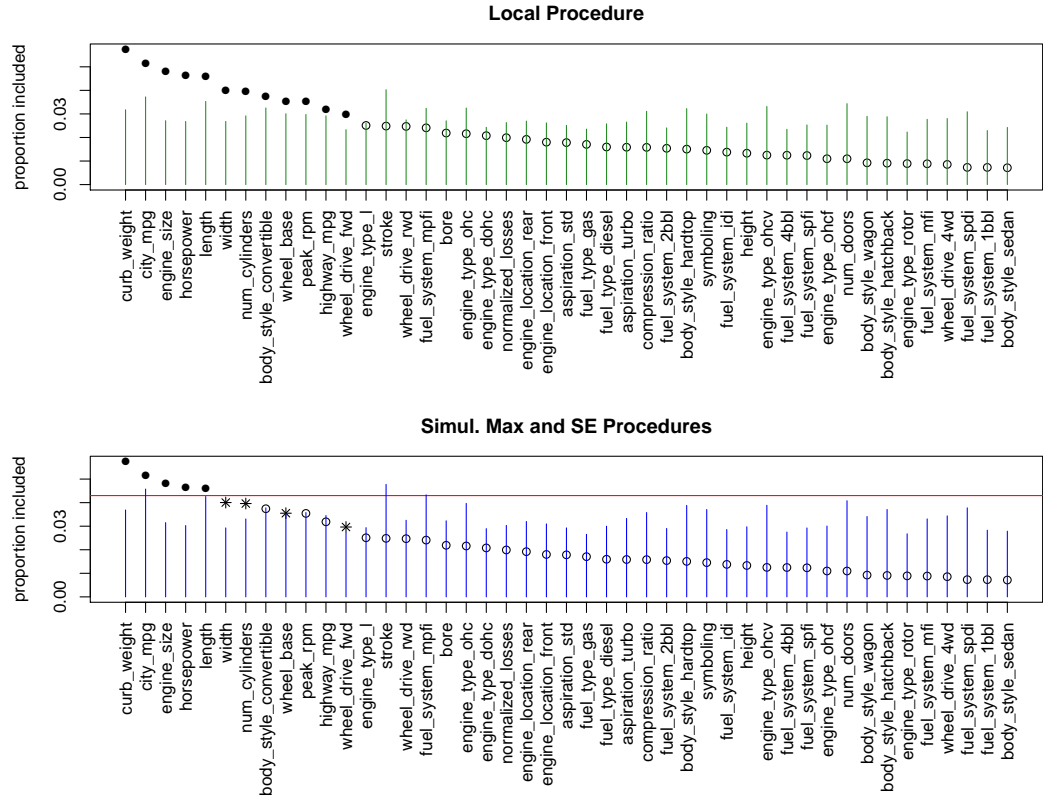


Figure 2.9: Visualization of the three variable selection procedures developed in Chapter 3 with $\alpha = 0.05$. The top plot illustrates the “Local” procedure. The green lines are the threshold levels determined from the permutation distributions that must be exceeded for a variable to be selected. The plotted points are the variable inclusion proportions for the observed data (averaged over five duplicate `bartMachine` models). If the observed value is higher than the green bar, the variable is included and is displayed as a solid dot; if not, it is not included and it is displayed as an open dot. The bottom plot illustrates both the “Global SE” and “Global Max” thresholds. The red line is the cutoff for “Global Max” and variables pass this threshold are displayed as solid dots. The blue lines represent the thresholds for the “Global SE” procedure. Variables that exceed this cutoff but not the “Global Max” threshold are displayed as asterisks. Open dots exceed neither threshold.

```
[10] "wheel_base"           "wheel_drive_fwd"      "wheel_drive_rwd"
[13] "width"
```

On this dataset, the “best” approach (as defined by out-of-sample prediction error) is the “Local” procedure.

2.3.10 Informed prior information on covariates

Chapter 3 also introduces a method for incorporating informed prior information about the predictors into the **BART** model. This can be achieved by modifying the prior on the splitting rules as well as the corresponding calculations in the Metropolis-Hastings step. In particular, covariates believed to influence the response can a priori be proposed more often as candidates for splitting rules. Useful prior information can aid in both variable selection and prediction tasks. We demonstrate the impact of a correctly informed prior in the context of the Friedman (1991) function (Equation 2.3).

$$\mathbf{y} = 10\sin(\pi\mathbf{x}_1\mathbf{x}_2) + 20(\mathbf{x}_3 - .5)^2 + 10\mathbf{x}_4 + 5\mathbf{x}_5 + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}_n(\mathbf{0}, \sigma^2\mathbf{I}). \quad (2.3)$$

To illustrate, we construct a design matrix \mathbf{X} where the first five columns are predictors which influence the response ($\mathbf{x}_1, \dots, \mathbf{x}_5$ in Equation 2.3) and the next 95 columns are predictors that do not affect the response.

All that is required is a specification of relative weights for each predictor. These weights are normalized internally to become probabilities. As an example, we assign 5 times the weight to the 5 true covariates of the model relative to the 95 useless covariates.

```
R> prior <- c(rep(5, times = 5), rep(1, times = 95))
```

We now sample 500 observations from the Friedman function and construct a default `bartMachine` model as well as a `bartMachine` model with the informed prior and compare their performance using the RMSE metric on a test set of another 500 observations.

```
R> bart_machine <- bartMachine(X, y)
R> bart_machine_informed <- bartMachine(X, y, cov_prior_vec = prior)

R> bart_predict_for_test_data(bart_machine, Xtest, ytest)$rmse
[1] 1.661159
R> bart_predict_for_test_data(bart_machine_informed,
  Xtest, ytest)$rmse
[1] 1.232925
```

There is a substantial improvement in out-of-sample predictive performance when a properly informed prior is used.

Note that by default the prior vector down-weights the indicator variables that result from dummifying categorical variables so that the total set of dummy variables has the same weight as a continuous covariate.

2.3.11 Interaction effect detection

In Section 2.3.5, we explored using variable inclusion proportions to understand the relative influences of different covariates. A similar procedure can be carried out for examining interaction effects within a BART model. This question was initially explored in Damien et al. (2013) where an interaction was considered to exist between two variables if they both appeared in at least one splitting rule in a given tree. We refine the definition of an interaction as follows.

We first begin with a $p \times p$ matrix of zeroes. Within a given tree, for each split rule variable j , we look at all split rule variables of child nodes, k , and we increment the j, k element of the matrix. Hence variables are considered to interact in a given tree *only if* they appear together in a contiguous downward path from the root node to a terminal node. Note that a variable may interact with itself (when fitting a linear effect, for instance). Since there is no order between the parent and child, we then add the j, k counts together with the k, j counts (if $j \neq k$). Summing across trees and MCMC iterations gives the total number of interactions for each pair of variables from which relative importance can be assessed.

We demonstrate interaction detection on the Friedman function using 10 covariates using the code below:

```
R> interaction_investigator(bart_machine, num_replicates_for_avg = 25,
  num_var_plot = 10, bottom_margin = 5)
```

Figure 2.10 shows the ten most important interactions in the model. The illustration is averaged over 25 model constructions to obtain stable estimates across many posterior modes in the sum-of-trees distribution. Notice that the interaction between \mathbf{x}_1 and \mathbf{x}_2 dominates all other interaction terms, as `bartMachine` is correctly capturing the single true interaction effect: the $\sin(\pi \mathbf{x}_1 \mathbf{x}_2)$ term in Equation 2.3. Choosing which of these interactions *significantly* affect the response is not addressed in this paper. The methods proposed in Chapter 3 may be applicable here and we consider this to be fruitful future work.

2.3.12 `bartMachine` Model Persistence Across R Sessions

A convenient feature of `bartMachine` is its ability to “serialize” the model. Serialization allows the user to construct models and have them persist across R sessions. The

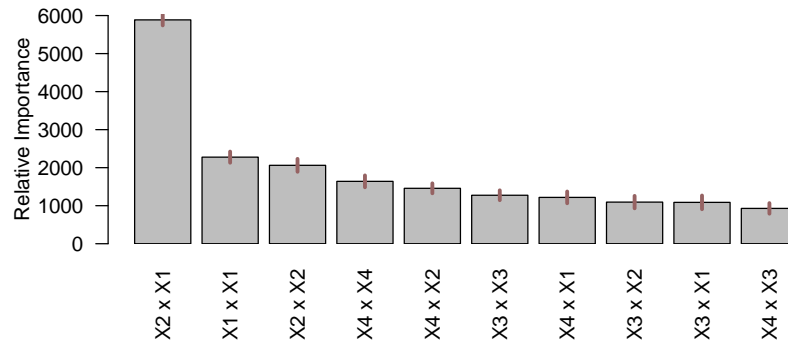


Figure 2.10: The top 10 average variable interaction counts (termed “relative importance”) in the default `bartMachine` model for the Friedman function data averaged over 25 model constructions. The segments atop the bars represent 95% confidence intervals.

cost is time during model creation and hard drive space. Thus, the `serialize` feature is defaulted to “off.” We demonstrate using the code below:

```
R> bart_machine <- bartMachine(X, y, serialize = TRUE)
R> save.image("bart_demo.RData")
R> q("no")
> R
R> options(java.parameters = "-Xmx2500m")
R> library(bartMachine)
R> load("bart_demo.RData")
R> predict(bart_machine, X[1 : 4, ])
[1] 20.0954617 14.8860727 10.9483889 11.4350277
```

The training data is the same as in the previous section: $n = 100$ and $p = 10$. For the default `bartMachine` settings, $m = 50$, number of burn-in MCMC iterations

is 250 and number of posterior samples is 1000. These settings yielded an almost instant serialization and a 2.1MB RData image file. For a more realistic dataset with $n = 5000$, $p = 1000$, $m = 100$ and 5000 posterior samples, the serialization and saving of the file took a few minutes and required 100MB.

Note that the package throws an error if the user attempts to make use of a `bartMachine` object in another session which was not serialized:

```
R> bart_machine <- bartMachine(X, y) #serialize is FALSE here
R> save.image("bart_demo.RData")
R> q("no")
> R

R> options(java.parameters = "-Xmx2500m")
R> library(bartMachine)
R> load("bart_demo.RData")
R> predict(bart_machine, X[1 : 4, ])
Error in check_serialization(object) :
  This bartMachine object was loaded
from an R image but was not serialized.
Please build bartMachine using the
option "serialize = TRUE" next time.
```

2.4 bartMachine Package Features for Classification

In this section we highlight the features that differ in `bartMachine` when the response is continuous (in the regression setting) versus when the response is dichotomous categorical variable (in the classification setting). The illustrative dataset consists of 332 Pima Indians obtained from the UCI repository. Of the 332 subjects, 109 were

diagnosed with diabetes, the binary response variable. There are seven continuous predictors which are body metrics such as blood pressure, glucose concentration, etc. and there is no missing data.

Building a `bartMachine` model for classification has the same computing parameters except that q, ν cannot be specified since there is no longer a prior on σ^2 (see Section 1.4.3). We first build a cross-validated model below.

```
R> library(MASS)
R> data(Pima.te)
R> X <- data.frame(Pima.te[, -8])
R> y <- Pima.te[, 8]
R> bart_machine_cv <- bartMachineCV(X, y)
... bartMachine CV win: k: 3 m: 50
R> bart_machine_cv
bartMachine v1.1.1 for classification

training data n = 332 and p = 7
built in 0.7 secs on 4 cores, 50 trees, 250 burn-in
and 1000 post. samples
```

confusion matrix:

	predicted No	predicted Yes	model errors
actual No	210.000	13.00	0.058
actual Yes	41.000	68.00	0.376
use errors	0.163	0.16	0.163

Classification models have an added hyperparameter, `prob_rule_class`, which is the rule for determining if the probability estimate is great enough to be classified into the positive category. We can see above that the `bartMachine` at times predicts “NO” for true “YES” outcomes and we suffer from a 37.6% error rate for this outcome. We can try to mitigate this error by lowering the threshold to increase the number of “YES” labels predicted:

```
R> bartMachine(X, y, prob_rule_class = 0.3)
```

```
bartMachine v1.1.1 for classification
```

```
training data n = 332 and p = 7
```

```
built in 0.6 secs on 4 cores, 50 trees, 250 burn-in  
and 1000 post. samples
```

```
confusion matrix:
```

	predicted No	predicted Yes	model errors
actual No	177.000	46.000	0.206
actual Yes	11.000	98.000	0.101
use errors	0.059	0.319	0.172

This lowers the model error to 10% for the “YES” class, but at the expense of increasing the error rate for the “NO” class. We encourage the user to cross-validate this rule based on an appropriate objective function for the problem at hand.

We can also check out-of-sample statistics:

```
R> oos_stats <- k_fold_cv(X, y, k_folds = 10)
```



```
R> oos_stats$confusion_matrix
```

	predicted No	predicted Yes	model errors
actual No	200.00	23.000	0.103
actual Yes	47.00	62.000	0.431
use errors	0.19	0.271	0.211

Note that it is possible to predict both class labels and probability estimates for given observations:

```
R> predict(bart_machine_cv, X[1 : 2, ], type = "prob")
```

```
[1] 0.6253160 0.1055975
```

```
R> predict(bart_machine_cv, X[1 : 2, ], type = "class")
```

```
[1] Yes No
```

```
Levels: No Yes
```

When using the covariate tests of Section 2.3.6, total misclassification error becomes the statistic of interest instead of Pseudo- R^2 . The p value is calculated now as the proportion of null samples with *lower* misclassification error. Figure 2.11 illustrates the test showing that predictor **age** seems to matter in the prediction of **Diabetes**, controlling for other predictors.

The partial dependence plots of Section 2.3.7 are now scaled as probit of the probability estimate. Figure 2.12 illustrates that as glucose increases, the probability of contracting **Diabetes** increases linearly on a probit scale.

Credible intervals are implemented for classification **bartMachine** and are displayed on the probit scale. Note that the prediction intervals of Section 2.3.4 do not exist for classification since \mathcal{E} noise is not part of the probit model.

```
R> calc_credible_intervals(bart_machine_cv, X[1 : 2, ])
```

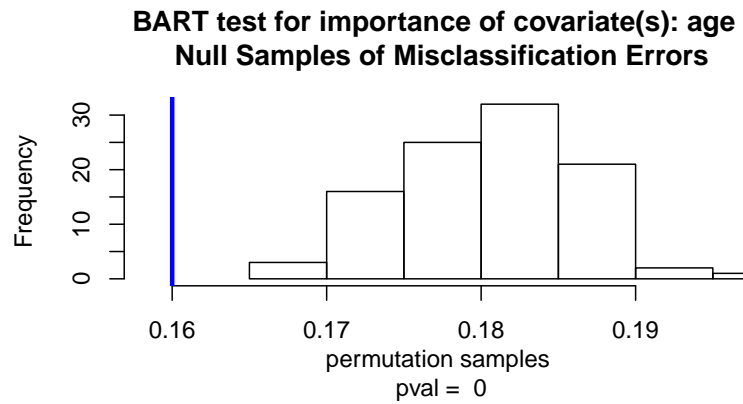


Figure 2.11: Test of covariate importance for predictor **age** on whether or not the subject will contract **Diabetes**.

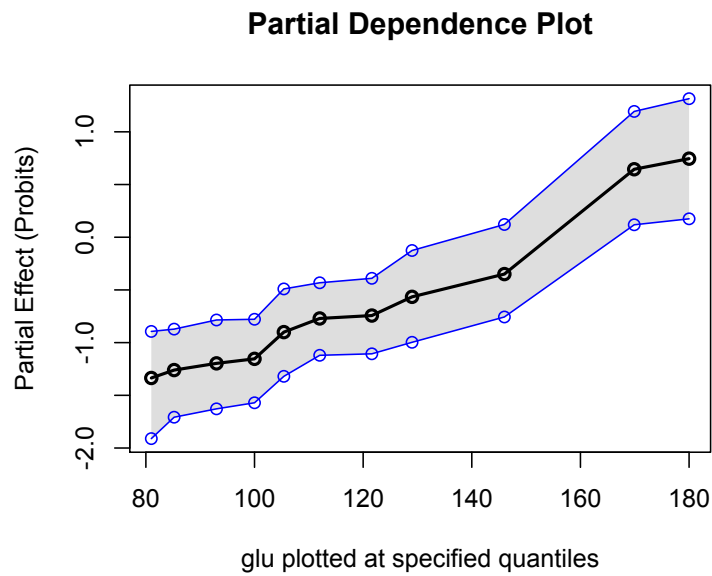


Figure 2.12: PDP for predictor **glu**. The blue lines are 95% credible intervals.

```

      ci_lower_bd ci_upper_bd
[1,]  0.34865355  0.8406097
[2,]  0.01686486  0.2673171

```

Other functions work similarly to regression except those that plot the responses and those that explicitly depend on RMSE as an error metric.

2.5 Conclusion

2.5.1 Forecasting “Bakeoff”

Finally, we compare the performance of `bartMachine` to `BayesTree` (and `RF`). We bake-off nine regression data sets and assessed out-of-fold RMSE using 10-fold cross-validation. We average the results across 20 replications of cross-validation. The results are displayed in Table 2.2.

	<code>bartMachine</code>	<code>BayesTree</code>	<code>RF</code>
boston	4.451	4.503	4.582
triazine	0.128*	0.130	0.119
ozone	4.147	4.144	4.064
baseball	709.197	709.437	729.188
wine.red	0.656	0.651*	0.642
ankara	1.348*	1.461	1.574
wine.white	0.759*	0.766	0.746
pole	11.713*	12.755	10.691
compactiv	3.262	2.795*	2.957

Table 2.2: RMSE values for three machine learning algorithms averaged across 20 replicates. Asterisks indicate a significant difference between `bartMachine` and `BayesTree` at a significance level of 5% with a Bonferroni correction. Comparisons with `randomForest`’s performance were not conducted.

We conclude that the implementation outlined in this paper performs approximately the same as the previous implementation with regards to predictive accuracy.

Table 2.3 shows the average run-time for each algorithm. Note that `bartMachine` is run using 4 cores.

	<code>bartMachine</code>	<code>BayesTree</code>	RF
boston	7.8	28.5	5.1
triazine	5.7	10.7	2.6
ozone	4.7	17.6	2.1
baseball	5.6	18.6	3.3
wine.red	13.5	51.1	10.6
ankara	12.8	27.0	10.9
wine.white	13.5	56.0	11.0
pole	18.2	7.0	12.0
compactiv	16.3	18.4	19.2

Table 2.3: Average run-times (in seconds) for each complete k-fold estimation for three machine learning algorithms.

2.5.2 Discussion

This article introduced `bartMachine`, a new R package which implements Bayesian additive regression trees. The goal of this package is to provide a fast, extensive and user-friendly implementation accessible to a wide range of data analysts, and increase the visibility of `BART` to a broader statistical audience. We hope we have provided organized, well-documented open-source code and we encourage the community to make innovations on this package.

Citation

Kaplener, A. and Bleich, J. (2015). `bartMachine`: Machine learning with Bayesian Additive Regression Trees. *Journal of Statistical Software*, *forthcoming*.

Variable Selection for BART

Abstract

We consider the task of discovering gene regulatory networks, which are defined as sets of genes and the corresponding transcription factors which regulate their expression levels. This can be viewed as a variable selection problem, potentially with high dimensionality. Variable selection is especially challenging in high dimensional settings, where it is difficult to detect subtle individual effects and interactions between predictors. **BART** provides a novel nonparametric alternative to parametric regression approaches, such as the lasso or stepwise regression, especially when the number of relevant predictors is sparse relative to the total number of available predictors and the fundamental relationships are nonlinear. We develop a principled permutation-based inferential approach for determining when the effect of a selected predictor is likely to be real. Going further, we adapt the **BART** procedure to incorporate informed prior information about variable importance. We present simulations demonstrating that our method compares favorably to existing parametric and nonparametric procedures in a variety of data settings. To demonstrate the potential of our approach in a biological context, we apply it to the task of inferring the gene regulatory network in yeast (*Saccharomyces cerevisiae*). We find that our **BART**-based procedure is best able

to recover the subset of covariates with the largest signal compared to other variable selection methods. The methods developed in this work are readily available in the R package `bartMachine`.

3.1 Introduction

An important statistical problem in many application areas is variable selection: identifying the subset of covariates that exert influence on a response variable. We consider the general framework where we have a continuous response variable \mathbf{y} and a large set of predictor variables $\mathbf{x}_1, \dots, \mathbf{x}_K$. We focus on variable selection in the sparse setting: only a relatively small subset of those predictor variables truly influences the response variable.

One such example of a sparse setting is the motivating application for this paper: inferring the gene regulatory network in budding yeast (*Saccharomyces cerevisiae*). In this application, we have a collection of approximately 40 transcription factor proteins (TFs) that act to regulate cellular processes in yeast by promoting or repressing transcription of specific genes. It is unknown which of the genes in our yeast data are regulated by each of the transcription factors. Therefore, the goal of the analysis is to discover the corresponding network of gene-TF relationships, which is known as a *gene regulatory network*. Each gene, however, is regulated by only a small subset of the TFs which makes this application a sparse setting for variable selection. The available data consist of gene expression measures for approximately 6000 genes in yeast across several hundred experiments, as well as expression measures for each of the approximately 40 transcription factors in those experiments (Jensen et al., 2007).

This gene regulatory network was previously studied in Jensen et al. (2007) with a focus on modeling the relationship between genes and transcription factors. The

authors considered a Bayesian linear hierarchical model with first-order interactions. In high-dimensional data sets, specifying even first-order pairwise interactions can substantially increase the complexity of the model. Additionally, given the elaborate nature of biological processes, there may be interest in exploring nonlinear relationships as well as higher-order interaction terms. In such cases, it may not be possible for the researcher to specify these terms in a linear model *a priori*. Indeed, Jensen et al. (2007) acknowledge the potential utility of such additions, but highlight the practical difficulties associated with the size of the resulting parameter space. Thus, we propose a variable selection procedure that relies on **BART**. **BART** dynamically estimates a model from the data, thereby allowing the researcher to potentially identify genetic regulatory networks without the need to specify higher order interaction terms or nonlinearities ahead of time.

Additionally, we have data from chromatin immunoprecipitation (ChIP) binding experiments (Lee et al., 2002). Such experiments use antibodies to isolate specific DNA sequences which are bound by a TF. This information can be used to discover potential binding locations for particular transcription factors within the genome. The ChIP data can be considered “prior information” that one may wish to make use of when investigating gene regulatory networks. Given the Bayesian nature of our approach, we propose a straightforward modification to **BART** which incorporates such prior information into our variable selection procedure.

In Section 3.2, we review some common techniques for variable selection. We emphasize the limitations of approaches relying on linear models and highlight variable selection via tree-based techniques. Section 3.3 focuses on modifying **BART** for variable selection. In Sections 3.3.1 and 3.3.2, we introduce how **BART** computes variable inclusion proportions and explore the properties of these proportions. In Section 3.3.3, we develop procedures for principled variable selection based upon **BART** output. In

Section 3.3.4 we extend the **BART** procedure to incorporate prior information about predictor variable importance. In Section 3.4, we compare our methodology to alternative variable selection approaches in both linear and nonlinear simulated data settings. In Section 3.5, we apply our **BART**-based variable selection procedure to the discovery of gene regulatory networks in budding yeast. Section 3.6 concludes with a brief discussion. As noted in Chapter 2, our variable selection procedures as well as the ability to incorporate informed prior information are readily available features in the **bartMachine** package. Code demonstrations are shown in Sections 2.3.9-2.3.10.

3.2 Techniques for Variable Selection

3.2.1 Linear Methods

The variable selection problem has been well-studied from both the classical and Bayesian perspective, though most previous work focuses on the case where the outcome variable is assumed to be a linear function of the available covariates. Stepwise regression (Hocking, 1976) using criteria such as the AIC or BIC is a common approach for variable selection from a large set of possible predictor variables. Best subsets regression (Miller, 2002) can also be employed, although this option becomes too computationally burdensome as K becomes large. Other popular linear variable selection methods are lasso regression (Tibshirani, 1996) and the elastic net (Zou and Hastie, 2005). Both of these approaches enforce sparsity on the subset of selected covariates by imposing penalties on non-zero coefficients. Park and Casella (2008) and Hans (2009) provide Bayesian treatments of lasso regression.

Perhaps the most popular Bayesian variable selection strategies are based on linear regression with a “spike-and-slab” prior distribution on the regression coefficients.

Initially proposed by Mitchell and Beauchamp (1988), who used a mixture prior of a point mass at zero and a uniform slab, George and McCulloch (1993) went on to use a mixture-of-normals prior, for which Markov chain Monte Carlo stochastic search of the posterior could be easily implemented. Eventually, most applications gravitated towards a limiting form of the normal mixture with a degenerate point mass at zero. More recent work involving spike-and-slab models has been developed in Ishwaran and Rao (2005), Li and Zhang (2010), Hans et al. (2007), Bottolo and Richardson (2010), Stingo and Vannucci (2011) and Rockova and George (2013). In these approaches, variable selection is based on the posterior probability that each predictor variable is in the slab distribution, and sparsity can be enforced by employing a prior that strongly favors the spike distribution at zero.

3.2.2 Tree-Based Methods

Each of the aforementioned approaches assumes that the response variable is a linear function of the predictor variables. A major drawback of linear models, both in the frequentist and Bayesian paradigms, is that they are ill-equipped to handle complex, nonlinear relationships between the predictors and response. Nonlinearities and interactions, which are seldom known with certainty, must be specified in advance by the researcher. In the case where the model is mis-specified, incorrect variables may be included and correct variables excluded.

As an alternative, we consider nonparametric methods which are flexible enough to fit a wide array of functional forms. We focus in particular on the ensemble-of-trees algorithms. Compared with linear models, these procedures are better able to approximate complicated response surfaces but are “black-boxes” in the sense that they offer less insight into how specific predictor variables relate to the response

variable.

Tree-based variable selection makes use of the internals of the decision tree structure. In particular, existing tree-based methods for variable selection focus on the set of splitting variables within the trees. For example, Gramacy et al. (2013) develop a backward stepwise variable selection procedure for dynamic trees (DT) by considering the average reduction in posterior predictive uncertainty within all nodes that use a particular predictor as the splitting variable. Also, the splitting variables in RF can also be used to develop variable selection approaches. For instance, one can consider the reduction in sum of square errors (node impurity in classification problems) associated with a particular predictor. Additionally, Díaz-Uriarte and Alvarez de Andrés (2006) consider reduction in out-of-bag mean square error associated with each predictor to develop a backward stepwise selection procedure.

We too consider the splitting variables for BART in developing our method, but our approach differs from the previously mentioned work in two aspects. First, we do not propose a backwards stepwise selection, but rather develop a permutation-based inferential approach. Second, we do not consider the overall improvement to fit provided by each predictor variable, but instead consider how often a particular predictor appears in a BART model. While simple, this metric shows promising performance for variable selection using BART.

3.3 Calibrating BART Output for Variable Selection

Recalling the BART model

$$\mathbf{Y} = f(\mathbf{X}) + \boldsymbol{\varepsilon} \approx \mathfrak{F}_1^{\text{leaf}}(\mathbf{X}) + \mathfrak{F}_2^{\text{leaf}}(\mathbf{X}) + \dots + \mathfrak{F}_m^{\text{leaf}}(\mathbf{X}) + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}_n(\mathbf{0}, \sigma^2 \mathbf{I}_n), \quad (3.1)$$

we first draw attention to the hyperparameter m , the number of trees in the ensemble. In Chipman et al. (2010), m is generally pre-specified by the researcher. The usual goal of BART is predictive performance, in which case a large value of m allows for increased flexibility when fitting a complicated response surface, thereby improving predictive performance. However, Chipman et al. (2010) recommend using a smaller value of m for the purposes of variable selection (we default to $m = 20$). When the number of trees in the ensemble is smaller, there are fewer opportunities for predictor variables to appear in the model and so they must compete with each other to be included. However, if m is too small, the Gibbs sampler in BART becomes trapped in local modes of the posterior distribution more often which can destabilize the results of the estimation procedure (Chipman et al., 1998). Also, there is not enough flexibility in the model to fit a variety of complicated functions. However, when the number of trees becomes too large, there is opportunity for unimportant variables to enter the model without impacting the overall model fit, thereby making variable selection more challenging.

Our explorations have shown that $m = 20$ represents a good compromise, although similar choices of m should not impact results. Under the sparse data settings we will examine in Sections 3.4 and 3.5, we show that this medium level of m aids the selection of important predictor variables even when the number of predictor variables is relatively large.

It is also worth recalling that in the default BART formulation, each predictor variable \mathbf{x}_k has an equal *a priori* chance of being chosen as a splitting variable for each tree in the ensemble. However, in many applications, we may have real prior information that suggests the importance of particular predictor variables. In Section 3.3.4, we will extend the BART procedure to incorporate prior information about specific predictor variables, which will be used to aid in discovering the yeast gene regulatory

network in Section 3.5.

3.3.1 BART Variable Inclusion Proportions

The primary output from BART is a set of predicted values $\hat{\mathbf{y}}$ for the response variable \mathbf{y} . Although these predicted values $\hat{\mathbf{y}}$ serve to describe the overall fit of the model, they are not directly useful for evaluating the relative importance of each predictor variable in order to select a subset of predictor variables. For this purpose, Chipman et al. (2010) begin exploring the “variable inclusion proportions” of each predictor variable. We extend their exploration into a principled method.

Across all m trees in the ensemble (Equation 3.1), we examine the set of predictor variables used for each splitting rule in each tree. Within each posterior Gibbs sample, we can compute the proportion of times that a split using \mathbf{x}_k as a splitting variable appears among all splitting variables in the ensemble. Since the output of BART consists of many posterior samples, we estimate the *variable inclusion proportion* p_k as the posterior mean of these proportions across all of the posterior samples.

Intuitively, a large variable inclusion proportion p_k is suggestive of a predictor variable \mathbf{x}_k being an important driver of the response. Chipman et al. (2010) suggest using $\mathbf{p} = (p_1, \dots, p_K)$ to rank variables $\mathbf{x}_1, \dots, \mathbf{x}_K$ in terms of relative importance. These variable inclusion proportions naturally build in some amount of multiplicity control since the p_k ’s have a fixed budget (in that they must sum to one) and that budget will become more restrictive as the number of predictor variables increases.

However, each variable inclusion proportion p_k cannot be interpreted as a posterior probability that the predictor variable \mathbf{x}_k has a “real effect”, defined as the impact of some linear or nonlinear association, on the response variable. This motivates the primary question being addressed by this chapter: *how large does the variable inclu-*

sion proportion p_k have to be in order to select predictor variable \mathbf{x}_k as an important variable?

As a preliminary study, we evaluate the behavior of the variable inclusion proportions in a “null” data setting, where we have a set of K predictor variables \mathbf{x}_k that are all unrelated to the outcome variable \mathbf{y} . Specifically, we generate each response variable y_i and each predictor variable x_{ik} independently from a standard normal distribution. In this null setting, one might expect that BART would choose amongst the predictor variables uniformly at random when adding variables to the ensemble of trees (Equation 3.1). In this scenario, each variable inclusion proportion would then be close to the inverse of the number of predictor variables, i.e. $p_k \approx 1/K$ for all k .

However, we have found empirically that in this scenario the variable inclusion proportions do not approach $1/K$ for all predictor variables. As an example, Figure 3.1 gives the variable inclusion proportions from a null simulation with $n = 250$ observations and $K = 40$ predictor variables, all of which are unrelated to the response variable \mathbf{y} .

In this setting, the variable inclusion proportions do not converge to $1/40 = .025$. As seen in Figure 3.1, some variable inclusion proportions remain substantially larger than $1/K$ and some are substantially smaller. We observed this same phenomenon with different levels of noise in the response variable.

3.3.2 Further Exploration of Null Simulation

We hypothesize that the variation between p_k ’s in Figure 3.1 can stem from two causes. First, even though the response and predictors were generated independently, they will still exhibit some random association. BART may be fitting noise, or “chance-capitalizing;” given its nonparametric flexibility, BART could be fitting to perceived

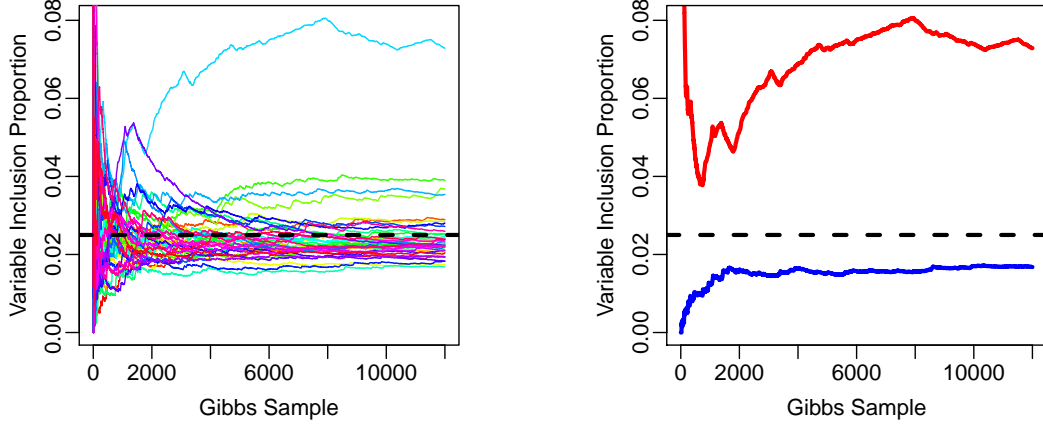


Figure 3.1: Variable inclusion proportions from **BART** model in null setting where each predictor variable is unrelated to the response variable. **Left:** Variable inclusion proportions for all $K = 40$ predictor variables over 12,000 Gibbs samples. **Right:** Tracking of the maximum and minimum of the variable inclusion proportions.

nonlinear associations that are actually just noise. Second, there might be inherent variation in the **BART** estimation procedure itself, possibly due to the Gibbs sampler getting stuck in a local maximum.

Thus, we consider an experiment to explore the source of this variation among the p_k 's. We generate 100 data sets under the same setting as that in Figure 3.1. Within each data set, we run **BART** 50 times with different initial values for the model parameters randomly drawn from the respective prior distributions. Let p_{ijk} denote the variable inclusion proportion for the i^{th} data set, j^{th} **BART** run, and the k^{th} predictor variable. We then consider the decomposition into three nested variances listed in Table 3.1. Note that we use standard deviations in our illustration that follows.

First consider what may happen if the source of Figure 3.1's observed pathology is purely due to **BART**'s Gibbs sampler getting stuck in different local posterior modes. On the first run for the first data set, **BART** would fall into a local mode where some

$s_{ik}^2 = \frac{1}{50} \sum_{j=1}^{50} (p_{ijk} - \bar{p}_{i.k})^2$	The variability of BART estimation for a particular predictor k in a particular data set i .
$s_k^2 = \frac{1}{100} \sum_{i=1}^{100} (\bar{p}_{i.k} - \bar{p}_{..k})^2$	The variability due to chance capitalization of the BART procedure for predictor k across data sets.
$s^2 = \frac{1}{40} \sum_{k=1}^{40} (\bar{p}_{..k} - \bar{p}_{...})^2$	The variability across predictors.

Table 3.1: The three nested variances.

predictors are naturally more important than others and hence the p_{11k} 's would be unequal. In the same data set, second run, **BART** might fall into a different local mode where the p_{12k} 's are unequal, but in a way that is different from the first run's p_{11k} 's. This type of process would occur over all 50 runs. Thus, the s_{1k} , the standard deviation of p_{ijk} over runs of **BART** on the first data set, would be large. Note that if there is no chance capitalization or overfitting, there should be no reason that averages of the proportions, the $\bar{p}_{1.k}$'s, should be different from $1/K$ over repeated runs. Then, when the second data set is introduced, **BART** will continue to get stuck in different local posterior modes and the s_{2k} 's should be large, but the $\bar{p}_{2.k}$'s should be near $1/K$. Hence, over all of the data sets, $\bar{p}_{i.k}$'s should be approximately $1/K$, implying that the s_k 's should be small. In sum, **BART** getting stuck in local modes suggests large s_{ik} 's and small s_k 's.

Next consider what may happen if the source of Figure 3.1's observed pathology is purely due to **BART** chance-capitalizing on noise. On the first data set, over each run, **BART** does not get stuck in local modes, and therefore the p_{i1k} 's across runs would be fairly stable. Hence, the s_{1k} 's would be small. However, in each of the runs, **BART** overfits in the *same way* for each data set. For example, perhaps **BART** perceives an association between x_1 and y on the first data set. Hence, the p_{1j1} 's would be larger than $1/K$ on all restarts (**BART** would select x_1 as a splitting rule often due to the

perceived association) and thus $\bar{p}_{1.1} > 1/K$. Then, in the second data set, **BART** may perceive an association between x_3 and y , resulting in p_{2j3} 's being larger on all runs ($\bar{p}_{2.3} > 1/K$). Thus, **BART** overfitting is indicated by small s_{ik} 's and large s_k 's.

Figure 3.2 illustrates the results of the simulations. Both sources of variation appear but for all predictors, the average s_{ik} is significantly smaller than the s_k . This finding suggests that within a particular data set, **BART** is chance-capitalizing and overfitting to the noise, which prevents the p_k 's from converging to $1/K$.⁶

Also note the overall average inclusion proportion $\bar{p}_{..}$ is $.025 = 1/K$, so across data sets and **BART** runs the variable inclusion proportions are correct on average. Further, the standard deviation across predictors s is small. This implies that the $\bar{p}_{..k}$'s are approximately $1/K$ as well, which indicates there is no systematic favoring of different covariates once the effect of overfitting by data set and remaining in local modes by run is averaged out.

We believe this experiment demonstrates that there is a large degree of chance capitalization present in the variable inclusion proportions in the “null” model. This implies that it is not possible to decide on an appropriate threshold for the p_k 's when selecting a subset of important predictor variables in real data settings. Further, the chance capitalization is idiosyncratic for any data set, making it challenging to pose a simple parametric model for the behavior in Figure 3.1 that would be useful in practice. This motivates our nonparametric approach to establishing thresholds for the variable inclusion proportions based on permutations of the response variable \mathbf{y} .

As noted above, there is some variability in the p_k 's between **BART** runs from different starting points. We found that averaging over results from five repetitions

⁶We also considered this experiment with orthogonalized predictors (not shown). This reduces the s_k 's (chance capitalization) in Figure 3.2 slightly, but the s_k 's are still larger than the average s_{ik} 's. Hence, even if there is no *linear* correlation between the predictors and the response, **BART** is capitalizing on nonlinear associations.

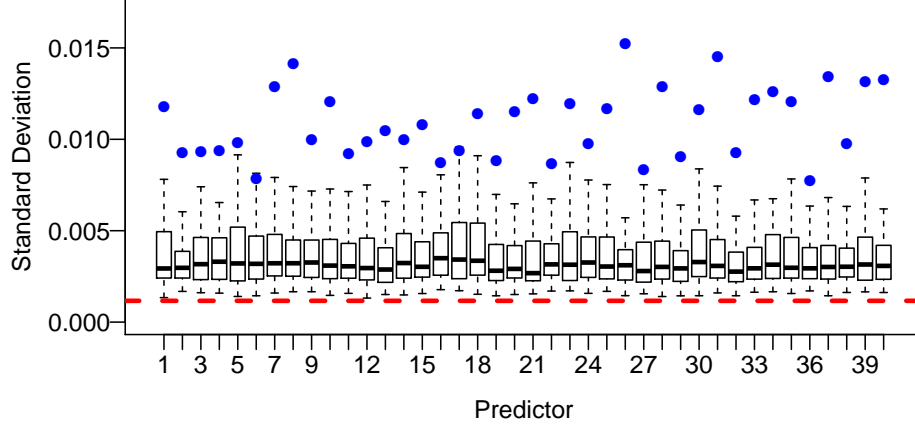


Figure 3.2: The boxplots represent the distribution of s_{ik} for each predictor. The circles represent the values of s_k and the dashed line corresponds to s . Note that the results are reported as standard deviations and points in the boxplots beyond the whiskers are omitted.

of the BART algorithm from different starting points was sufficient to provide stable estimates of the variable inclusion proportions and use these averaged values as our variable inclusion proportions for the remainder of the chapter.

3.3.3 Variable Inclusion Proportions under Permuted Responses

We now address our key question: *how large does the variable inclusion frequency p_k have to be in order to select predictor variable \mathbf{x}_k ?* To determine an appropriate selection threshold, we employ a permutation-based approach to generate a null distribution for the variable inclusion proportions $\mathbf{p} = (p_1, \dots, p_K)$.

Specifically, we create P permutations of the response vector: $\mathbf{y}_1^*, \mathbf{y}_2^*, \dots, \mathbf{y}_P^*$. For each of these permuted response vectors \mathbf{y}_p^* , we run the BART model using \mathbf{y}_p^* as the response and the original $\mathbf{x}_1, \dots, \mathbf{x}_K$ as predictor variables. This permutation strategy preserves possible dependencies among the predictor variables while removing any

dependency between the predictor variables and the response variable.

We retain the variable inclusion proportions estimated from the BART run using each permuted response \mathbf{y}_p^* . We use the notation $p_{k,p}^*$ for the variable inclusion proportion from BART for predictor \mathbf{x}_k from the p -th permuted response, and we use the notation \mathbf{p}_p^* for the vector of all variable inclusion proportions from the p -th permuted response. We use the variable inclusion proportions $\mathbf{p}_1^*, \mathbf{p}_2^*, \dots, \mathbf{p}_P^*$ across all P permutations as the null distribution for our variable inclusion proportions \mathbf{p} from the real (unpermuted) response \mathbf{y} .

The remaining issue is selecting an appropriate threshold for predictor \mathbf{x}_k based on the permutation null distribution $\mathbf{p}_1^*, \mathbf{p}_2^*, \dots, \mathbf{p}_P^*$. We will consider three different threshold strategies that vary in terms of the stringency of their resulting variable selection procedure.

The first strategy is a “**local**” threshold: we calculate a threshold for each variable inclusion proportion p_k for each predictor \mathbf{x}_k based only on the permutation null distribution of p_k . Specifically, we take the $1 - \alpha$ quantile of the distribution of $p_{k,1}^*, p_{k,2}^*, \dots, p_{k,P}^*$ and only select predictor \mathbf{x}_k if p_k exceeds this $1 - \alpha$ quantile.

The second strategy is a “**global max**” threshold: we calculate a threshold for the variable inclusion proportion p_k for predictor \mathbf{x}_k based on the maximum across the permutation distributions of the variable inclusion proportions for all predictor variables. Specifically, we first calculate $p_{\max,p}^* = \max \{p_{1,p}^*, p_{2,p}^*, \dots, p_{K,p}^*\}$, the largest variable inclusion proportion across all predictor variables in permutation p . We then calculate the $1 - \alpha$ quantile of the distribution of $p_{\max,1}^*, p_{\max,2}^*, \dots, p_{\max,P}^*$ and only select predictor \mathbf{x}_k if p_k exceeds this $1 - \alpha$ quantile.

The first “local” strategy and the second “global max” strategy are opposite extremes in terms of the stringency of the resulting variable selection. The local strategy is least stringent since the variable inclusion proportion p_k for predictor \mathbf{x}_k needs to

only be extreme within its own permutation distribution in order to be selected. The global maximum strategy is most stringent since the variable inclusion proportion p_k for predictor \mathbf{x}_k must be extreme relative to the permutation distribution across all predictor variables in order to be selected.

We consider a third strategy that is also global across predictor variables, but is less stringent than the global max strategy. The third “**global SE**” strategy uses the mean and standard deviation from the permutation distribution of each variable inclusion proportion p_k to create a global threshold for all predictor variables. Specifically, letting m_k and s_k be the mean and standard deviation of variable inclusion proportion p_k^* for predictor \mathbf{x}_k across all permutations, we calculate

$$C^* = \inf_{C \in \mathbb{R}^+} \left\{ \forall k, \frac{1}{P} \sum_{p=1}^P \mathbb{1}_{p_{k,p}^* \leq m_k + C \cdot s_k} > 1 - \alpha \right\}.$$

The value C^* is the smallest global multiplier that gives simultaneous $1 - \alpha$ coverage across the permutation distributions of p_k for all predictor variables. The predictor \mathbf{x}_k is then only selected if $p_k > m_k + C^* \cdot s_k$. This third strategy is a compromise between the local permutation distribution for variable k (by incorporating each mean m_k and standard deviation s_k) and the global permutation distributions of the other predictor variables (through C^*). We outline all three thresholding procedures in more detail in Appendix A.2.

As an example of these three thresholding strategies, we provide a brief preview of our application to the yeast gene regulatory network in Section 3.5. In that application, the response variable \mathbf{y} consists of the expression measures for a particular gene across approximately 300 conditions and the predictor variables are the expression values for approximately 40 transcription factors in those same 300 conditions.

In Figure 3.3, we give the fifteen predictor variables with the largest variable inclusion proportions from the **BART** model implemented on the data for a particular yeast gene YAL004W. In the top plot, we see the different “local” thresholds for each predictor variable. Four of the predictor variables had variable inclusion proportions p_k that exceeded their local threshold and were selected under this first strategy. In the bottom plot, we see the single “global max” threshold for all predictor variables as well as the different “global SE” thresholds for each predictor variable. Two of the predictor variables had variable inclusion proportions p_k that exceeded their global SE thresholds whereas only one predictor variable exceeded the global max threshold.

This example illustrates that our three threshold strategies can differ substantially in terms of the stringency of the resulting variable selection. Depending on our prior expectations about the sparsity in our predictor variables, we may prefer the high stringency of the global max strategy, the low stringency of the local strategy, or the intermediary global SE strategy.

In practice, it may be difficult to know *a priori* the level of stringency that is desired for a real data application. Thus, we propose a **cross-validation strategy** for deciding between our three thresholding strategies for variable selection. Using k-fold cross validation, the available observations can be partitioned into training and holdout subsets. For each partition, we can implement all three thresholding strategies on the training subset of the data and use the thresholding strategy with the smallest prediction error across the holdout subsets. We call this procedure “**BART-Best**” and provide implementation details in Appendix A.2.

Our permutation-based approach for variable selection does not require any additional assumptions beyond those of the **BART** model. Once again, the sum-of-trees plus normal errors is a flexible assumption that should perform well across a wide range of data settings, especially relative to methods that make stronger parametric

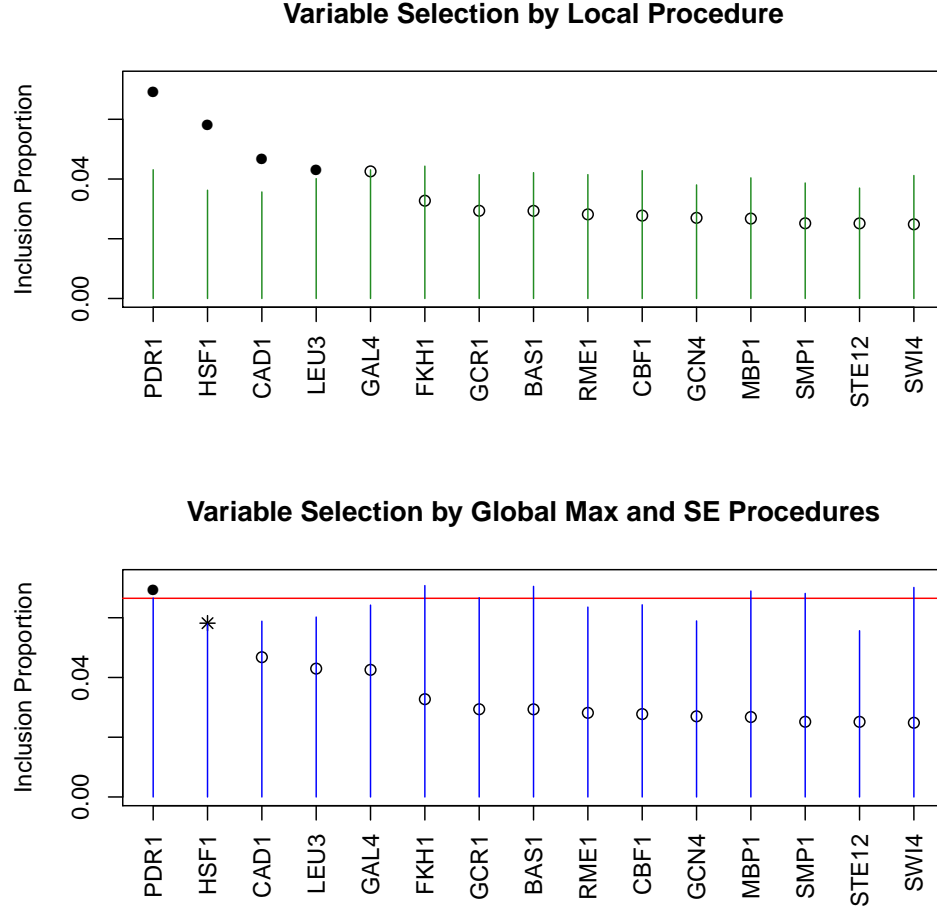


Figure 3.3: The fifteen largest variable inclusion proportions from BART implemented on the yeast gene YAL004W with $\alpha = .05$. **Top:** The tips of the green bands are the “local” thresholds of our first variable selection strategy. Solid dots are selected predictor variables whereas hollow dots are unselected predictor variables. **Bottom:** The red line is the threshold from our second “global max” strategy. The tips of the blue bands are the thresholds from our “global SE” strategy. The one solid dot is the predictor selected by both strategies. The star is the additional predictor variable selected by only the global SE strategy. The hollow dots are unselected predictor variables.

demands. Also, it is important to note that we view each of the strategies described in this section as a procedure for variable selection based on well-founded statistical principles, but do not actually associate any particular formal hypothesis testing with our approach. Finally, a disadvantage of our permutation-based proposal is the computational cost of running BART on a large set of permuted response variables \mathbf{y}^* . However, it should be noted that the permuted response vector runs can be computed in parallel on multiple cores when such resources are available.

3.3.4 Real Prior Information in BART-based Variable Selection

Most variable selection approaches do not allow for *a priori* preferences for particular predictor variables. However, in many applications, there may be available prior information that suggests particular predictor variables may be more valuable than others.

As an example, the yeast regulatory data in Section 3.5 consist of expression measures \mathbf{y}_g for a particular gene g as the response variable with predictor variables \mathbf{x}_k being the expression values for ≈ 40 transcription factors. In addition to the expression data, we also have an accompanying ChIP-binding data set (Lee et al., 2002) that indicates for each gene g which of the ≈ 40 transcription factors are likely to bind near that gene. We can view these ChIP-binding measures as prior probabilities that particular predictor variables \mathbf{x}_k will be important for the response variable \mathbf{y} .

The most natural way to give prior preference to particular variables in BART is to alter the prior on the splitting rules. As mentioned in Section 3.3, by default each predictor \mathbf{x}_k has an equal *a priori* chance of being chosen as a splitting rule for each tree branch in the BART ensemble. We propose altering the prior of the standard BART implementation so that more weight is given to the predictor variables

that have a higher prior probability of being important when randomly selecting a particular predictor variable for a splitting rule. Additionally, the prior on the tree structure, which is needed for the Metropolis-Hastings ratio computation, is appropriately adjusted. This strategy has some precedent as Chipman et al. (1998) discuss non-uniform criteria for splitting rules in the context of an earlier Bayesian Classification and Regression Tree implementation. Note that when employing one of the strategies discussed in Section 3.3.3, the prior is reset to discrete uniform when generating the permutation distribution as it is assumed that there is no relationship between the predictors and the response.

In Section 3.4.3, we present a simulation-based evaluation of the effects on correct variable selection when an informed prior distribution is either correctly specified, giving additional weight to the predictor variables with true influence on the response, or incorrectly specified, giving additional weight to predictor variables that are unrelated to the response. Before our simulation study of the effects of prior information, we first present an extensive simulation study that compares our BART-based variable selection procedure to several other approaches.

3.4 Simulation Evaluation of BART-based Variable Selection

We use a variety of simulated data settings to evaluate the ability of our BART-based procedure to select the subset of predictor variables that have a true influence on a response variable. We examine settings where the response is a linear function of the predictor variables in Section 3.4.1 as well as settings where the response is a nonlinear function of the predictor variables in Section 3.4.2. We also examine the effects of correctly versus incorrectly specified informed prior distributions in Section 3.4.3. For

each simulated data setting, we compare the performance of several different variable selection approaches:

1. **BART-based Variable Selection:** As outlined in Section 3.3, we use the variable inclusion proportions from **BART** to rank and select predictor variables. We evaluate the performance of the three proposed thresholding strategies as well as “BART-Best,” the (five-fold) cross-validation strategy for choosing amongst our thresholding strategies. In each case, we set $\alpha = 0.05$ and the number of trees m is set to 20. Default settings from Chipman et al. (2010) are used for all other hyperparameters. The variable selection procedures are implemented in the R package **bartMachine** (Kapelner and Bleich, 2015).
2. **Stepwise Regression:** Backward stepwise regression using the **stepAIC** function in R.⁷
3. **Lasso Regression:** Regression with a lasso (L1) penalty can be used for variable selection by selecting the subset of variables with nonzero coefficient estimates. For this procedure, an additional penalty parameter λ must be specified, which controls the amount of shrinkage towards zero in the coefficients. We use the **glmnet** package in R (Friedman et al., 2010), which uses ten-fold cross-validation to select the value of the penalty parameter λ .
4. **Random Forests (RF):** Similarly to **BART**, **RF** must be adapted to the task of variable selection.⁸ The **randomForest** package in R (Liaw and Wiener, 2002) produces an “importance score” for each predictor variable: the change in out-of-bag mean square error when that predictor is not allowed to contribute to the

⁷We also considered forward stepwise regression but found that backward stepwise regression performed better in our simulated data settings.

⁸Existing variable selection implementations for **RF** from Díaz-Uriarte and Alvarez de Andrés (2006) and Deng and Runger (2012) are not implemented for regression problems to the best of our knowledge.

model. Breiman and Cutler (2013) suggest selecting only variables where the importance score exceeds the $1 - \alpha$ quantile of a standard normal distribution. We follow their approach and further suggest a new approach: using the Bonferroni-corrected $(1 - \alpha)/p$ quantile of a standard normal distribution. We employ a five-fold cross-validation approach to pick the best of these two thresholding strategies in each simulated data setting and let $\alpha = .05$. Default parameter settings for RF are used.

5. **Dynamic Trees (DT):** Gramacy et al. (2013) introduce a backwards variable selection procedure for DT. For each predictor, the authors compute the average reduction in posterior predictive uncertainty across all nodes using the given predictor as a splitting variable. The authors then propose a relevance probability, which is the proportion of posterior samples in which the reduction in predictive uncertainty is positive. Variables are deselected if their relevance probability does not exceed a certain threshold. After removing variables, the procedure is repeated until the log-Bayes factor of the larger model over the smaller model is positive, suggesting a preference for the larger model. We construct DT using the R package `dynaTree` (Taddy et al., 2011) with 5000 particles and a constant leaf model. We employ the default relevance threshold suggested by the authors of .50.
6. **Spike-and-Slab Regression (Spike-slab):** We employ the spike-and-slab regression procedure outlined in Ishwaran and Rao (2005) and Ishwaran and Rao (2010). The procedure first fits a spike-and-slab regression model and then performs variable selection via the generalized elastic net. Variables with non-zero coefficients are considered relevant. The method is applicable to both high and low dimensional problems, as in the high dimensional setting, a filtering of the variables is first performed for dimension reduction. The procedure is implemented in

the R package `spikeslab` (Ishwaran et al., 2013).

Each of the above methods will be compared on the ability to select “useful” predictor variables, the subset of predictor variables that truly affect the response variable. We can quantify this performance by tabulating the number of true positive (TP) selections, false positive (FP) selections, true negative (TN) selections and false negative (FN) selections. The *precision* of a variable selection method is the proportion of truly useful variables among all predictor variables that are selected,

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \quad (3.2)$$

The *recall* of a variable selection method is the proportion of truly useful variables selected among all truly useful predictor variables,

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \quad (3.3)$$

We can combine the precision and recall together into a single performance criterion,

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \quad (3.4)$$

which is the harmonic mean of precision and recall, balancing a procedure’s capability to make necessary identifications with its ability to avoid including irrelevant predictors. This F_1 *measure* is called the “effectiveness” by van Rijsbergen (1979) and is used routinely in information retrieval and categorization problems.

While many variable selection simulations found in the literature rely on out-of-sample RMSE to assess performance of a procedure, we believe the F_1 score is a better alternative. Out-of-sample RMSE inherently overweights recall vis-à-vis

precision since predictive performance depends more heavily on including covariates which generate signal. This is especially true for adaptive learning algorithms.

We chose the balanced⁹ F_1 metric because we want to demonstrate flexible performance while balancing both recall and precision. For example, if an investigator is searching for harmful physiological agents that can affect health outcomes, identifying the complete set of agents is important (recall). If the investigator is looking to fund new, potentially expensive research based on discoveries (as in our application in Section 3.5) avoiding fruitless directions is most important (precision).

3.4.1 Simulation Setting 1: Linear Relationship

We first examine the performance of the various variable selection approaches in a situation where the response variable is a linear function of the predictor variables. Specifically, we generate each predictor vector \mathbf{x}_j from a normal distribution

$$\mathbf{x}_1, \dots, \mathbf{x}_p \stackrel{iid}{\sim} \mathcal{N}_n(\mathbf{0}, \mathbf{I}), \quad (3.5)$$

and then the response variable \mathbf{y} is generated as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}_n(\mathbf{0}, \sigma^2 \mathbf{I}), \quad (3.6)$$

where $\boldsymbol{\beta} = [\mathbf{1}_{p_0}, \mathbf{0}_{p-p_0}]^\top$. In other words, there are p_0 predictor variables that are truly related to the response \mathbf{y} , and $p - p_0$ predictor variables that are spurious. The *sparsity* of a particular data setting is reflected in the proportion p_0/p of predictor variables that actually influence the response.

Fifty data sets were generated for each possible combination of the following different parameter settings: $p \in \{20, 100, 200, 500, 1000\}$, $p_0/p \in \{0.01, 0.05, 0.1, 0.2\}$

⁹The F_1 measure can be generalized with different weights on precision and recall.

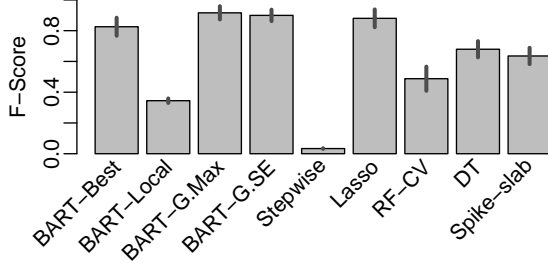
and $\sigma^2 \in \{1, 5, 20\}$. In each of the 60 possible settings, the sample size was fixed at $n = 250$.

Figure 3.4 gives the F_1 performance measure for each variable selection method for 8 of the 60 simulation settings. We choose to illustrate these simulation results as they are representative of our overall findings. Here, higher values of F_1 indicate better performance. Complete tables of precision, recall, and F_1 measure values for the simulations shown in Figure 3.4 can be found in the supplementary materials of Bleich et al. (2014).

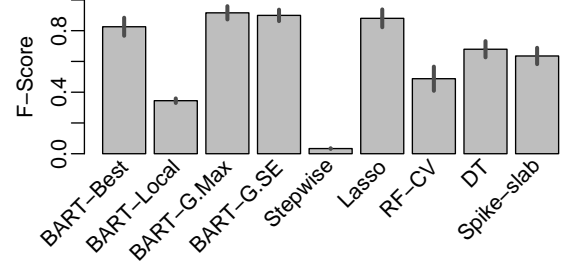
We first focus on the comparisons in performance between the four thresholding strategies for our BART-based variable selection procedure: our three thresholding strategies plus the BART-Best cross-validated threshold strategy. First, we consider the case where $p = 200$. In the more sparse settings (Figures 3.4a and 3.4b), the more stringent global max and global SE strategies perform better than the less stringent local thresholding strategy. However, the local thresholding strategy performs better in the less sparse settings (Figures 3.4c and 3.4d). The BART-Best procedure with a cross-validated threshold performs slightly worse than the best of the three thresholds in each setting, but fares quite well uniformly. Hence, the cross-validated threshold strategy represents a good choice when the level of sparsity is not known *a priori*.

For the settings where $p = 500$, the findings are relatively similar. The local thresholding strategy performs well given the fact that the data is less sparse. Performance also degrades when moving from the low noise settings (Figure 3.4e and 3.4g) to the high noise settings (Figure 3.4f and 3.4h). Note that BART-Best does not perform particularly well in Figure 3.4h.

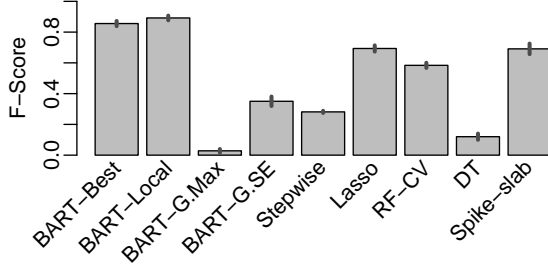
Compared to the alternative approaches when $p = 200$, we see that BART-Best performs better than all of the alternatives in the lower noise, more sparse setting (Figures 3.4a) and is competitive with the lasso in the lower noise, less sparse setting



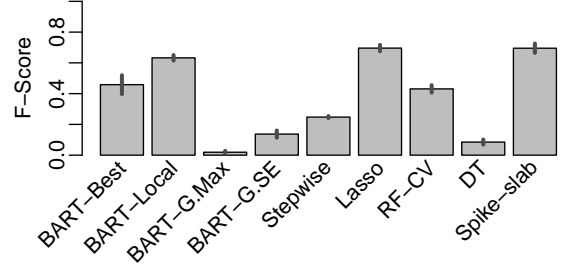
(a) $p = 200, p_0 = 2, \sigma^2 = 5$



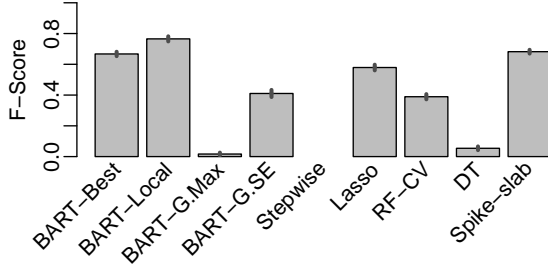
(b) $p = 200, p_0 = 2, \sigma^2 = 20$



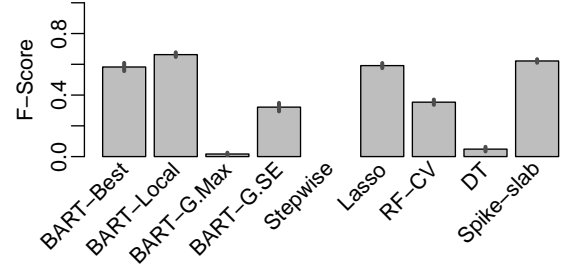
(c) $p = 200, p_0 = 20, \sigma^2 = 5$



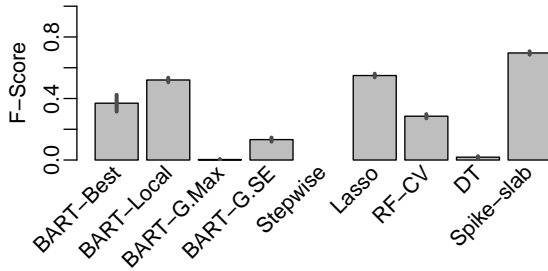
(d) $p = 200, p_0 = 20, \sigma^2 = 20$



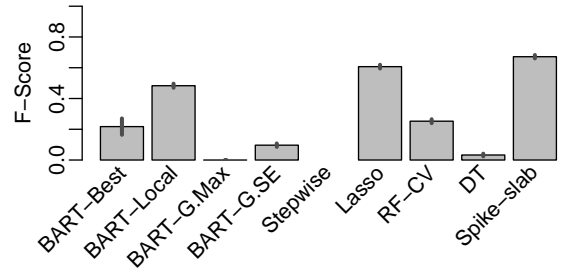
(e) $p = 500, p_0 = 25, \sigma^2 = 1$



(f) $p = 500, p_0 = 25, \sigma^2 = 5$



(g) $p = 500, p_0 = 50, \sigma^2 = 1$



(h) $p = 500, p_0 = 50, \sigma^2 = 5$

Figure 3.4: Average F_1 measures for different variable selection approaches on simulated data under the linear model setting across 50 simulations. The black bars represent 90% error bars for the average. Results for $p = 200$ and $p = 500$ are shown.

(Figure 3.4c). **BART-Best** is competitive with the lasso in the higher noise, more sparse setting (Figure 3.4b) and beaten by the linear methods in the higher noise, less sparse setting (Figure 3.4d). When $p = 500$, the cross-validated **BART** is competitive with the lasso and **Spike-slab** and outperforms the nonlinear methods when $p_0 = 25$ (Figure 3.4e and 3.4f). When $p_0 = 50$ (Figure 3.4g and 3.4h), the cross-validated **BART** performs worse than the lasso and **Spike-slab**, and has performance on par with the cross-validated **RF**.

Overall, the competitive performance of our **BART**-based approach is especially impressive since **BART** does not assume a linear relationship between the response and predictor variables. One would expect that stepwise regression, lasso regression, and **Spike-slab** would have an advantage since these methods assume a linear model which matches the data generating process in this setting. Like **BART**, **RF** and **DT** also do not assume a linear model, but in most of the cases we examined, our **BART**-based variable selection procedure performs better than **RF** and **DT**. We note that **DT** does not perform well on this simulation, possibly suggesting the need for a cross-validation procedure to choose appropriate relevance thresholds in different data settings.

Additionally, we briefly address the computational aspect of our four proposed approaches here by giving an estimate of the runtimes. For this data with $n = 250$ and $p = 200$, the three strategies (local, global max and global SE) are estimated together in one `bartMachine` function in about 90 seconds. The cross-validated **BART-Best** procedure takes about 7 minutes.

3.4.2 Simulation Setting 2: Nonlinear Relationship

We next examine the performance of the variable selection methods in a situation where the response variable is a nonlinear function of the predictor variables. Specif-

ically, we generate each predictor vector \mathbf{x}_j from a uniform distribution,

$$\mathbf{x}_1, \dots, \mathbf{x}_p \stackrel{iid}{\sim} \text{U}_n(0, 1),$$

and then the response variable \mathbf{y} is generated via the Friedman function as

$$\mathbf{y} = 10\sin(\pi\mathbf{x}_1\mathbf{x}_2) + 20(\mathbf{x}_3 - .5)^2 + 10\mathbf{x}_4 + 5\mathbf{x}_5 + \boldsymbol{\mathcal{E}}, \quad \boldsymbol{\mathcal{E}} \sim \mathcal{N}_n(\mathbf{0}, \sigma^2\mathbf{I}). \quad (3.7)$$

The Friedman function is challenging for variable selection models due to its interactions and nonlinearities. In this data setting, only the first five predictors truly influence the response, while any additional predictor variables are spurious.

Fifty data sets were generated for each possible combination of $\sigma^2 \in \{5, 100, 625\}$ and $p \in \{25, 100, 200, 500, 1000\}$. Since the number of relevant predictor variables is fixed at five, we simulate over a wide range of sparsity values ranging from $p_0/p = 0.2$ down to $p_0/p = 0.005$. In each data set, the sample size was fixed at $n = 250$.

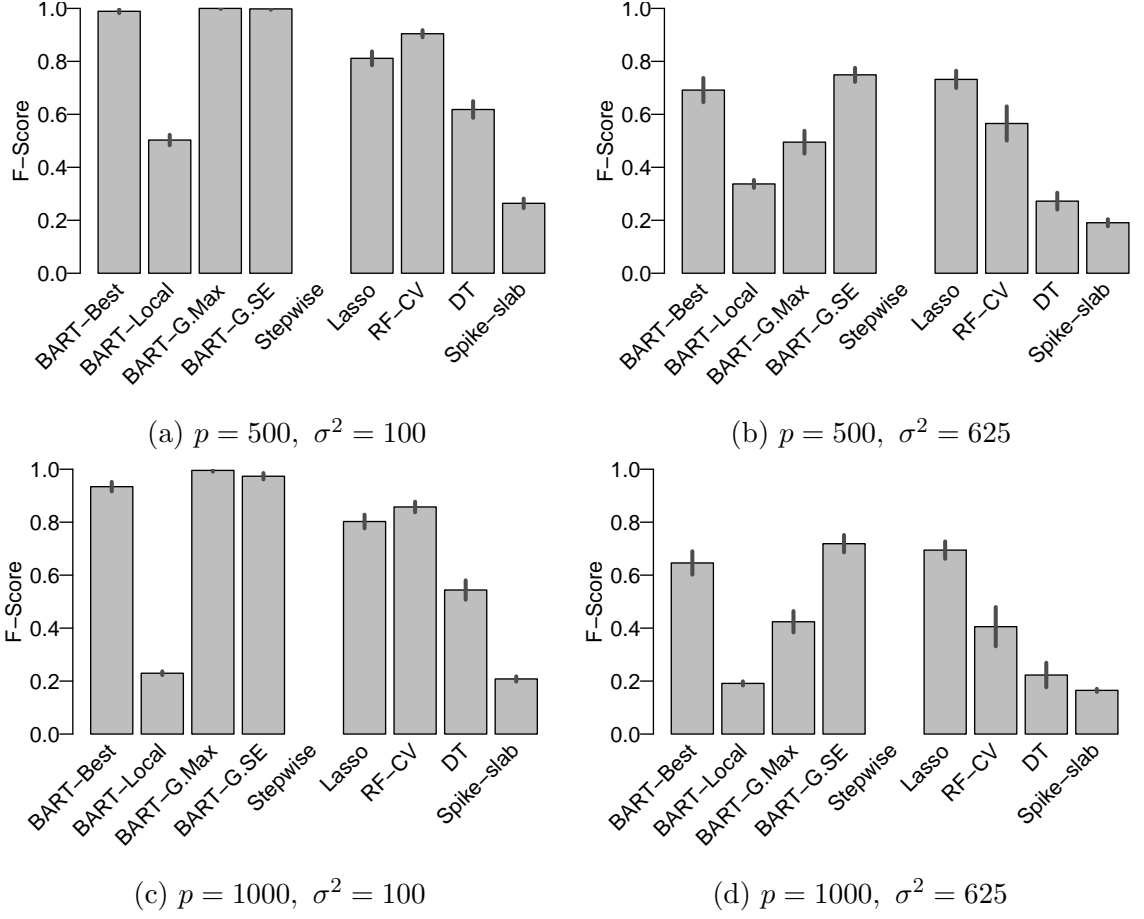


Figure 3.5: Average F_1 measures across 50 simulations for different variable selection approaches on simulated data under the Friedman model setting. The black bars represent 90% error bars for the average. Moving from the top row to the bottom shifts from low to high dimensionality and moving from the left column to the right shifts from low to high noise.

Figure 3.5 illustrates the F_1 performance measure for each variable selection method for four of the (p, σ^2) simulation pairs. We have chosen to illustrate these simulation results as they are representative of our overall findings. Backward stepwise regression via `stepAIC` could not be run in these settings where $n < p$ and is excluded from these comparisons (values in Figure 3.5 for this procedure are set to

0). Complete tables of precision, recall, and F_1 measure values for the simulations shown in Figure 3.5 are given in the supplementary materials of Bleich et al. (2014).

Just comparing the four thresholding strategies of our BART-based procedure, we see that the more stringent selection criteria have better F_1 performance measures in all of these sparse cases. The cross-validated threshold version of our BART procedure performs about as well as the best individual threshold in each case.

Compared to the other variable selection procedures, the cross-validated BART-Best has the strongest overall performance. Our cross-validated procedure outperforms DT and RF-CV in all situations. The assumption of linearity puts the lasso and Spike-slab at a disadvantage in this nonlinear setting. Spike-slab does not perform well on this data, although lasso performs well.¹⁰ BART-Best and the cross-validated RF have the best performance in the low noise settings (Figures 3.5a and 3.5c), as they do not assume linearity. Moving to the high noise settings (Figures 3.5b and 3.5d), BART and RF both see a degradation in performance, and BART-Best and the lasso are the best performers, followed by the cross-validated RF.

3.4.3 Simulation Setting 3: Linear Model with Informed Priors

In the next set of simulations, we explore the impact of incorporating informed priors into the BART model, as discussed in Section 3.3.4. We will evaluate the performance of our BART-based variable selection procedure in cases where the prior information is correctly specified as well as in cases where the prior information is incorrectly specified.

¹⁰We note that the lasso’s performance here is unexpectedly high. For this example, lasso is able to recover the predictors that are interacted within the sine function. This seems to be an artifact of this particular data generating process, and we would expect lasso to perform worse on other nonlinear response functions.

We will use the linear model in Section 3.4.1 as our data generating process. We will consider a specific case of the scheme outlined in Section 3.3.4 where particular subsets of predictor variables are given twice as much weight as the rest of the predictor variables. With a noninformative prior, each predictor variable has a probability of $1/p$ of being selected as the splitting variable for a splitting rule. For the informed prior, a subset of p_0 predictor variables are given twice as much weight, which gives those variables a larger probability of $2/(p + p_0)$ of being selected as a splitting variable.

For the fifty data sets generated under each combination of the parameter settings in the simulations of Section 3.4.1, we implemented three different versions of **BART**: (1) **BART** with a noninformative prior on the predictor variables, (2) **BART** with a “correctly” informed prior (twice the weight on the subset of predictor variables that have a true effect on response) and (3) **BART** with an “incorrectly” informed prior (twice the weight on a random subset of spurious predictor variables). For each of these **BART** models, predictor variables were then selected using the cross-validated threshold strategy.

Figure 3.6 gives the F_1 measures for the three different **BART** priors in four of the data settings outlined in Section 3.4.1.

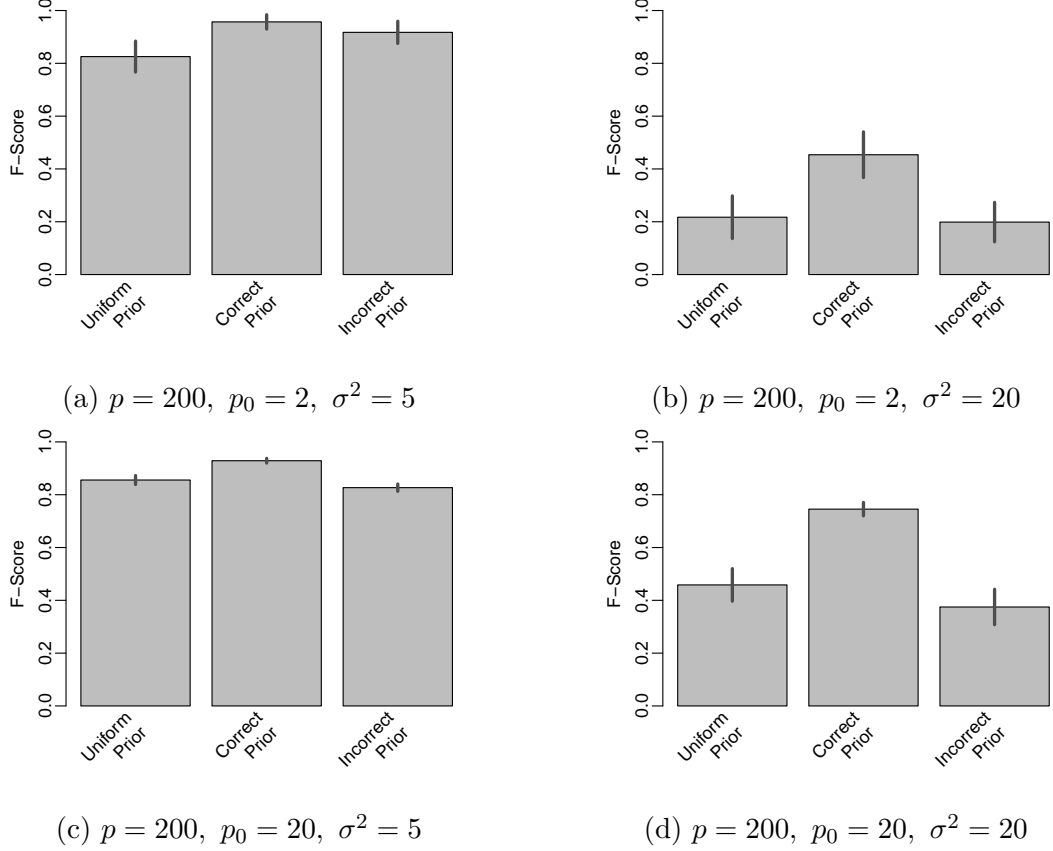


Figure 3.6: Average F_1 measures across 50 simulations for **BART**-based variable selection under three different prior choices. The black bars represent 90% error bars for the average. The settings shown are the same as those in Figures 3.4a-3.4d.

There are two key observations from the results in Figure 3.6. First, correct prior information can substantially benefit the variable selection ability of our **BART** adaptation, especially in higher noise settings (Figures 3.6b and 3.6d). Second, incorrect prior information does not degrade performance in any of the cases, which suggests that our **BART**-based variable selection procedure is robust to the mis-specification of an informed prior on the predictor variables. This seems to be a consequence of the Metropolis-Hastings step, which tends to not accept splitting rules that substantially reduce the model's posterior value, regardless of how often they are proposed.

To summarize our simulation studies in Section 3.4, our BART-based variable selection procedure is competitive with alternative approaches when there is linear relationship between the predictor variables and the response, and performs better than alternative approaches in a nonlinear data setting. BART-based variable selection can be further improved by correctly specifying prior information (when available) that gives preference to particular predictor variables, and appears to be robust to mis-specification of this prior information.

3.5 Application to Gene Regulation in Yeast

Experimental advances in molecular biology have led to the availability of high-dimensional genomic data in a variety of biological applications. We will apply our BART-based variable selection methodology to infer the gene regulatory network in budding yeast (*Saccharomyces cerevisiae*). One of the primary mechanisms by which genes are regulated is through the action of transcription factors, which are proteins that increase or decrease the expression of a specific set of genes.

The data for our analyses are expression measures for 6026 genes in yeast across 314 experiments. For those same 314 experiments, we also have expression measures for 39 known transcription factors. For each of the 6026 genes, our goal is to identify the subset of the 39 transcription factors that have a real regulatory relationship with that particular gene.

We consider each of the 6026 genes as a separate variable selection problem. For a particular gene g , we model the expression measures for that gene as a 314×1 response vector \mathbf{y}_g and we have 39 predictor variables $(\mathbf{x}_1, \dots, \mathbf{x}_{39})$ which are the expression measures of each of the 39 transcription factors. This same data were previously analyzed using a linear regression approach in Jensen et al. (2007), but

we will avoid assumptions of linearity by employing our **BART**-based variable selection procedure.

We also have additional data available for this problem that can be used as prior information on our predictor variables. Lee et al. (2002) performed chromatin immunoprecipitation (ChIP) experiments for each of the 39 transcription factors that we are using as predictor variables. The outcome of these experiments is the estimated probabilities m_{gk} that gene g is physically bound by each transcription factor k . Chen et al. (2007) give details on how these probabilities m_{gk} are derived from the ChIP data.¹¹

We will incorporate these estimated probabilities into our **BART**-based variable selection approach as prior information. When selecting predictor variables for splitting rules, we give more weight to the transcription factors k with larger prior probabilities m_{gk} in the **BART** model for gene g . Specifically, we have a splitting variable weight w_{gk} for predictor \mathbf{x}_k in the **BART** model for gene g , which we calculate as

$$w_{gk} = 1 + c \cdot m_{gk}. \quad (3.8)$$

In the **BART** model for gene g , each predictor \mathbf{x}_k is chosen for a splitting rule with probability proportional to w_{gk} . The global parameter c controls how influential the informed prior probabilities m_{gk} are on the splitting rules in **BART**. Setting $c = 0$ reduces our informed prior to the uniform splitting rules of the standard **BART** implementation. Larger values of c increases the weights of predictor variables with large prior probabilities m_{gk} , giving the informed prior extra influence.

In a real data setting such as our yeast application, it is difficult to know how much influence to give our informed priors on the predictor variables. We will consider several different values of $c = \{0, 1, 2, 4, 10000\}$ and choose the value that results in

¹¹Probabilities were truncated to be between 5% and 95%.

the smallest prediction error on a subset of the observed data that is held-out from our **BART** model estimation. Specifically, recall that we have 314 expression measures for each gene in our data set. For each gene, we randomly partition these observations into an 80% training set, 10% tuning set, and 10% hold-out set. For each value of $c = \{0, 1, 2, 4, 10000\}$, we fit a **BART** model on the 80% training set and then choose the value of c that gives the smallest prediction error on the 10% tuning set. This same 10% tuning set is also used to choose the best threshold procedure among the three options outlined in Section 3.3.3. We will use the terminology “**BART-Best**” to refer to the **BART**-based variable selection procedure that is validated over the choice of c and the three thresholding strategies. While we could also cross-validate over the significance level α , we fix $\alpha = .05$ due to computational concerns given the large number of data sets to be analyzed.

For each gene, we evaluate our approach by re-fitting **BART** using only the variables selected by our **BART**-based variable selection model and evaluate the prediction accuracy on the final 10% hold-out set of data for that gene. This same 10% hold-out set of data for each gene is also used to evaluate the prediction accuracy of various alternative variable selection methods. We consider the alternative methods of stepwise regression, lasso regression, **RF**, **DT**, and **Spike-slab** in a similar fashion to Section 3.4. The 10% tuning set is used to choose the value of the penalty parameter λ for lasso regression as well as the importance score threshold for **RF**. For **DT**, we use a constant leaf model for variable selection and then construct a linear leaf model using the selected variables for prediction.

We also consider three simpler approaches that do not select particular predictor variables: (1) “**BART-Full**” which is the **BART** model using all variables, (2) ordinary least squares regression (**OLS**) with all predictor variables included and (3) the “null” model: the sample average of the response which does not make use of any predictors.

In the null model, we do include an intercept so we are predicting for the hold-out set of expression measures for each gene with the average expression level of that gene in the training set.

We first examined the distribution of RMSEs across the 6026 genes. We found that each procedure improves over the null model with no covariates, suggesting that some subset of transcription factors is predictive of gene expression for most of the 6026 genes. However, for a minority of genes, the null model is competitive, suggesting that the 39 available transcription factors may not be biologically relevant to every one of these genes. The non-null variable selection methods show generally similar performance in terms of the distribution of RMSEs, and a corresponding figure can be found in the supplementary materials of Bleich et al. (2014). It is important to note that predictive accuracy in the form of out-of-sample RMSE is not the most desirable metric for comparing variable selection techniques because it overweights recall relative to precision.

In Figure 3.7, we show the distribution of the number of selected predictor variables (TFs) across the 6026 genes, where we see substantial differences between the variable selection procedures. Figure 3.7 confirms that **BART-G.max** is selecting very few TFs for each gene. Even more interesting is the comparison of **BART-Best** to stepwise regression, lasso regression, **RF**, and **Spike-slab**. **BART-Best** is selecting far fewer TFs than these alternative procedures. Interestingly, **DT**, the other Bayesian tree-based algorithm, selects a number of TFs most comparable to **BART-Best**.

Given the relatively similar performance of methods in terms of RMSE and the more substantial differences in number of variables selected, we propose the following combined measure of performance for each variable selection method:

$$(\text{RMSE reduction per predictor})_{\text{method}} = \frac{\text{RMSE}_{\text{null}} - \text{RMSE}_{\text{method}}}{\text{NumPred}_{\text{method}}},$$

where $\text{RMSE}_{\text{method}}$ and $\text{NumPred}_{\text{method}}$ are respectively the out-of-sample RMSE and number of predictors selected for a particular method. This performance metric answers the question: how much “gain” are we getting for adding each predictor variable suggested by a variable selection approach? Methods that give larger RMSE reduction per predictor variable are preferred.

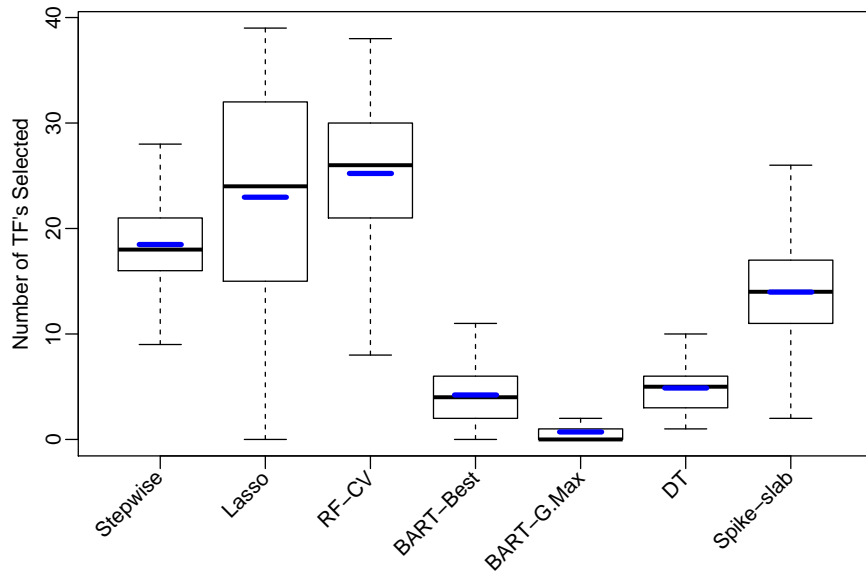


Figure 3.7: Distributions of the number of predictor variables selected for each method across all 6026 genes. Blue bars represent the average number of selected predictor variables. Not shown are the null model which uses no predictors as well as OLS and the full BART model which both use all predictors. Points beyond the whiskers are omitted.

Figure 3.8 gives the RMSE reduction per predictor for each of our variable selection procedures. Note that we only plot cases where at least one predictor variable

is selected, since RMSE reduction per predictor is only defined if the number of predictors selected is greater than zero.

Our **BART-Best** variable selection procedure gives generally larger (better) values of the RMSE reduction per predictor measure than stepwise regression, lasso regression, RF, and **Spike-slab**. DT is the closer competitor, but does slightly worse, on average, than **BART-Best**. Also, both the **BART-Full** and OLS procedures, where no variable selection is performed, perform worse than the variable selection procedures.

BART-G.max, the **BART**-based procedure under the global max threshold, seems to perform even better than the **BART-Best** procedure in terms of the RMSE reduction per predictor measure. However, recall that we are plotting only cases where at least one predictor was selected. **BART-G.max** selects at least one transcription factor for only 2866 of the 6026 genes, though it shows the best RMSE reduction per predictor in these cases. By comparison, **BART-Best** selects at least one transcription for 5459 of the 6026 genes while showing better RMSE reduction per predictor than the non-**BART** variable selection procedures.

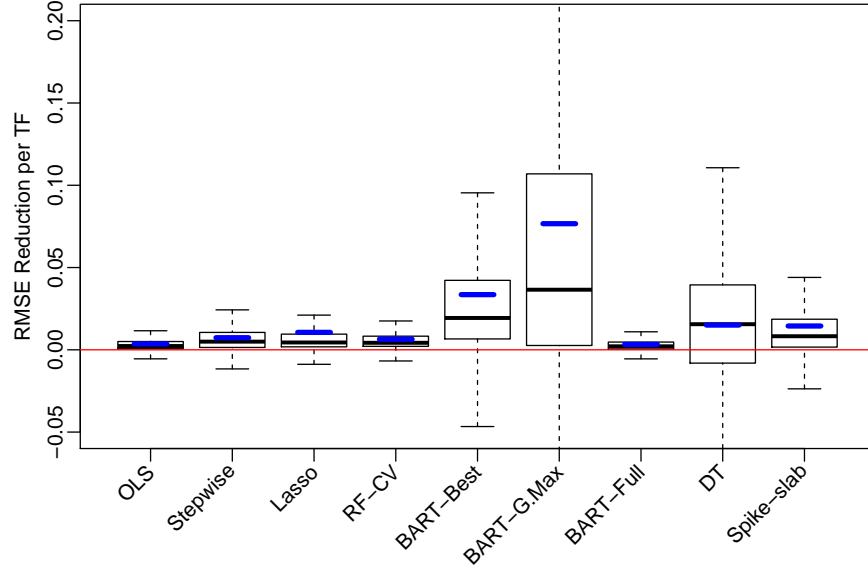


Figure 3.8: Distributions of the RMSE reduction per predictor for each method across all 6026 genes. Blue bars represent the average RMSE reduction per predictor. Points beyond the whiskers are omitted.

Additionally, Table 3.2 shows the proportion of times each choice of prior influence c appeared in the "BART-Best" model. Almost a quarter of the time, the prior information was not used. However, there is also a large number of genes for which the prior was considered to have useful information and was incorporated into the procedure.

c Value	Percentage of Genes
0	23.3%
0.5	16.1%
1	15.4%
2	14.9%
4	14.6%
10000	15.7%

Table 3.2: Distribution of prior influence values c used across the 6026 genes.

Jensen et al. (2007) also used the same gene expression data (and ChIP-based prior

information) to infer gene-TF regulatory relationships. A direct model comparison between our **BART**-based procedures and their approach is difficult since Jensen et al. (2007) fit a simultaneous model across all genes whereas our current **BART**-based analysis fits a predictive model for each gene separately. In both analyses, prior information for each gene-TF pairing from ChIP binding data (Lee et al., 2002) was used.¹² However, in Jensen et al. (2007) the prior information for a particular TF was given the same weight (relative to the likelihood) for each gene in the data set. In our analysis, each gene was analyzed separately and so the prior information for a particular TF can be weighted differently for each gene.

A result of this modeling difference is that the prior information appears to have been given less weight by our **BART**-based procedure across genes, as evidenced by the substantial proportion of genes in Table 3.2 that were given zero or low weight ($c = 0$ or $c = 0.5$). Since that prior information played the role in Jensen et al. (2007) of promoting sparsity, a consequence of that prior information being given less weight in our **BART**-based analysis is reduced promotion of sparsity.

This consequence is evident in Figure 3.9, where we compare the number of selected TFs. The x-axis gives the 39 transcription factors that served as the predictor variables for each of our 6026 genes. The y-axis is the number of genes for which that TF was selected as a predictor variable by each of three procedures: **BART**-Best, **BART**-G.max and the analysis of Jensen et al. (2007). The most striking feature of Figure 3.9 is that each TF was selected for many more genes under our **BART**-Best procedure compared to **BART**-G.max, which also selected more variables than the analysis of Jensen et al. (2007). This result indicates that selecting more TFs per gene leads to the best out-of-sample predictive performance (i.e. **BART**-Best). It could be

¹²Jensen et al. (2007) used additional prior information based on promoter sequence data that we did not use in our analysis.

that Jensen et al. (2007) was over-enforcing sparsity, but that previous method also differed from our current approach in terms of assuming a linear relationship between the response and predictor variables.

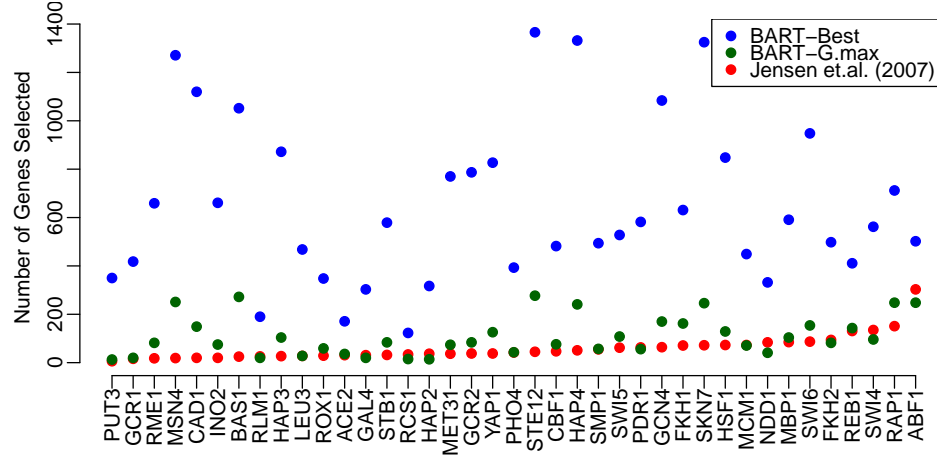


Figure 3.9: Number of genes for which each TF was selected. Results are compared for BART-Best, BART-G.Max, and the linear hierarchical model developed in Jensen et al. (2007).

3.6 Conclusion

We adapt BART to the task of variable selection by employing a permutation procedure to establish a null distribution for the variable inclusion proportion of each predictor. We present several thresholding strategies that reflect different beliefs about the degree of sparsity amongst the predictor variables, as well as a cross-validation procedure for choosing the best threshold when the degree of sparsity is not known *a priori*.

In contrast with popular variable selection methods such as stepwise regression and lasso regression, our BART-based approach does not make strong assumptions of

linearity in the relationship between the response and predictors. We also provide a principled means to incorporate prior information about the relative importance of different predictor variables into our procedures.

We used several simulated data settings to compare our BART-based approach to alternative variable selection methods such as stepwise regression, lasso regression, random forests, and dynamic trees. Our variable selection procedures are competitive with these alternatives in the setting where there is a linear relationship between response and predictors, and performs better than these alternatives in a nonlinear setting. Additional simulation studies suggest that our procedures can be further improved by correctly specifying prior information (if such information is available) and seems to be robust when the prior information is incorrectly specified.

We applied our variable selection procedure, as well as alternative methods, to the task of selecting a subset of transcription factors that are relevant to the expression of individual genes in yeast (*Saccharomyces cerevisiae*). In this application, our BART-based variable selection procedure generally selected fewer predictor variables while achieving similar out-of-sample RMSE compared to the lasso and random forests. We combined these two observations into a single performance measure, RMSE reduction per predictor. In this application of inferring regulatory relationships in yeast, our BART-based variable selection demonstrates much better predictive performance than alternative methods such as lasso and random forests while selecting more transcription factors than the previous approach of Jensen et al. (2007).

While we found success using the variable inclusion proportions as the basis for our procedure, fruitful future work would be to explore the effect of a variance reduction metric, such as that explored in Gramacy et al. (2013) within BART.

Citation

Bleich, J., Kapelner, A., George, E.I., and Jensen S.T. (2014). Variable selection for BART: An application to gene regulation. *Annals of Applied Statistics*, 8(3):1750-1781.

Incorporating Missingness into BART

Abstract

We present a method for incorporating missing data into general prediction problems which use nonparametric statistical learning. We focus on **BART** for incorporating missingness into decision trees. Our procedure extends the native partitioning mechanisms found in tree-based models and does not require imputation. Simulations on generated models and real data indicate that our procedure offers promise for both selection model and pattern-mixture frameworks as measured by out-of-sample predictive accuracy. We also illustrate **BART**'s abilities to incorporate missingness into uncertainty intervals. Our implementation is readily available in the **R** package `bartMachine`.

4.1 Introduction

This article addresses prediction problems where covariate information is missing during model construction and is also missing in future observations for which we are obligated to generate a forecast. Our aim is to develop a nonparametric statisti-

cal learning extension which incorporates missingness into *both* the training and the forecasting phases. In the spirit of nonparametric learning, we wish to incorporate the missingness into both phases automatically, without the need for pre-specified modeling or imputation.

We limit our focus to tree-based statistical learning, which has demonstrated strong predictive performance and has consequently received considerable attention in recent years. Popular implementations of **RF**, **SGB**, and similar algorithms do not incorporate covariate missingness natively without relying on either imputation or a complete case analysis of observations with no missing information. Additionally, no means for incorporating missing data in **BART** has been published to date. Our goal here is to develop a principled way of adapting **BART**’s machinery to incorporate missing data that takes advantage of the Bayesian framework.

Our proposed method modifies the recursive partitioning scheme during construction of the decision trees to incorporate missing data by the introduction of new splitting rules known as “missing incorporated in attributes” (MIA). Here, missingness itself also becomes a valid splitting criterion. By relying on the Metropolis-Hastings algorithm embedded in **BART**, our method frequently sends missing data to whichever of the two daughter nodes increases the overall model posterior value. Due to these benefits as well as conceptual simplicity, we chose to implement MIA-within-**BART** and we henceforth refer to it as “**BARTm**.”

Taking advantage of this modified set of splitting rules during model construction does not require imputation, a method which relies on assumptions that cannot be easily verified. Our approach is equally viable for continuous and nominal covariate data and both selection and pattern-mixture models for missing data.

Since missingness is handled natively within the algorithm, **BARTm** can generate predictions on future data with missing entries as well. The amount of uncertainty

associated with the predicted values increases with the amount of information lost due to missingness; thereby missingness is incorporated into the standard error of the prediction in a principled way. Also, our proposed procedure has negligible impact on the runtime during both model construction and prediction phases.

In Sections 4.2.1 - 4.2.3, we provide a framework for statistical learning with missingness with a focus on decision trees. We explain the adaptations of **BART** which yields **BARTm** in Section 4.3. We then demonstrate **BARTm**'s predictive performance on generated models in Section 4.4 as well as real data with a variety of missing data scenarios in Section 4.5. We conclude with Section 4.6. **BARTm** is implemented as a feature in **bartMachine** and a code demonstration is shown in Section 2.3.8.

4.2 Background

4.2.1 A Framework for Missing Data in Statistical Learning

Consider p covariates $\mathbf{X} := [X_1, \dots, X_p]$, a continuous response \mathbf{Y} and an unknown function f where $\mathbf{Y} = f(\mathbf{X}) + \boldsymbol{\varepsilon}$. We denote $\boldsymbol{\varepsilon}$ as the noise in the response unexplained by f . The goal of statistical learning is to use the *training set*, $[\mathbf{y}_{\text{train}}, \mathbf{X}_{\text{train}}]$ which consists of n observations drawn from the population $\mathbb{P}(\mathbf{Y}, \mathbf{X})$, to produce an estimate \hat{f} , the best guess of $\mathbb{E}[\mathbf{Y} \mid \mathbf{X}]$, which can then be used to generate predictions on future test observations with an unknown response. We denote these future observations as \mathbf{X}_* which we assume are likewise drawn from the same population as the training set.

Missingness is one of the scourges of data analysis, plaguing statistical learning by causing missing entries in both the training matrix $\mathbf{X}_{\text{train}}$ as well as missing entries in the future records \mathbf{X}_* . In the statistical learning context, the training set is

defined as the observations which do not exhibit missingness in their response, $\mathbf{y}_{\text{train}}$. Records with missing responses cannot be used to construct models which estimate f . Imputing missing values in the response for the new \mathbf{X}_* is equivalent to “prediction” and is the primary goal of statistical learning. Thus, “missingness” considered in this paper is missingness *only* in $\mathbf{X}_{\text{train}}$ and \mathbf{X}_* . We denote missingness in the $p_M \leq p$ features of \mathbf{X} which suffer from missingness as $\mathbf{M} := [M_1, \dots, M_{p_M}]$, binary vectors where 1 indicates missing and 0 indicates present, and covariates that are present with $\mathbf{X}_{\text{obs}} := [X_{\text{obs}_1}, \dots, X_{\text{obs}_p}]$. The main goal of statistical learning with missingness is to estimate $\mathbb{E}[\mathbf{Y} \mid \mathbf{X}_{\text{obs}}, \mathbf{M}]$.

To frame missing data models in statistical learning, we now borrow from the canonical framework of selection and pattern-mixture models with one key difference. As explained above, in the statistical learning context, \mathbf{Y} cannot be missing. Thus, \mathbf{M} denotes missingness in the covariates \mathbf{X} and not the conventional missingness in \mathbf{Y} .

Conditional on \mathbf{X} , *selection models* (Little, 1993, Equation 1) factor the full data likelihood as

$$\mathbb{P}(\mathbf{Y}, \mathbf{M} \mid \mathbf{X}, \boldsymbol{\theta}, \boldsymbol{\gamma}) = \mathbb{P}(\mathbf{Y} \mid \mathbf{X}, \boldsymbol{\theta}) \mathbb{P}(\mathbf{M} \mid \mathbf{X}, \boldsymbol{\gamma}) \quad (4.1)$$

where $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$ are parameter vectors which are assumed distinct. The first term on the right-hand side reflects that the marginal likelihood for the response $\mathbb{P}(\mathbf{Y} \mid \mathbf{X}, \boldsymbol{\theta})$ is independent of the missingness \mathbf{M} . The second term on the right conventionally conditions on \mathbf{Y} . In the forecasting paradigm explored herein, missingness is *assumed independent* of the response because \mathbf{Y} is often yet to be realized and thus its unknown value should not influence \mathbf{M} , the missingness of the previously realized covariates.

Conditional on \mathbf{X} , *pattern-mixture models* (Little, 1993, Equation 2) partition the full data likelihood as

$$\mathbb{P}(\mathbf{Y}, \mathbf{M} \mid \mathbf{X}, \boldsymbol{\theta}, \boldsymbol{\gamma}) = \mathbb{P}(\mathbf{Y} \mid \mathbf{M}, \mathbf{X}, \boldsymbol{\theta}) \mathbb{P}(\mathbf{M} \mid \mathbf{X}, \boldsymbol{\gamma}). \quad (4.2)$$

where $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$ are parameter vectors which are again assumed distinct. The difference between the above and Equation (4.1) is the marginal likelihood of the response may now depend on \mathbf{M} . This implies that there can be different response models under different patterns of missingness in the p_M covariates.

In both selection and pattern-mixture paradigms, the term on the right is the *missing data mechanism*, which traditionally (but not always) is the mechanism controlling missingness in the response. In our framework however, the missing data mechanism controls missingness only in \mathbf{X} : the covariates (along with additional parameters $\boldsymbol{\gamma}$) create missingness within themselves which inevitably needs to be incorporated during model construction and forecasting. Thus, the missing data mechanism is conceptually equivalent in both the selection and pattern-mixture paradigms in our framework.

The conceptual difference between the selection and pattern-mixture models in the statistical learning setting can be envisioned as follows. Imagine the full covariates \mathbf{X} are realized, but due to the missing data mechanism, \mathbf{X} is latent and we instead observe \mathbf{X}_{obs} and \mathbf{M} . In the selection paradigm, \mathbf{Y} is realized only from the full covariates via $\mathbb{P}(\mathbf{Y} \mid \mathbf{X}, \boldsymbol{\theta})$. However, in the pattern-mixture paradigm, both \mathbf{X} and \mathbf{M} intermix to create many collated response models $\{\mathbb{P}(\mathbf{Y} \mid \mathbf{X}, \boldsymbol{\theta}, \mathbf{M} = m)\}_{m \in \mathcal{M}}$ where each m -vector corresponds to a different arrangement of missingness in the covariates. Thus, under our assumptions, selection models are a subset of pattern-

mixture models. Note that pattern-mixture models are chronically under-identified and difficult to explicitly model in practice. We address why our proposed method is well-suited to handle prediction problems in pattern-mixture model scenarios in Section 4.3.

We now present Little and Rubin (2002)’s taxonomy of missing data mechanisms which are traditionally framed in the selection model paradigm but here apply to both paradigms: (1) missing completely at random (MCAR), (2) missing at random (MAR) and (3) not missing at random (NMAR). MCAR is a mechanism that generates missingness in the j th covariate X_j without regard to the value of X_j itself nor the values and missingness of any other covariates, denoted \mathbf{X}_{-j} ; it is determined by the exogenous parameter(s) γ exclusively. The MAR mechanism generates missingness in the j th covariate without regard to X_j , its own value, but can depend on values of other attributes \mathbf{X}_{-j} as well as γ . The NMAR mechanism features the additional dependence on the value of X_j itself as well as unobserved covariates (note that explicit dependence on unobserved covariates was not explored as missing data mechanisms in this paper). We summarize these mechanisms in Table 4.1.

In our framework, each of the $p_M \leq p$ covariates with missingness are assumed to have their own missing data mechanism. Thus, the full missing data mechanism for the whole covariate space, $\mathbb{P}(\mathbf{M} \mid \mathbf{X}, \gamma)$, can be arbitrarily convoluted, exhibiting combinations of MCAR, MAR and NMAR among its p_M covariates and each missing data mechanism relationship may be highly non-linear with complicated interactions. When developing our methodology, we make no outright assumptions on the forms or distributions of these mechanisms.

We conclude this section by emphasizing that in the nonparametric statistical learning framework where predictive performance is the objective, there is no need for explicit inference about $\boldsymbol{\theta}$ (which may have unknown structure and arbitrary, possibly

missing data mechanism	$\mathbb{P}(\mathbf{M}_j \mid X_{j,\text{miss}}, \mathbf{X}_{-j,\text{miss}}, \mathbf{X}_{-j,\text{obs}}, \boldsymbol{\gamma}) = \dots$
MCAR	$\mathbb{P}(\mathbf{M}_j \mid \boldsymbol{\gamma})$
MAR	$\mathbb{P}(\mathbf{M}_j \mid \mathbf{X}_{-j,\text{obs}}, \boldsymbol{\gamma})$
NMAR	(does not simplify)

Table 4.1: Missing data mechanism models in the context of statistical learning. \mathbf{M}_j is an indicator vector which takes the value one when the j^{th} covariate is missing for the i th observation. $\mathbf{X}_{-j,\text{obs}}$ are the observed values of the other covariates, besides j . $\mathbf{X}_{-j,\text{miss}}$ are the values of the other covariates, besides j , which are not observed because they are missing.

infinite, dimension). Instead, the learning algorithm performs “black-box” estimation of the data generating process such that the output \hat{f} estimates the $\mathbb{E}[\mathbf{Y} \mid \mathbf{X}_{\text{obs}}, \mathbf{M}]$ function. Thus, if we can successfully estimate this conditional expectation function directly, then accurate forecasts can be obtained. This is the approach that BARTm takes.

4.2.2 Strategies for Incorporating Missing Data in Statistical Learning

There are many nonparametric approaches to handling missing data that have been employed in statistical learning. The simplest strategy for incorporating missingness into model building is to simply remove the observations in $\mathbf{X}_{\text{train}}$ that contain at least one missing measurement. This is called “list-wise deletion” or “complete case analysis.” It is well known that complete case analysis will be unbiased for MCAR and MAR selection models where missingness does not depend on the response when the target of estimation is $\mathbb{E}[\mathbf{Y} \mid \mathbf{X}]$. However, when forecasting, the data analyst must additionally be guaranteed that \mathbf{X}_* (the future observations for which we wish to predict response values) has no missing observations, since it is not possible to

generate forecasts for any incomplete cases.

By properly modeling missingness, incomplete cases can be included and more information about $\mathbb{E}[\mathbf{Y} \mid \mathbf{X}]$ becomes available, potentially yielding higher predictive performance. One popular strategy is to impute the missing entries. The imputed $\mathbf{X}_{\text{train}}$ is then used *as if* it were the real covariate data when constructing \hat{f} and the imputed \mathbf{X}_* is then used as if it were the real covariate data during forecasting. To carry out imputation, the recommended strategy is to model the predictive distribution of the missing covariate(s) and then use draws from the model to fill in the missing values. *Multiple imputation* imputes many times and averages over the estimates for the parameter(s) of interest from each imputation’s “full” dataset (Rubin, 1978). In statistical learning, a prediction could be calculated by averaging the predictions from many \hat{f} ’s built from many imputed $\mathbf{X}_{\text{train}}$ ’s and then further averaging over many imputed \mathbf{X}_* ’s. In practice, having knowledge of both the missing data mechanism and each probability model is very difficult and has usually given way to nonparametric methods such as k -nearest neighbors (Troyanskaya et al., 2001) for continuous covariates and saturated multinomial modeling (Schafer, 1997) for categorical covariates. The widely used R package `randomForest` (Liaw and Wiener, 2002) imputes via “hot-decking” (Little and Rubin, 2002).

A more recent approach, `MissForest` (Stekhoven and Bühlmann, 2012), fits non-parametric imputation models for any combination of continuous and categorical input data, even when the response is unobserved. In this unsupervised procedure, initial guesses for the imputed values are made. Then, for each attribute with missingness, the observed values of that attribute are treated as the response and a RF model is fit using the remaining attributes as predictors. Predictions for the missing values are obtained via the trained RF model and serve as updated imputations. The process proceeds iteratively through each attribute with missingness and then repeats until

a stopping criterion is achieved. The authors argue that their procedure intrinsically constitutes multiple imputation due to RF’s averaging over many unpruned decision trees. The authors also state that their method will perform particularly well when “the data include complex interactions or non-linear relations between variables of unequal scales and different type.” Although no explicit reference is given to Little and Rubin (2002)’s taxonomy in their work, we expect **MissForest** to perform well in situations generally well-suited for imputation, namely, the MCAR and MAR selection models discussed in Section 4.2.1. **MissForest** would not be suited for NMAR missing data mechanisms as imputation values for X_j can only be modeled from \mathbf{X}_{-j} in their implementation. Additionally, implementing **MissForest** would not be recommended for pattern-mixture scenarios because imputation is insufficient to capture differing response patterns.

Since **BART** is composed primarily of a sum-of-regression-trees model, we now review strategies for incorporating missing data in tree-based models.

4.2.3 Missing data in Binary Decision Trees

There have been many previous efforts to handle missingness in decision trees, most of which rely on the structure of the decision trees themselves. Examples include surrogate variable splitting (Therneau and Atkinson, 1997), “Missing Incorporated in Attributes” (MIA, Twala et al., 2008, Section 2) and many others (see Ding and Simonoff, 2010 and Twala, 2009). MIA, the particular focus for this work, is a procedure that natively uses missingness when greedily constructing the rules for the decision tree’s internal nodes. More specifically, the procedure relies on a modification of the potential splitting rules at each partition. We summarize the procedure in Algorithm 3 and we explain how the expanded set of rules is incorporated into the

BART estimation procedure in Section 4.3.

Algorithm 3 *Splitting rule choices when constructing a new tree branch in MIA.*

The algorithm chooses one of the following three rules for all splitting variables and all splitting values c . Since there are p splitting attributes and at most $n - 1$ unique values to split on, a greedy splitting algorithm using MIA considers $2(n - 1)p + p_M$ possible splitting rules at each iteration instead of the classic $(n - 1)p$.

- 1: If x_{ij} is present and $x_{ij} \leq c$, send this observation left (\leftarrow); otherwise, send this observation right (\rightarrow). If x_{ij} is missing, send this observation left (\leftarrow).
 - 2: If x_{ij} is present and $x_{ij} \leq c$, send this observation left (\leftarrow); otherwise, send this observation right (\rightarrow). If x_{ij} is missing, send this observation right (\rightarrow).
 - 3: If x_{ij} is missing, send this observation left (\leftarrow); if it is present, regardless of its value, send this observation right (\rightarrow) .
-

There are many advantages of the MIA approach. First, MIA has the ability to model complex MAR and NMAR relationships, as evidenced in both Twala et al. (2008) and our results found in Sections 4.4 and 4.5. Since missingness is integrated into the splitting rules, forecasts can be made without imputing when \mathbf{X}_* contains missingness.

Another strong advantage of MIA is the ability to split on feature missingness (line 3 of Algorithm 3). This splitting rule choice allows for the tree to better capture pattern-mixture models where missingness directly influences the response model. Generally speaking, imputation ignores pattern-mixture models; missingness is only viewed as holes to be filled in and forgotten.

Due to these benefits as well as conceptual simplicity, we chose to implement MIA-within-BART, denoted “BARTm”, when enhancing BART to handle missing data.

4.3 Missing Incorporated in Attributes within BART

Implementing BARTm is straightforward. Recall from Section 1.4.1 that the prior on the splitting rules within the decision tree branches as being discrete uniform on the

possible splitting attributes and discrete uniform on the possible splitting values. To account for Lines 1 and 2 in the MIA procedure (Algorithm 3), the splitting attribute x_j and split value are proposed within **BART**, but now we additionally propose a direction (left or right with equal probability) for records to be sent when the records have missing values in x_j . A possible splitting rule would therefore be “ $x_{ij} < c$ and dispense to the left if x_{ij} is missing.” To account for Line 3 in the algorithm, splitting on missingness itself, we create dummy vectors of length n for each of the p_M attributes with missingness, denoted $\mathbf{M}_1, \dots, \mathbf{M}_{p_M}$, which assume the value 1 when the entry is missing and 0 when the entry is present. We then augment the original training matrix together with these dummies and use the augmented training matrix, $\mathbf{X}'_{\text{train}} := [\mathbf{X}_{\text{train}}, M_1, \dots, M_{p_M}]$, as the training data in the **BARTm** algorithm. Once again, the prior on the splitting rules is the same as in the original **BART** but now with the additional consideration that the direction of missingness is equally likely left or right conditional on the splitting attribute and value.

We expect **BARTm** to exhibit greater predictive performance over MIA in classical decision trees for two reasons. First, **BARTm**’s sum-of-trees model offers much greater fitting flexibility compared to a single tree. Additionally, due to the greedy nature of decision trees, once a split is chosen, the direction in which missingness is sent cannot be reversed. **BARTm** can alter its trees by pruning and regrowing nodes or changing splitting rules. These proposed modifications to the trees are accepted or rejected stochastically using the Metropolis-Hastings machinery depending on how strongly the proposed move increases the model’s posterior value.

We hypothesize that **BARTm**’s stochastic search for splitting rules allows observations with missingness to be grouped with observations having similar response values. Due to the Metropolis-Hastings step, the algorithm will attempt to move towards splitting rules and corresponding groupings that increase overall model like-

likelihood $\mathbb{P}(\mathbf{Y} \mid \mathbf{X}, \mathbf{M})$. In essence, **BARTm** is “feeling around” predictor space for a location where the missing data increases the overall marginal likelihood. For selection models, since splitting rules can depend on any covariate (including the covariate with missing data), it should be possible to generate successful groupings for the missing data under both MAR and NMAR mechanisms.

We describe simple examples of rules that increase overall model likelihood. Suppose there are two covariates X_1 and X_2 and we are fitting a **BARTm** model with one tree. In a simple MAR example, imagine a mechanism where X_2 is increasingly likely to go missing for large values of X_1 . The model can partition this data in two steps to increase overall likelihood: (1) A split on a large value of X_1 and then (2) a split on M_2 . As a simple NMAR example, suppose a mechanism where X_2 is more likely to be missing for large values of X_2 . **BARTm** can select splits of the form “ $x_2 > c$ and x_2 is missing” with c large. Here, the missing data is appropriately kept with larger values of X_2 and overall likelihood should be increased.

When missingness does not depend on any other covariates, it should be more difficult to find appropriate ways to partition the missing data, and we hypothesize that **BARTm** will be least effective for selection models with MCAR missing data mechanisms. We hypothesize this is due to the regularization prior on the depths of the trees coupled with the fact that all missing data must move to the same daughter node. In short, the trees do not extend deeply enough to create sufficiently complex partitioning schemes to handle the MCAR mechanism.

We also hypothesize that **BARTm** has potential to perform well on pattern-mixture models due to the partitioning nature of the regression tree. **BARTm** can partition the data based on different patterns of missingness by using missingness as a valid split value. Then, underneath these splits, different submodels for the different patterns can be constructed. More concretely, consider a simple saturated pattern mixture

model where the model is $f_A(X_1)$ if X_2 is missing and $f_B(X_1)$ if X_2 is present. The model can split immediately on M_2 and attempt to fit $f_A(X_1)$ in a tree below the left node and $f_B(X_1)$ in a tree below the right node.

In light of the above examples, it should be noted that the MIA steps within the Bayesian framework can also conceptually be viewed as combining pattern mixture models with imputation. Conditional on a splitting rule, non-missing values of a covariate are transformed into an indicator that takes the value of 1 if the splitting rule condition is satisfied. Here, MIA rule 1 effectively imputes 1 for the missing covariate and analogously MIA rule 2 effectively imputes 0 for the missing covariate.

Another motivation for adapting MIA to **BART** arises from computational concerns. **BART** is a computationally intensive algorithm, but its runtime increases negligibly in the number of covariates (see Chipman et al., 2010, Section 6). Hence, **BARTm** leaves the computational time virtually unchanged with the addition of the p_M new missingness dummy covariates. Another possible strategy would be to develop an iterative imputation procedure using **BART** similar to that in Stekhoven and Bühlmann (2012) or a model averaging procedure using a multiple imputation framework, but we believe these approaches would be substantially more computationally intensive.

4.4 Generated Data Simulations

4.4.1 A Simple pattern-mixture Model

We begin with an illustration of **BARTm**'s ability to directly estimate $\mathbb{E}[\mathbf{Y} \mid \mathbf{X}_{\text{obs}}, \mathbf{M}]$ and additionally provide uncertainty intervals. We consider the following nonlinear response surface:

$$\mathbf{Y} = g(X_1, X_2, X_3) + \mathbf{B}M_3 + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \stackrel{iid}{\sim} \mathcal{N}(0, \sigma_e^2), \quad B \stackrel{iid}{\sim} \mathcal{N}(\mu_b, \sigma_b^2), \quad (4.3)$$

$$g(X_1, X_2, X_3) = X_1 + X_2 + 2X_3 - X_1^2 + X_2^2 + X_1X_2$$

$$[X_1, X_2, X_3] \stackrel{iid}{\sim} \mathcal{N}_3 \left(\mathbf{0}, \sigma_x^2 \begin{bmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_1 \\ \rho_2 & \rho_1 & 1 \end{bmatrix} \right),$$

where $\sigma_x^2 = 1$, $\rho_1 = 0.2$, $\rho_2 = 0.4$, $\sigma_e^2 = 1$, $\mu_b = 10$ and $\sigma_b^2 = 0.5$. Note that the pattern-mixture model is induced by M_3 (the missingness in X_3). Under this missingness pattern, the response is offset by B , a draw from a normal distribution. Figure 4.1a displays the $n = 500$ sample of the response from the model coloured by M_3 to illustrate the separation of the two response patterns. We choose the following jointly NMAR missing data mechanism for X_2 and X_3 which was chosen to be simple for the sake of ensuring that the illustration is clear. The next section features more realistic mechanisms.

$$1 : X_2 \text{ is missing with probability } 0.3 \text{ if } X_2 \geq 0 \quad (4.4)$$

$$2 : X_3 \text{ is missing with probability } 0.3 \text{ if } X_1 \geq 0.$$

If the **BARTm** model assumptions hold and is successfully able to estimate $\mathbb{E}[\mathbf{Y} \mid \mathbf{X}_{\text{obs}}, \mathbf{M}]$, then the true $\mathbb{E}[\mathbf{Y} \mid \mathbf{X}_{\text{obs}}, \mathbf{M}]$ is highly likely to be contained within a 95% credible interval for the prediction. We first check to see whether **BARTm** can capture the correct response when $\mathbf{X}_{\text{train}}$ has missing entries but \mathbf{X}_* does not. Predicting for $\mathbf{x}_* = [0 \ 0 \ 0]$ should give $\mathbb{E}[\mathbf{Y} \mid \mathbf{X} = \mathbf{x}_*] = 0$ for the prediction. Figure 4.1b illustrates that **BARTm** captures the expected value within its 95% credible interval.

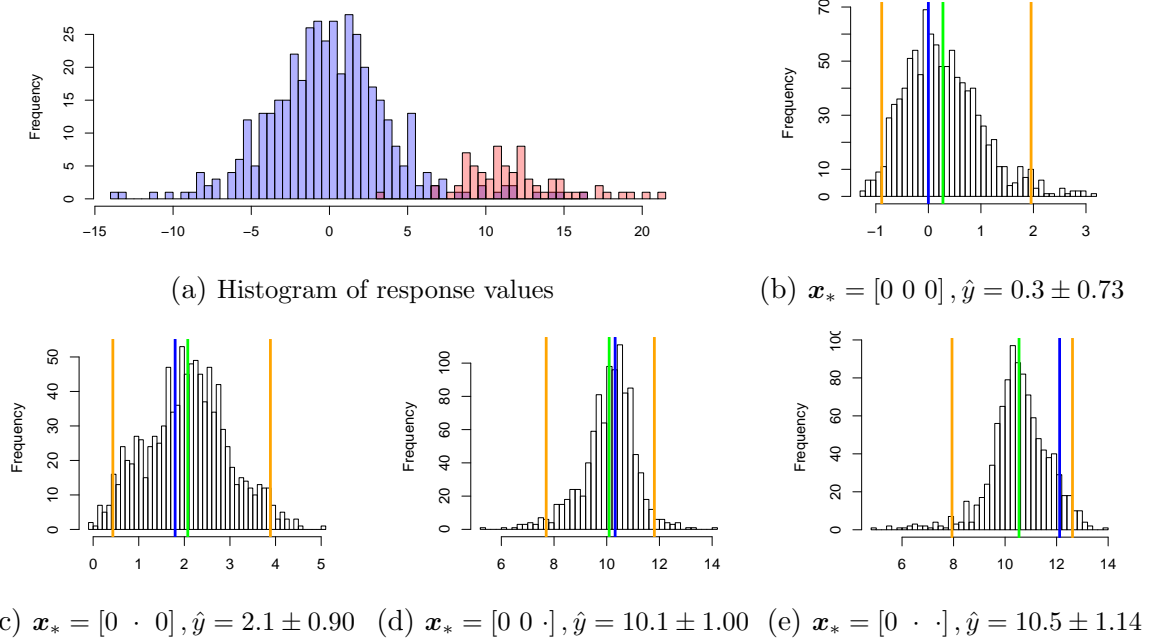


Figure 4.1: (a) A $n = 500$ sample of the responses of the model in Equation (4.3). Coloured in blue are the responses when X_3 is present and red are responses when X_3 is missing. (b-e) 1,000 burned-in posterior draws from a **BARTm** model for different values of \mathbf{x}_* drawn from the data generating process found in Equation (4.3). The **green** line is **BARTm**'s forecasted \hat{y} (the average of the posterior burned-in samples). The **blue** line is the true conditional expectation. The two **yellow** lines are the bounds of the 95% credible interval for $\mathbb{E}[\mathbf{Y} \mid \mathbf{X}_{\text{obs}} = \mathbf{x}_*, \mathbf{M} = \mathbf{m}_*]$.

Next we explore how well **BARTm** estimates the conditional expectation when missingness occurs within the new observation \mathbf{x}_* . We examine how **BARTm** handles missingness in attribute X_2 by predicting on $\mathbf{x}_* = [0 \ \cdot \ 0]$ where the “ \cdot ” denotes missingness. By Equation (4.4), X_2 is missing 30% of the time if X_2 itself is greater than 0. By evaluating the moments of a truncated normal distribution, it follows that **BARTm** should estimate $\mathbb{E}[X_2 + X_2^2 \mid X_2 > 0] = \sqrt{2/\pi} + 1 \approx 1.80$. Figure 4.1c indicates that **BARTm**'s credible interval captures this expected value. Note the larger variance of the

posterior distribution relative to Figure 4.1b reflecting the higher uncertainty due to x_{*2} going missing. This larger interval is an additional benefit of our methodology. As the trees are built during the Gibbs sampling, the splitting rules on X_2 are accompanied by a protocol for missingness: missing data will flow left or right (once a Metropolis-Hastings proposal is accepted) and this direction is chosen randomly. Thus, when \mathbf{x}_* is predicted with x_{*2} missing, missing records flow left and right over the many burned-in Gibbs samples creating a wider distribution of predicted values, and thus a wider credible interval. This is an important point — **BARTm** can give a rough estimate of how much information is lost when values in new records become missing by looking at the change in the standard error of a predicted value. Note that if **BART**’s hyperparameters are considered “tuning parameters,” the credible intervals’ endpoints are not interpretable. However, the relative lengths of the intervals can still signify different levels of forecast confidence to the practitioner.

We next consider how **BARTm** performs when X_3 is missing by predicting on $\mathbf{x}_* = [0 \ 0 \cdot]$. By Equation (4.4), **BARTm** should estimate $\mathbb{E}[X_3 \mid X_1 > 0] = .4\sqrt{2/\pi} \approx .32$ (which follows directly from the properties of the conditional distribution of a bivariate normal distribution recalling that $\text{Corr}[X_1, X_3] = 0.4$). When X_3 is missing, there is a different response pattern, and the response is shifted up by B . Since $\mathbb{E}[B] = 10$, **BARTm** should predict approximately 10.32. The credible interval found in Figure 4.1d indicates that **BARTm**’s credible interval again covers the conditional expectation.

Finally, we consider the case where X_2 and X_3 are simultaneously missing. Predicting on $\mathbf{x}_* = [0 \cdot \cdot]$ has a conditional expectation of $\mathbb{E}[X_2 + X_2^2 \mid X_2 > 0] + \mathbb{E}[X_3 \mid X_1 > 0] + \mathbb{E}[B] \approx 12.12$. Once again, the posterior draws displayed in Figure 4.1e indicate that **BARTm** reasonably estimates the conditional expectation. Note that the credible interval here is wider than in Figure 4.1d due to the additional missingness of X_2 .

Figure 4.1 illustrates a representative sample of prediction estimates with corresponding intervals for **BARTm**. In our experiments, we predicted across a wide range of values for \mathbf{x} (other than 0) and obtained similar results.

4.4.2 Selection Model Performance

In order to gauge **BARTm**'s out-of-sample predictive performance on selection models and to evaluate the improvement over model-building on complete cases, we construct the same model as in Equation (4.3) withholding the offset B (which previously induced the pattern-mixture). Thus $\mathbf{Y} = g(X_1, X_2, X_3) + \boldsymbol{\varepsilon}$. We impose three independently simulated scenarios illustrating performance under the following missingness mechanisms. The first is MCAR; X_1 is missing with probability γ . The second is MAR; X_3 is missing according to a non-linear probit model depending on the other two covariates:

$$\mathbb{P}(M_3 = 1 \mid X_1, X_2) = \Phi(\gamma_0 + \gamma_1 X_1 + \gamma_1 X_2^2). \quad (4.5)$$

The last is NMAR; X_2 is missing according to a similar non-linear probit model this time depending on itself and X_1 :

$$\mathbb{P}(M_2 = 1 \mid X_1, X_2) = \Phi(\gamma_0 + \gamma_1 X_1^2 + \gamma_1 X_2). \quad (4.6)$$

For each simulation, we set the number of training observations to $n = 250$ and simulate 500 times. Additionally, via appropriate selection of parameters, each simulation is carried out with varying levels of missing data: approximately $\{0, 10, \dots, 70\}$ percent of rows have at least one missing covariate entry. For the MCAR dataset,

the corresponding $\gamma = \{0, 0.03, 0.07, 0.11, 0.16, 0.26, 0.33\}$ and for both the MAR and NMAR datasets, $\gamma_0 = -3$ and $\gamma_1 = \{0, 0.8, 1.4, 2.0, 2.7, 4.0, 7.0, 30\}$.

For each missing data mechanism, we record results for four different scenarios: (1) $\mathbf{X}_{\text{train}}$ and \mathbf{X}_* contain missingness (2) $\mathbf{X}_{\text{train}}$ contains missingness and \mathbf{X}_* is devoid of missing data (in this case, \mathbf{X}_* is generated without the missing data mechanism to maintain a constant number of rows). (3) only complete cases of $\mathbf{X}_{\text{train}}$ are used to build the model but \mathbf{X}_* contains missingness and (4) only complete cases of $\mathbf{X}_{\text{train}}$ are used to build the model and \mathbf{X}_* is devoid of missing data.

We make a number of hypotheses about the relationship between the predictive performance of using incomplete cases (all observations) compared to the complete case performance. As we discussed in Section 4.3, **BARTm** should be able model the expectation of the marginal likelihood in selection models, thus we expect models built with incomplete cases to predict better than models that were built with only the complete cases. The complete case models suffer from performance degradation for two main reasons. First, these models are built with a smaller sample size and hence their estimate of $\mathbb{E}[\mathbf{Y} \mid \mathbf{X}_{\text{obs}}, \mathbf{M}]$ has higher variance. Second, the lack of missingness during the training phase does not allow the model to learn how to properly model the missingness, resulting in the missing data being filtered randomly from node to node during forecasting. These hypotheses are explored in Figure 4.2 by comparing the solid blue and solid red lines.

Further, during forecasting, we expect \mathbf{X}_* samples with incomplete cases to have worse performance than the full \mathbf{X}_* samples (devoid of missingness) simply because missingness is equivalent to information loss. However, for the NMAR model, as the amount of missingness increases, we expect predictive performance on \mathbf{X}_* without missingness to eventually be surpassed by the predictive performance on \mathbf{X}_* with missingness. Eventually there will be so much missingness in X_2 that (1) the trained

model on missingness will only be able to create models by using \mathbf{M}_2 and expect \mathbf{M}_2 in the future \mathbf{X}_* and (2) the trained model on complete cases will never observe the response of the function where X_2 went missing. These hypotheses are explored in Figure 4.2 by comparing the solid lines to the dashed lines within the same colour.

The results for the four scenarios under the three missing data mechanisms comport with our hypotheses. The solid red line is uniformly higher than the solid blue line, confirming degradation for complete case model forecasting on new data with missingness. The dotted lines are lower than their solid counterparts indicating that providing more covariate information yields higher predictive accuracy. The one exception is for NMAR. After the number of rows with missingness exceeds 40%, forecasts on only complete cases begin to perform worse than the forecasts on data with missingness for models built with missingness (BARTm).

For this set of simulations, BARTm performs better than BART models that ignore missingness in the training phase. The next section demonstrates BARTm’s performance on a real data set and compares its performance to a nonparametric statistical learning procedure that relies on imputation.

4.5 Real Data Example

The Boston Housing data (BHD) measures 14 features about housing in the $n = 506$ census tracts in Boston in 1970. For model building, the response variable is usually taken to be the median home value of the tract. Covariates we explicitly make use of in our exposition are the average number of rooms per dwelling (`rm`), per capita crime rate by town (`crim`), percent lower income status of the population (`lstat`), parts per million nitrogen oxide concentration in the air (`nox`), full-value property tax rate (`tax`), proportion of owner-occupied units built prior to 1940 (`age`), proportion of

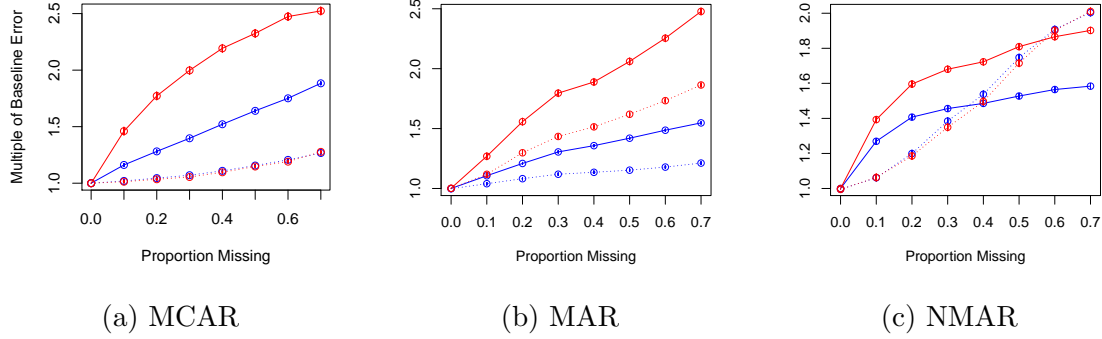


Figure 4.2: Simulation results of the response model for the three missing data mechanisms explained in the text. The y -axis measures the multiple of out-of-sample root mean square error (RMSE) relative to the performance in the absence of missingness. The x -axis is the approximate proportion of missing data. **Blue** lines correspond to the two scenarios where **BART** was built with all cases in $\mathbf{X}_{\text{train}}$ and **red** lines correspond to the two scenarios where **BART** was built with only the complete cases of $\mathbf{X}_{\text{train}}$. Solid lines correspond to the two scenarios where \mathbf{X}_* included missing data and dotted lines correspond to the two scenarios where \mathbf{X}_* had no missing data. Vertical segments at each point illustrate 95% confidence intervals.

non-retail business acres per town (`indus`) and an index of accessibility to highways (`rad`).

For these simulations, we evaluate the performance of three procedures (1) `BARTm` (2) `RF` with $\mathbf{X}_{\text{train}}$ and \mathbf{X}_* imputed via `MissForest` and (3) `BART` with $\mathbf{X}_{\text{train}}$ and \mathbf{X}_* imputed via `MissForest`. Note that in these simulations we assume *a priori* that \mathbf{X}_* will have missing data. Thus, the complete case comparisons proposed in Section 4.4.2 were not possible. We gauge out-of-sample predictive performance as measured by the out-of-sample RMSE for the three procedures on the independent simulation scenarios described in Table 4.2.

Scenario	Description
Selection Model MCAR	<code>rm</code> , <code>crim</code> , <code>lstat</code> , <code>nox</code> and <code>tax</code> are each missing w.p. γ
Selection Model MAR	<code>rm</code> and <code>crim</code> are missing via: $\mathbb{P}(\mathbf{M}_{\text{rm}} = 1) = \Phi(\gamma_0 + \gamma_1(\text{indus} + \text{lstat} + \text{age}))$ $\mathbb{P}(\mathbf{M}_{\text{crim}} = 1) = \Phi(\gamma_0 + \gamma_1(\text{nox} + \text{rad} + \text{tax}))$
Selection Model NMAR	<code>rm</code> and <code>crim</code> are missing via: $\mathbb{P}(\mathbf{M}_{\text{rm}} = 1) = \Phi(\gamma_0 + \gamma_1(\text{rm} + \text{lstat}))$ $\mathbb{P}(\mathbf{M}_{\text{crim}} = 1) = \Phi(\gamma_0 + \gamma_1(\text{crim} + \text{nox}))$
Pattern-Mixture	The MAR selection model above and two offsets: (1) if $\mathbf{M}_{\text{rm}} = 1$, the response is increased by $\mathcal{N}(\mu_b, \sigma_b^2)$ (2) if $\mathbf{M}_{\text{crim}} = 1$, the response is decreased by $\mathcal{N}(\mu_b, \sigma_b^2)$

Table 4.2: Distinct missingness scenarios for the BHD simulations. Monospace codes are names of covariates in the BHD. Note that `rm` has sample correlations with `indus`, `lstat` and `age` of -0.39, -0.61 and -0.24 and `crim` has sample correlations with `nox`, `rad`, and `tax` of 0.42, 0.63 and 0.58. These high correlations should allow for imputations that perform well.

Similar to Section 4.4.2, each simulation is carried out with different levels of missing data, approximately $\{0, 10, 20, \dots, 70\}$ percent of rows have at least one missing covariate entry. For the MCAR scenario, the corresponding $\gamma = \{0, 0.02, 0.04, 0.07, 0.10, 0.13, 0.17\}$, for the MAR scenario and pattern-mixture

scenario, $\gamma_1 = \{0, 1.3, 1.5, 1.7, 2.1, 2.6, 3.1, 3.8\}$ and γ_0 is constant at -3 and for the NMAR scenario $\gamma_1 = \{0, 3.3, 3.6, 3.9, 4.1, 4.3, 4.6, 4.8\}$ and γ_0 is constant at -3. Similar to Section 4.4.1, we induce a pattern-mixture model by creating a normally distributed offset based on missingness (we create two such offsets here). Here, we choose μ_b to be 25% of the range in y and σ_b to be $\mu_b/4$. These values are arbitrarily set for illustration purposes. It is important to note that the performance gap of BARTm versus RF with imputation can be arbitrarily increased by making μ_b larger.

For each scenario and each level of missing data, we run 500 simulations. In each simulation, we first draw missingness via the designated scenario found in Table 4.2. Then, we randomly partition 80% of the 506 BHD observations (now with missingness) as $[\mathbf{y}_{\text{train}}, \mathbf{X}_{\text{train}}]$ and the remaining 20% as $[\mathbf{y}_*, \mathbf{X}_*]$. We build all three models (BARTm, RF with `MissForest` and BART with `MissForest`) on $\mathbf{X}_{\text{train}}$, forecast on \mathbf{X}_* and record the out-of-sample RMSE. Thus, we integrate over idiosyncrasies that could be found in a single draw from the missing data mechanism and idiosyncrasies that could be found in a single train-test partition. When using `MissForest` during training, we impute values for the missing entries in $\mathbf{X}_{\text{train}}$ using $[\mathbf{y}_{\text{train}}, \mathbf{X}_{\text{train}}]$ column-binded together. To obtain forecasts, we impute the missing values in \mathbf{X}_* using $[\mathbf{X}_{\text{train}}, \mathbf{X}_*]$ row-binded together then predict using the bottom rows (i.e. those corresponding to the imputed test data). Note that we use `MissForest` in both RF and BART to ensure that any difference in predictive capabilities of BART and random forests are not driving the results.

For the MCAR selection model, we hypothesize that the `MissForest`-based imputation procedures will outperform BARTm due to the conceptual reasons discussed in Section 4.3. For the MAR selection model, we hypothesize similar performance between BARTm and both `MissForest`-based imputation procedures, as both MIA and imputation are designed to perform well in this scenario. In the NMAR selection

model and pattern-mixture model, we hypothesize that **BARTm** will outperform both **MissForest**-based imputation procedures, as **MissForest** (1) cannot make use of the values in the missingness columns it is trying to impute and (2) cannot construct different submodels based on missingness. Although imputation methods are not designed to handle these scenarios, it is important to run this simulation to ensure that **BARTm**, which *is* designed to succeed in these scenarios, has superior out-of-sample predictive performance.

The results displayed in Figure 4.3 largely comport with our hypotheses. Methods using **MissForest** perform better on the MCAR selection model scenario (Figure 4.3a) and **BARTm** is stronger in the NMAR scenario (Figure 4.3c) and pattern-mixture scenario (Figure 4.3d). It is worth noting that in the MAR selection model scenario (Figure 4.3b), **BARTm** begins to outperform the imputation-based methods once the percentage of missing data becomes greater than 20%. The performance of the imputation-based algorithms degrades rapidly here, while **BARTm**’s performance remains fairly stable, even with 70% of the rows having at least one missing entry. In conclusion, **BARTm** performs better than **MissForest** because it is not “limited” to what can be imputed from the data on-hand. This advantage generally grows with the amount of missingness.

4.6 Discussion

We propose a means of incorporating missing data into statistical learning for prediction problems where missingness may appear during both the training and forecasting phases. Our procedure, **BARTm**, implements “missing incorporated in attributes” (MIA), a technique recently explored for use in decision trees, into Bayesian Additive Regression Trees, a newly developed tree-based statistical learning algorithm for clas-

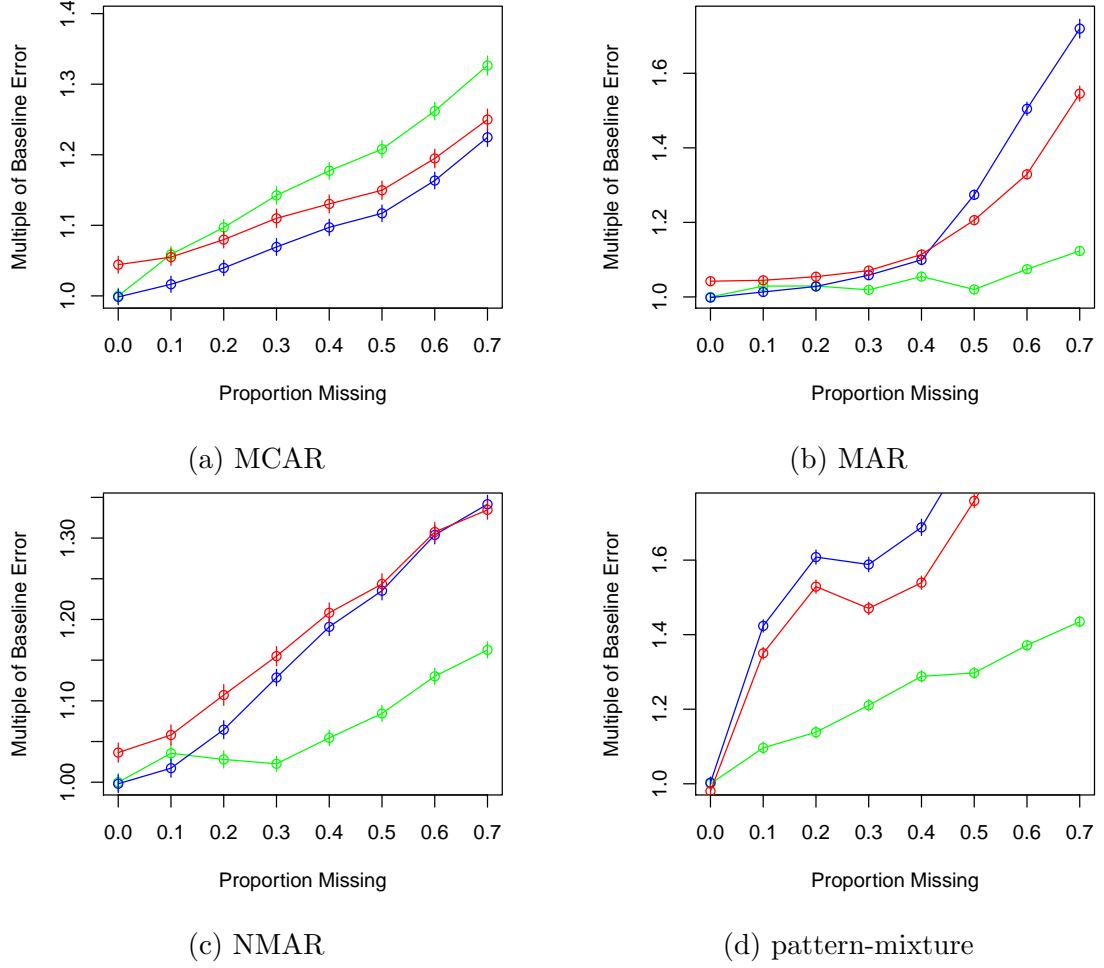


Figure 4.3: Simulations for different probabilities of missingness across the four simulated missing data scenarios in the BHD. The y-axis is out-of-sample RMSE relative to BART's out-of-sample RMSE on the full dataset. Lines in **green** plot BART's performance, lines in **red** plot RF with MissForest's performance, and lines in **blue** plot BART-with-MissForest's performance. Note that the MissForest-based imputation might perform worse in practice because here we allow imputation of the entire test set. In practice, it is likely that test observations appear sequentially. Vertical segments at each point illustrate 95% confidence intervals.

sification and regression. MIA natively incorporates missingness by sending missing observations to one of the two daughter nodes. Missingness is incorporated into splitting rules which are chosen via Metropolis-Hastings sampling. This innovation allows missingness itself to be used as a legitimate value within splitting criteria, resulting in no need for imputing in the training or new data and no need to drop incomplete cases.

For the simulations explored in this article, **BARTm**'s performance was generally superior to models built using complete cases, especially when missingness appeared in the test data as well. Additionally, **BARTm** provided higher predictive performance on the MAR selection model relative to **MissForest**, a nonparametric imputation technique. We also observe promising performance on NMAR selection models and pattern-mixture models in simulations. Additionally, **BARTm**'s Bayesian nature provides informative credible intervals reflecting uncertainty when the forecasting data has missing covariates.

Due to space considerations, the exploration in this article was focused on regression and we were unable to investigate our method for estimating probabilities in the binary classification setting. The R package **bartMachine** implements **BARTm** for both classification and regression and we view a survey of its classification performance as important future work.

Due to MIA's observed promise, we recommend it as a viable strategy to handle missingness in other tree-based statistical learning methods. Future work should also consider exploration of methods that combine imputation with MIA appropriately, in order to enhance predictive performance for MCAR missing data mechanisms. Other important future work includes investigating whether MIA's steps 1 and 2 can be altered so that each observation is sent in a random direction, an idea which draws inspiration from Rubin (1981)'s Bayesian bootstrap.

Citation

Kapelner, A. and Bleich, J. (2015). Prediction with missing data via Bayesian Additive Regression Trees. *Canadian Journal of Statistics, in press*.

BART with Parametric Models of Heteroskedasticity

Abstract

We incorporate heteroskedasticity into **BART** by modeling the log of the error variance parameter as a linear function of prespecified covariates. Under this scheme, the Gibbs sampling procedure for the original sum-of-trees model is easily modified, and the parameters for the variance model are updated via a Metropolis-Hastings step. We demonstrate the promise of our approach by providing more appropriate posterior predictive intervals than homoskedastic **BART** in heteroskedastic settings and demonstrating the model's resistance to overfitting.

5.1 Introduction

We consider the the following general heteroskedastic regression framework to characterize the relationship between a continuous response vector \mathbf{y} and a set of p predictor variables $\mathbf{X} := [\mathbf{x}_1, \dots, \mathbf{x}_p]$ which can be continuous or categorical:

$$\mathbf{y} = f(\mathbf{X}) + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}_n(\mathbf{0}, \sigma^2 \mathbf{D})$$

\mathbf{D} denotes the diagonal matrix whose entries are scaling factors for the error variance for each observation. In this model, the response is considered to be an unknown function f of the predictors and the observations, while independent, exhibit non-constant error variance. The goal of this article is to model the relationship between the predictors and response with the aim of generating accurate predictions. To this end we model f with **BART**. As developed in Chapter 1, the original **BART** model is constrained to have homoskedastic error variance ($\mathbf{D} = \mathbf{I}$). Here, we extend the model to flexibly handle an error variance structure which is a linear model of pre-specified covariates and we name our procedure “heteroskedastic **BART**” or “**HBART**.” Similar to Huber-White sandwich estimation (White, 1980), appropriately modeling the diagonal entries of \mathbf{D} “downweights” high variance observations. This allows for (a) a more accurate model as measured by predictive performance on future observations as well as (b) posterior credible and predictive intervals which appropriately reflect the changing heteroskedasticity in predictor space.

In Section 5.2, we provide an overview of the literature on heteroskedastic regression modeling in a Bayesian paradigm. In Section 5.3, we introduce **HBART**, highlighting the necessary modifications to the original homoskedastic **BART**. In Section 5.4, we provide simulations to showcase the desirable properties of **HBART**, including less overfitting for high noise observations as well as more appropriate uncertainty intervals for predictions in the presence of heteroskedasticity. Section 5.5 explores two applications to real data. We conclude and offer future research directions in Section 5.6. The method developed in this chapter will be implemented in an upcoming

release of `bartMachine`.

5.2 Bayesian Heteroskedastic Regression

Early approaches for heteroskedastic regression primarily focused on point estimation for the parameters governing the underlying heteroskedasticity of the model (for an overview, see Carroll and Ruppert, 1988). Potential problems with point estimation gave rise to a proposal of a fully Bayesian approach for heteroskedastic linear regression, where the non-constant variance depends on simple functions of an unknown parameter θ and a set of weights w_i (Boscardin and Gelman, 1994).

Cepeda and Gamerman (2001) introduce a Bayesian regression model where the conditional mean of the response is modeled using a linear function of covariates $\mathbf{x}_1, \dots, \mathbf{x}_p$ plus heteroskedastic noise. They model the variance for each observation as a monotonic differentiable function of a linear combination of another set of covariates, $g(\mathbf{z}_1, \dots, \mathbf{z}_k)$. Additionally, the function g is chosen to ensure positivity of the variance terms. The authors rely on a block Gibbs sampling approach (Geman and Geman, 1984), sampling the parameters for the mean function and variance function in two stages. In particular, the parameters for the variance function are updated via a Metropolis-Hastings step (Hastings, 1970) using the approach of Gamerman (1997), which relies on an iteratively reweighted least squares model to generate suitable proposal distributions.

More recent approaches have focused on relaxing the assumptions of linear additive components for modeling the mean and variance functions. Yau and Kohn (2003) propose nonparametric models for each of these two functions by employing penalized regression spline estimation for both models. Chan et al. (2006) extend this nonparametric model to allow for semiparametric modeling of both the mean

and variance functions, using radial basis functions for nonparametric components. Additionally, their approach can handle a large number of basis terms by introducing Bayesian variable selection priors, thereby allowing model estimation to be locally adaptive. Leslie et al. (2007) relax the assumption of normal errors and developed a heteroskedastic linear regression model with general error distributions by relying on a Dirichlet process mixture prior.

Both Chan et al. (2006) and Leslie et al. (2007) rely on the sampling scheme developed in Gamerman (1997) to obtain draws from the posterior distribution of the parameters for the variance function. Our work similarly draws heavily on this technique.

5.3 Augmenting BART to Incorporate Heteroskedasticity

In its original formulation by Chipman et al. (2010), the authors assume that the response \mathbf{y} could be modeled as a sum-of-trees model of the covariates $\mathbf{X} := [\mathbf{x}_{\cdot 1}, \dots, \mathbf{x}_{\cdot p}]$ plus homoskedastic normal noise:

$$\mathbf{Y} = \sum_{i=1}^m \mathfrak{F}_i^{\mathcal{L}}(\mathbf{X}) + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}_n(\mathbf{0}, \sigma^2 \mathbf{I}_n). \quad (5.1)$$

We propose an extension to BART by allowing each $\sigma_1^2, \dots, \sigma_n^2$ to be scaled by the exponential of a linear parametric function of k covariates $\mathbf{Z} := [\mathbf{z}_{\cdot 1}, \dots, \mathbf{z}_{\cdot k}]$, the “heteroskedasticity covariates” which are potentially distinct from the covariates used to define the model for the mean function, $\mathbf{x}_{\cdot 1}, \dots, \mathbf{x}_{\cdot p}$. Our heteroskedastic model, HBART, is given as

$$\mathbf{Y} = \sum_{i=1}^m \mathfrak{F}_i^{\otimes}(\mathbf{X}) + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}_n \left(\mathbf{0}, \sigma^2 \begin{bmatrix} \exp(\mathbf{z}_{1\cdot} \cdot \boldsymbol{\gamma}) & & 0 \\ & \ddots & \\ 0 & & \exp(\mathbf{z}_{n\cdot} \cdot \boldsymbol{\gamma}) \end{bmatrix} \right) \quad (5.2)$$

where $\boldsymbol{\gamma} := [\gamma_1, \dots, \gamma_k]^\top$ is a column vector of linear coefficients for the k heteroskedasticity covariates. Thus, the variance of each observation is specified as a log-linear model:

$$\ln(\sigma_i^2) = \ln(\sigma^2) + \mathbf{z}_i \cdot \boldsymbol{\gamma} \quad \text{for } i = 1, \dots, n. \quad (5.3)$$

It is important to note that **BART**, by design, is an overparameterized model with “an abundance of unidentified parameters” (Chipman et al., 2010) allowing for a highly flexible fit. First, given the unidentifiable nature of the model, our focus is not on valid inference for $\boldsymbol{\gamma}$. Instead, we incorporate heteroskedasticity to aid in forecasting and generating posterior uncertainty intervals. Second, due to the already complex nature of the original **BART** algorithm, we employ parametric models for heteroskedasticity versus more sophisticated alternatives (such as the proposal of Chan et al., 2006) in order to prevent the model from becoming “too flexible.” Given highly flexible, unidentifiable estimation of both the mean and variance functions, the algorithm may have difficulty distinguishing between signal and noise, thereby shirking on its primary duty which is accurate estimation of the mean model. Hence, parametric models of heteroskedasticity represent the first step towards understanding how flexible **BART** can be in nonparametric function estimation when the homoskedasticity assumption is relaxed.

The remainder of the section is dedicated to describing the priors on **HBART** as

well as the Gibbs sampling procedure for obtaining posterior inference.

5.3.1 Priors and Likelihood

As discussed in Section 1.4.1, the **BART** model requires three priors. The first prior is on the tree structures themselves and the second is on the leaf parameters. The third prior is on the error variance σ^2 . **HBART** requires these same three priors as well as a prior on γ . By assumption, the priors on σ^2 , γ , \mathfrak{T} and \mathcal{L} are independent of one another:

$$\begin{aligned} \mathbb{P}\left(\mathfrak{T}_1^{\mathcal{L}}, \dots, \mathfrak{T}_m^{\mathcal{L}}, \sigma^2, \gamma\right) &= \left[\prod_t \mathbb{P}\left(\mathfrak{T}_t^{\mathcal{L}}\right) \right] \mathbb{P}\left(\sigma^2\right) \mathbb{P}(\gamma) \\ &= \left[\prod_t \mathbb{P}\left(\mathcal{L}_t \mid \mathfrak{T}_t\right) \mathbb{P}\left(\mathfrak{T}_t\right) \right] \mathbb{P}\left(\sigma^2\right) \mathbb{P}(\gamma) \\ &= \left[\prod_t \prod_{\ell} \mathbb{P}\left(\mu_{t,\ell} \mid \mathfrak{T}_t\right) \mathbb{P}\left(\mathfrak{T}_t\right) \right] \mathbb{P}\left(\sigma^2\right) \mathbb{P}(\gamma) \end{aligned}$$

where the last line follows from an assumption of conditional independence of the leaf parameters given the tree structure.

The priors on the tree structures, leaf parameters and splitting rules proposals are the same as those used in the original **BART** model and were discussed in Section 1.4.1.

For the homoskedastic implementation, recall that the prior is on the error variance and is chosen to be $\sigma^2 \sim \text{InvGamma}(\nu/2, \nu\lambda/2)$. λ is determined from the data so that there is a $q = 90\%$ a priori chance (by default) that the **BART** model will improve upon the RMSE from an ordinary least squares regression (therefore, the majority of the prior probability mass lies below the RMSE of a least squares regression). We use this same data-informed prior for the σ^2 parameter in **HBART**, the logarithm of which serves as the intercept term in the log-linear model for the variances (Equation 5.3).

Following the approach of Gamerman (1997), we place a multivariate normal prior on γ given as:

$$\gamma \sim \mathcal{N}_k \left(\gamma_0, \begin{bmatrix} \Sigma_{11} & & 0 \\ & \ddots & \\ 0 & & \Sigma_{kk} \end{bmatrix} \right) \quad (5.4)$$

We make the simplifying assumption that each component of the prior is independent of one another, but this can be easily generalized.

Along with a set of priors, **HBART** (and **BART**) consists of the likelihood of responses in the leaf nodes. The likelihood is assumed to be normal with the mean being the “best guess” of the leaf parameters at the current moment and variance being the best guess of the variance at the moment i.e. $\mathbf{y}_\ell \sim \mathcal{N}(\mu_\ell, \sigma_i^2/m)$. These “best guesses” are the values being conditioned on in the Gibbs sampler during each iteration.

5.3.2 Sampling from the Posterior

The Gibbs sampler can be used to obtain draws from $\mathbb{P}(\mathfrak{T}_1^{\mathcal{L}}, \dots, \mathfrak{T}_m^{\mathcal{L}}, \sigma^2, \gamma \mid \mathbf{y}, \mathbf{X})$, the posterior distribution of the model parameters. As with the original sampling scheme for **BART**, **HBART** also relies on “Bayesian backfitting” (Hastie and Tibshirani, 2000) to fit each tree iteratively, holding all other $m - 1$ trees constant. This is achieved by considering the residual response when updating the j^{th} tree $\mathbf{R}_j := \mathbf{y} - \sum_{t \neq j} \mathfrak{T}_t^{\mathcal{L}}(\mathbf{X})$.

The Gibbs sampler for **HBART** is similar to that of **BART** and we briefly review here. The Gibbs sampler for **HBART** proceeds by first proposing a change to the first tree’s structure \mathfrak{T}_1 which are accepted or rejected via a Metropolis-Hastings step (Hastings, 1970). Tree structures are altered by introducing small changes: growing a terminal node by adding two terminal daughter nodes, pruning two terminal daughter nodes,

or changing a split rule. Given the tree structure, samples from the posterior of the b leaf parameters $\varnothing_1 := \{\mu_1, \dots, \mu_b\}$ are then drawn from the conjugate-normal posterior distribution. This procedure proceeds iteratively for each tree, using an updated set of partial residuals \mathbf{R}_j .

Once each tree structure and leaf values has been updated, a draw from the posterior of σ^2 conditional on all other parameters is made based on the full model residuals $\mathcal{E} := \mathbf{y} - \sum_{t=1}^m \mathfrak{F}_t^{\varnothing}(\mathbf{X})$. Finally, a draw from the posterior of γ conditional the other parameters is computed via a Metropolis-Hastings step.

The steps of the full HBART procedure are illustrated below:

$$\begin{aligned}
1 : \quad \mathfrak{F}_1 & \mid R_{-1}, \sigma^2, \gamma \\
2 : \quad \varnothing_1 & \mid \mathfrak{F}_1, R_{-1}, \sigma^2, \gamma \\
& \vdots \\
2m - 1 : \quad \mathfrak{F}_m & \mid R_{-m}, \sigma^2, \gamma \\
2m : \quad \varnothing_m & \mid \mathfrak{F}_m, R_{-m}, \sigma^2, \gamma \\
2m + 1 : \quad \sigma^2 & \mid \mathfrak{F}_1^{\varnothing}, \dots, \mathfrak{F}_m^{\varnothing}, \gamma, \mathcal{E} \\
2m + 2 : \quad \gamma & \mid \sigma^2, \mathcal{E}
\end{aligned} \tag{5.5}$$

All $2m + 2$ steps represent a *single* Gibbs iteration¹³. After a sufficient burn-in period, burned-in draws from the posterior of f are obtained. As with BART, a point prediction $\hat{f}(\mathbf{x})$ for HBART can be obtained by taking the average of the burned-in values of $\mathfrak{F}_1^{\varnothing}, \dots, \mathfrak{F}_m^{\varnothing}$ evaluated at a given \mathbf{x} ; posterior credible intervals for f are computed by using the quantiles of the burned-in values. Posterior predictive intervals

¹³BART relies on a similar scheme, removing the conditioning on γ at each step and not requiring step $2m + 2$.

at a given \mathbf{x} are computed as follows:

1. Draw $f(\mathbf{x})$ via

i) drawing one of the burned-in sum-of-trees collections $\mathfrak{T}_1^{\text{leaf}}, \dots, \mathfrak{T}_m^{\text{leaf}}$ and

ii) computing $f(\mathbf{x}) = \sum_{i=1}^m \mathfrak{T}_i^{\text{leaf}}(\mathbf{x})$.

2. Draw $\sigma^2(\mathbf{x})$ via

i) obtaining γ and σ^2 from the same Gibbs sample from which the sum-of-trees was obtained,

ii) determining the \mathbf{z} which corresponds to the \mathbf{x} of interest and

iii) computing $\sigma^2(\mathbf{x})$ by evaluating γ , σ^2 and \mathbf{z} in the exponentiation of Equation 5.3.

3. Take one draw from $\mathcal{N}(f(\mathbf{x}), \sigma^2(\mathbf{x}))$ which is the BART model (Equation 5.1)

4. Collect draws from step 3 by repeating steps 1–3 many times. Then, return the desired quantiles.

Note that HBART requires modifications to the original BART likelihood calculations necessary for Metropolis-Hastings steps to alter the tree structures (steps 1, 3, \dots , $2m - 1$ of Equation 5.5). Also, the posterior distributions for the leaf parameters must be updated (steps 2, 4, \dots , $2m$ of Equation 5.5). It is worth noting that these modifications are valid for any heteroskedastic BART model and not just the log-linear HBART model proposed in this work, as they computed as functions of $\sigma_1^2, \dots, \sigma_n^2$. Additionally, modifications to the posterior distribution of σ^2 (step $2m + 1$ of Equation 5.5) is required. Finally, the sample of γ (step $2m + 2$ of Equation 5.5) is obtained using the Metropolis-Hastings procedure with the proposal distribution

outlined in Gamerman (1997). We provide explicit computational details for each of these steps in Appendix A.3.

5.4 Simulations

5.4.1 Univariate Model

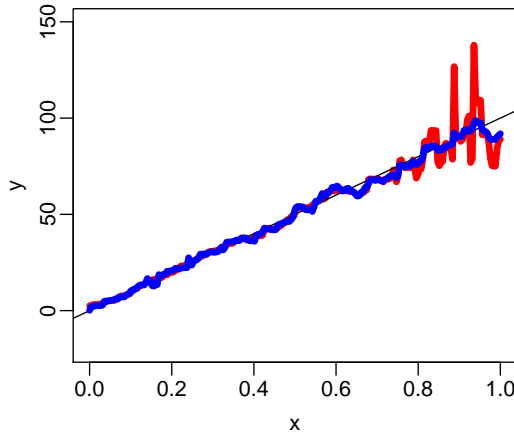
We begin by comparing the performance of **BART** versus **HBART** in a simple univariate setting. Consider a single predictor \mathbf{x} which is a uniformly spaced sequence of n points on $[0, 1]$. Then, we consider two models. The first is heteroskedastic and is given by

$$Y_i = 100x_i + \mathcal{E}_i, \quad \mathcal{E}_i \stackrel{\text{ind}}{\sim} \mathcal{N}(0, \sigma_i^2), \quad \sigma_i^2 = \exp(7x_i). \quad (5.6)$$

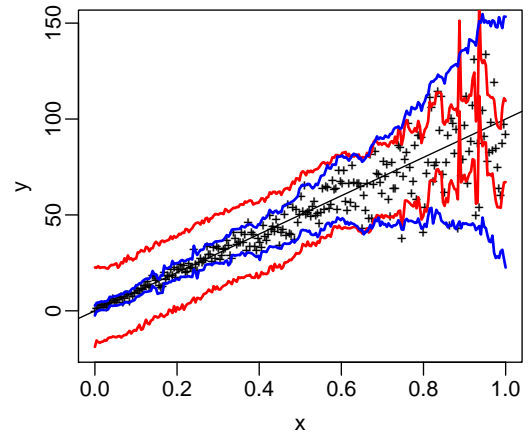
The second is homoskedastic and is given by

$$Y_i = 100x_i + \mathcal{E}_i, \quad \mathcal{E}_i \stackrel{\text{ind}}{\sim} \mathcal{N}(0, 5^2). \quad (5.7)$$

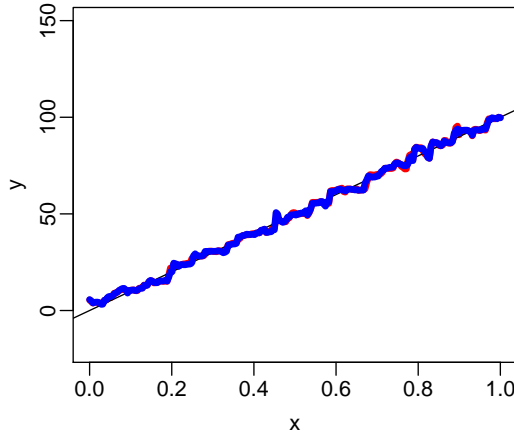
For **HBART**, the matrix of covariates for the parametric variance model will be set to $\mathbf{Z} = [\mathbf{x}]$, our one uniformly spaced covariate. Parenthetically, note that the default in our software implementation is to set $\mathbf{Z} = \mathbf{X}$. Therefore, **HBART** will have a correctly specified variance function for the model given in Equation 5.6. We include the homoskedastic model as well to benchmark **HBART**'s performance to determine if the unnecessary extra complexity of the variance model degrades the algorithm's performance.



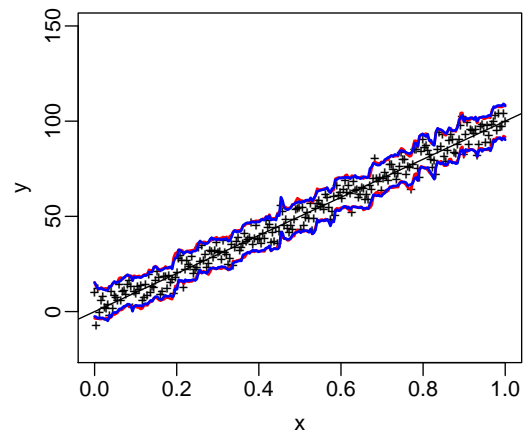
(a) Heteroskedastic Model,
Posterior Mean Estimates



(b) Heteroskedastic Model,
Posterior Predictive Intervals



(c) Homoskedastic Model,
Posterior Mean Estimates



(d) Homoskedastic Model,
Posterior Predictive Intervals

Figure 5.1: BART's and HBART's posterior mean estimates and 90% posterior predictive intervals for each algorithm built from a sample of $n = 250$ observations drawn from the processes in Equation 5.6 (a and b) and Equation 5.7 (c and d). **Red** lines correspond to the results of a BART model and **blue** lines correspond to the results from a HBART model. The black lines in (b) and (d) correspond to the true conditional mean function ($Y_i = 100x_i$) and the black '+'s represent the actual observations.

Figure 5.1 compares **BART** to **HBART** in both the heteroskedastic and homoskedastic simulated models by gauging two metrics: accuracy of estimation of f and appropriateness of posterior predictive intervals.

Figure 5.1a highlights the posterior means \hat{f} estimated by the two different models. Note that **HBART** provides a better estimate of the true f when $x > 0.7$, the region of relatively high variance. By estimating large variance in this region, it has the flexibility to downweight these high variance observations, allowing for more shrinkage towards the global average and away from the noisy local (within-node) sample mean. In contrast, **BART** overfits in this region. By assuming homoskedasticity, the algorithm is handicapped, and is obligated to move its mean function up when the noise term is large and positive and down when the noise term is large and negative. Note that both algorithms perform well when $x < 0.7$, where the data has lower variance.

Figure 5.1b provides 90% posterior predictive intervals for future observations (as explained in the procedure outlined in Section 5.3.2). Given the homoskedasticity assumption of **BART**, the prediction intervals at each x -location are of constant width. This implies that the intervals are too wide at low values of x and too narrow at higher values of x . In contrast, **HBART** provides more appropriate prediction intervals, narrow at low x and wide at high x , thus correctly reflecting the heteroskedasticity in the underlying data-generating model. Although not the primary focus of this paper, examining the burned in γ values yielded a 90% credible interval of $[6.52, 7.65]$, which captures 7, the value of the linear coefficient in the log-linear variance model of Equation 5.6.

For the homoskedastic model, Figures 5.1c and 5.1d highlight that **BART** and **HBART** yield virtually identical results in terms of mean function estimation and predictive intervals. Thus, **HBART** seems to be robust in the absence of heteroskedasticity for this illustration.

5.4.2 Multivariate Model

We now consider the following data generating process, similar to the model simulated in Cepeda and Gamerman (2001):

$$Y_i = f(\mathbf{x}_i) + \mathcal{E}_i, \quad f(\mathbf{x}_i) = -35 + .35x_{1,i} - 1.7x_{2,i}, \quad \mathcal{E}_i \stackrel{iid}{\sim} \mathcal{N}(0, \sigma_i^2) \quad (5.8)$$

$$x_{1,i} \stackrel{iid}{\sim} \text{U}(0, 400), \quad x_{2,i} \stackrel{iid}{\sim} \text{U}(10, 23), \quad x_{3,i} \stackrel{iid}{\sim} \text{U}(0, 10)$$

$$\sigma_i^2 = \exp(-6 + .03x_{1,i} + .4x_{3,i}) \quad (5.9)$$

We set the number of observations to be $n = 500$ and then we fit an **HBART** model using the default $\mathbf{Z} = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3]$. Thus, the variance model is misspecified (the true model is not a function of \mathbf{x}_2). We again compare **HBART**'s performance to that of **BART**.

Figure 5.2 plots actual values of the conditional mean function versus fitted values. This illustration demonstrates that in areas of high variance, **BART** has difficulty separating the mean function from the noise, and as a result, provides wide credible intervals for the true f . **HBART**, on the other hand, provides more narrow credible intervals, indicating that the algorithm was able to successfully separate the mean function from the heteroskedastic variance structure in the data generating process.

We next evaluate the performance of **HBART** versus **BART** in terms of out-of-sample predictive performance. We consider two models: one with homoskedastic errors, $\mathcal{E} \stackrel{iid}{\sim} \mathcal{N}(0, 3^2)$, and one with heteroskedastic error structure according to Equation 5.9. For each error structure, we then generate $n = 500$ training observations based on the data generating process outlined in Equation 5.8. Both **HBART** and **BART** models are constructed on the training set and performance is evaluated in terms of root mean square error (RMSE) on an additional $n = 500$ independent test observations drawn

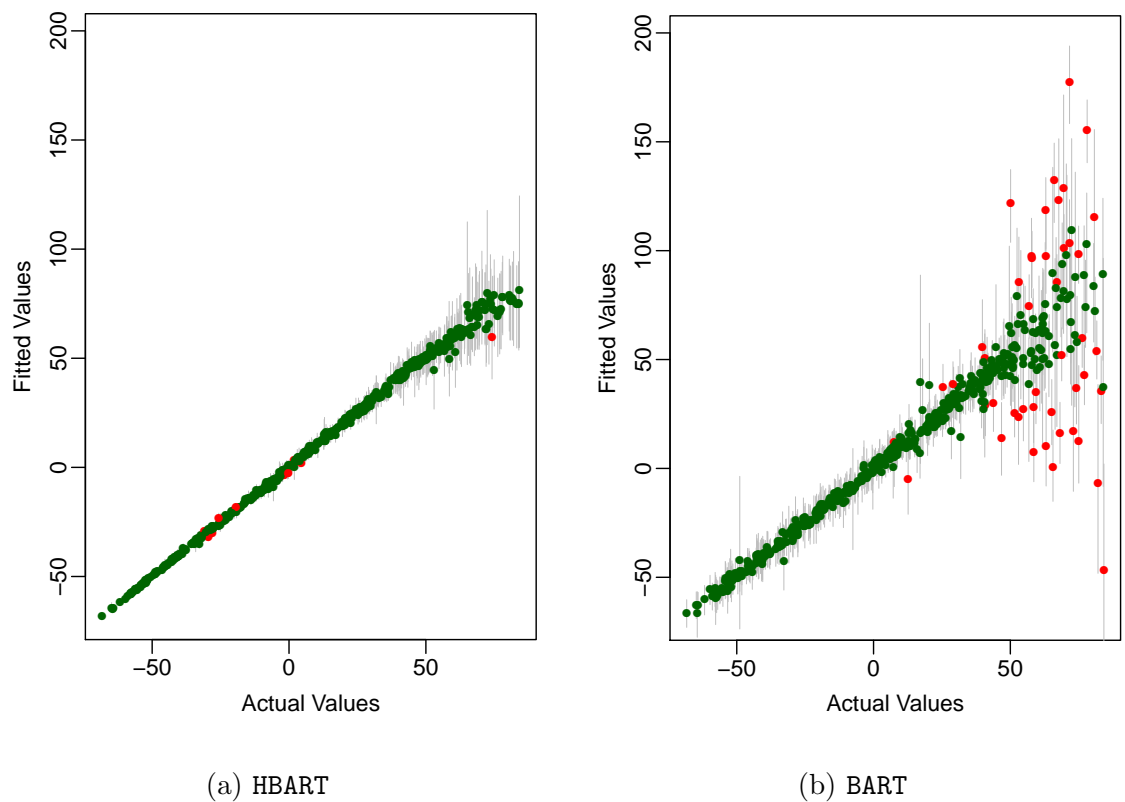


Figure 5.2: Estimates of the conditional mean function f for HBART and BART with associated 90% credible intervals. The x-axis is the true value of the conditional mean function and the y-value is the model estimate. Gray lines illustrate 90% credible intervals. If the true conditional mean falls within the interval, the point is colored **green** and points in **red** signify the true value falls outside of the interval.

from the same data generating process.

Figure 5.3 displays the out-of-sample performance results for 100 simulations. For the heteroskedastic model, HBART significantly outperforms BART (Figure 5.3a). For the homoskedastic model, the performance of BART and HBART are statistically equal and the results of both algorithms are quite stable (Figure 5.3b). This plot provides more evidence of the robustness of HBART's performance in the absence of heteroskedasticity.

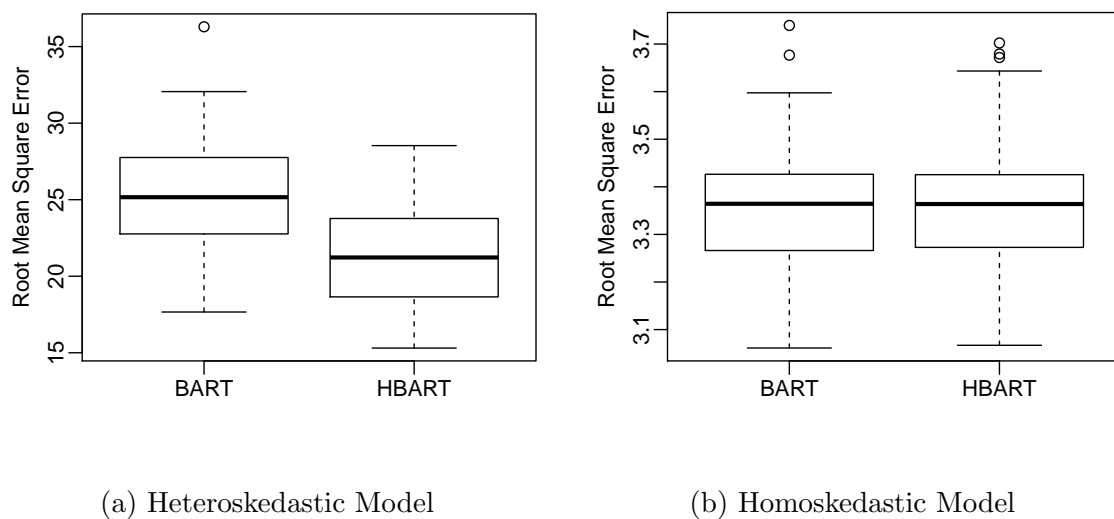


Figure 5.3: Distribution of out-of-sample RMSE for BART and HBART fit in 100 simulations for (a) the heteroskedastic data generating process of Equations 5.8 and 5.9 and (b) the homoskedastic data generating process of Equation 5.8 with $\mathcal{E} \stackrel{iid}{\sim} \mathcal{N}(0, 3^2)$. The models are built on 500 observations and tested on 500 independent observations.

5.5 Real Data Examples

5.5.1 Lidar Data

We consider the LIDAR data set explored in Ruppert et al. (2003). The data set contains 221 observations. The response denoted $\text{Log}(\text{Ratio})$ is the logarithm of the ratio of reflected laser-emitted light from two sources and the predictor denoted Range is the distance traveled before the light is reflected back to its source. Leslie et al. (2007) explored this data set by fitting both nonparametric mean and variance functions to the data under the assumptions of both normal and non-normal error distributions.

We fit the data using BART and HBART. For HBART, \mathbf{Z} is taken to be $[\text{Range}, \text{Range}^2]$ where the two columns are orthogonalized. Figure 5.4 illustrates the posterior mean estimates for both HBART and BART. One will notice that both algorithms estimate relatively similar posterior mean functions with HBART's estimation being slightly more smooth in the region of higher variance than that of BART. Figure 5.1b shows 90% posterior predictive intervals for the two algorithms. The intervals for HBART seem to appropriately reflect the heteroskedasticity in the data, while BART's predictive intervals are too wide at low values of Range and too narrow at high values of Range .

5.5.2 Motorcycle Data

We next consider a dataset of simulated motorcycle crashes that was compiled by Schmidt et al. (1981). The observations in the data set consist of accelerometer readings (acceleration) taken from riders' helmets at 133 different points in time (time) after a simulated impact. As discussed in Gramacy (2007), many researchers find that this dataset exhibits multiple regimes in both the mean function and variance

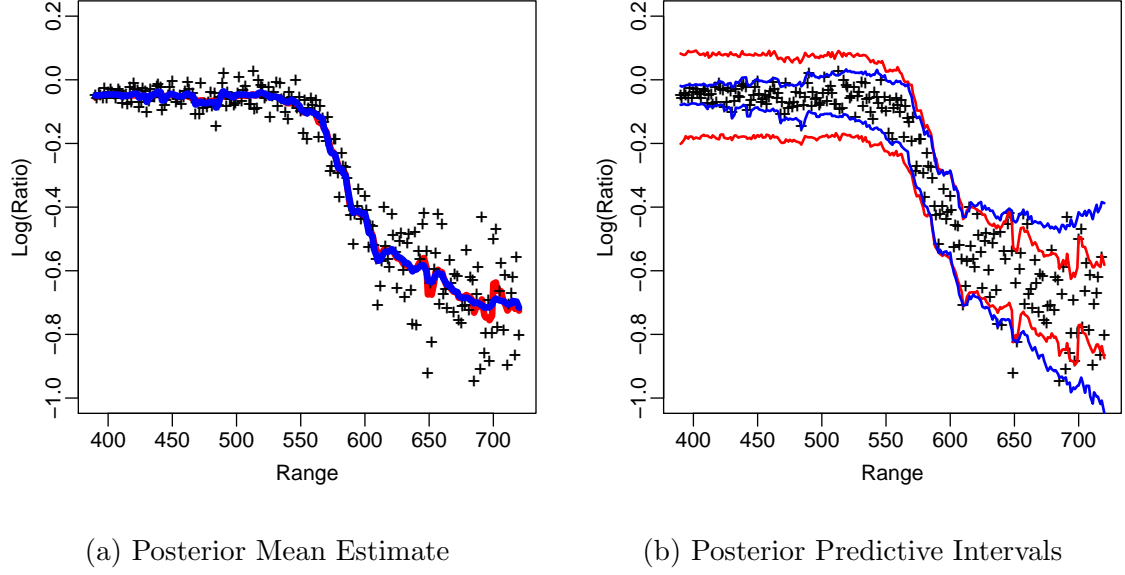


Figure 5.4: BART’s and HBART’s posterior mean estimates and 90% posterior predictive intervals for the Lidar data. **Red** lines correspond to the BART model and **blue** lines correspond to the HBART model.

function over time.

This dataset was also explored by Taddy et al. (2011) who remarked that the 90% posterior predictive interval for BART appeared to “variously over or under estimate data uncertainty around the regression mean. In particular, BART’s global variance term is misspecified for this heteroskedastic data.” We attempt to remedy this problem with HBART. Exploring a scatterplot of the data, the model seems to be characterized by a low variance regime followed by a high variance regime and then an additional low variance regime (see Figure 5.5). Hence, when building an HBART model, we do not use the default \mathbf{Z} . To capture the low-high-low variance relationship, we specify the model to be quadratic in the predictor, $\mathbf{Z} = [\text{time}, \text{time}^2]$ where the two columns are orthogonalized.

Figure 5.5 displays 90% posterior predictive intervals for HBART, BART, dynamic

trees (`dynaTree`, Taddy et al., 2011), and treed Gaussian processes (`tgp`, Gramacy, 2007). Each of the algorithms has a similar estimate of the posterior mean process (unshown), but there are some differences in the posterior predictive intervals. `BART`'s predictive intervals are much too wide at low and high values of time and perhaps too narrow for the intermediate values of time. However, `HBART`'s intervals are quite similar to those of `dynaTree`, being widest for intermediate values of time and more narrow near the beginning and end. Interestingly, `HBART` is the only of the four models that builds a narrower predictive interval at the higher values of time.

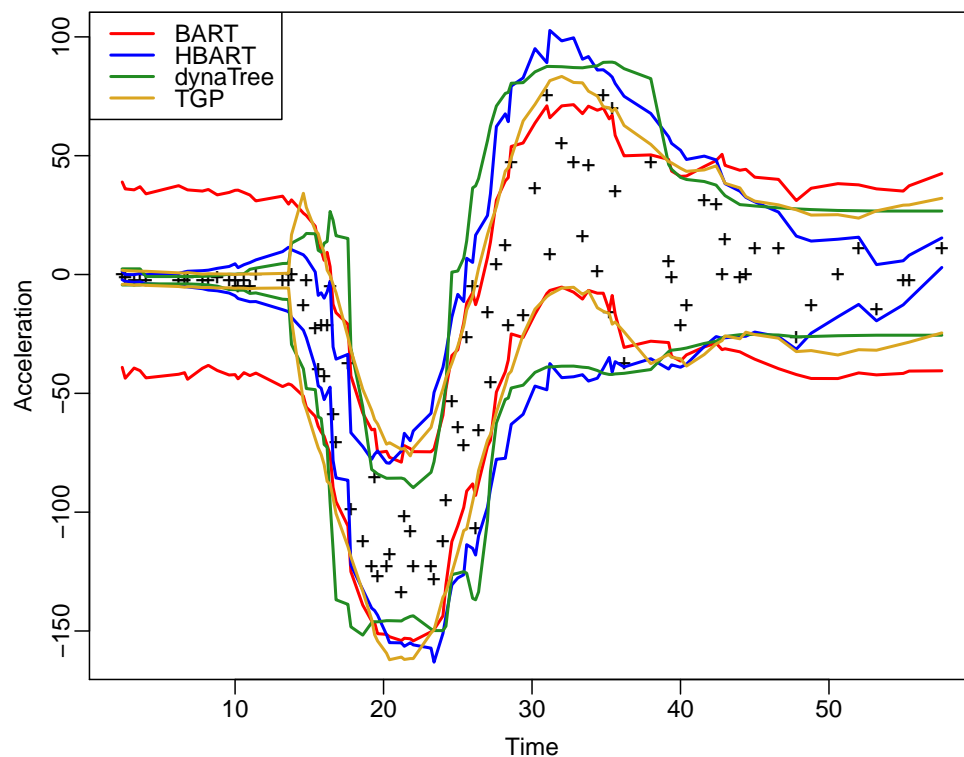


Figure 5.5: 90% posterior predictive intervals for `BART`, `HBART`, `dynaTree` and `TGP`.

5.6 Discussion

We have proposed **HBART**, an extension of **BART** that relaxes the assumption of homoskedasticity in the model errors. In particular, we have developed a model that allows for a multiplicative heteroskedastic error structure, where the multiplicative factor is the exponential of a parametric function of some set of covariates.

Through simulations and explorations of real data, we have demonstrated **HBART**'s potential for generating more appropriate posterior predictive intervals in the presence of appropriately modeled heteroskedastic data versus **BART**. Additionally, **BART** suffers from overfitting in areas of very high variance and **HBART** seems to offer promise in ameliorating this issue. In our explorations, the added complexity of fitting a model to the error terms did not hinder **HBART**'s performance on homoskedastic data. In this situation, **HBART**'s estimates of the posterior means, predictive intervals, and out-of-sample RMSE were very similar to **BART**'s.

Originally proposed in Chipman et al. (2010) and implemented in Kapelner and Bleich (2015), it is possible to cross-validate over a number of the hyperparameters of the **BART** model. Future work will involve extending **BART-CV** to **HBART-CV**, where it is possible to cross-validate over a selection of prior variances for the parametric variance model to impose varying degrees of shrinkage by modifying the Σ_{jj} hyperparameters. One final direction of research is to further relax the assumptions on the error structure. For instance, one could incorporate more flexible variance models such as smoothing splines instead of the standard linear model or relax the assumption of normality of the errors by considering Dirichlet mixture priors.

Citation

Bleich, J. and Kapelner, A. (2015). BART with parametric models of heteroskedasticity. *arXiv:1402.5397*.

Ensemble-of-Trees Algorithms in Criminology

Abstract

There is a substantial and powerful literature in statistics and computer science clearly demonstrating that modern machine learning procedures can forecast more accurately than conventional parametric statistical models such as logistic regression. Yet, several recent studies have claimed that for criminal justice applications, forecasting accuracy is about the same. In this chapter, we address the apparent contradiction. Forecasting accuracy will depend on the complexity of the decision boundary. When that boundary is simple, most forecasting tools will have similar accuracy. When that boundary is complex, procedures such as machine learning, that proceed adaptively from the data will improve forecasting accuracy, sometimes dramatically. The complexity of the decision boundary will in practice be unknown, and there can be substantial risks to gambling on simplicity. Criminal justice decision makers and other stakeholders can be seriously misled with rippling effects going well beyond the immediate offender. There seems to be no reason for continuing to rely on traditional forecasting tools such as logistic regression.

6.1 Introduction

Forecasts of recidivism have been widely used in the United States to inform parole decisions since the 1920s (Burgess, 1928; Borden, 1928). Of late, such forecasts are being proposed for a much wider range of criminal justice decisions. One important example is recent calls for predictions of “future dangerousness” to help shape sentencing (Pew Center of the States, 2011; Casey et al., 2011). The recommendations build on related risk assessment tools already operational in many jurisdictions, some mandated by legislation (Kleiman et al., 2007; Turner et al., 2009; Hyatt et al., 2011; Skeem and Monahan, 2011; Oregon Youth Authority, 2011).

With such widespread enthusiasm and very high stakes, one might assume forecasting accuracy has been properly evaluated and determined to be good. In fact, competent evaluations can be difficult to find for a wide variety of criminal justice decisions. Some of the problems have a long history (Ohlin and Duncan, 1949; Ohlin and Lawrence, 1952; Reiss Jr, 1951). For example, it is relatively rare for evaluations to be based on “test data” that were not used to construct the forecasting procedures. The danger is grossly overoptimistic assessments. More recent commentaries have documented a number of other problems, sometimes including no evaluation at all (Farrington and Tarling, 1985; Gottfredson and Moriarty, 2006; Berk, 2007, 2012)

The need for thorough and thoughtful evaluations has become even more important over the past decade because in addition to calls for a more routine use of crime forecasts, new forecasting tools from computer science and statistics have been developed. Often supported by formal proofs, simulations, and comparative applications across many different data sets, these tools promise improved accuracy in principle (Breiman et al., 1984; Breiman, 1996, 2001a; Vapnik, 1998; Friedman, 2002; Chipman et al., 2010), including several instructive criminal justice applications in print as well

(Berk, 2012).

Yet, there are also several recent articles claiming that for criminal justice applications, the new tools perform no better than the old tools (Yang et al., 2010; Liu et al., 2011; Tollenaar and Van der Heijden, 2013). Logistic regression is a favorite conventional approach. The conclusion seems to be “why bother?” For criminal justice forecasting applications, the new procedures are mostly hype.

“The conclusion is that using selected modern statistical, data mining and machine learning models provides no real advantage over logistic regression and LDA.¹⁴ If variables are suitably transformed and included in the model, there seems to be no additional predictive performance by searching for intricate interactions and/or non-linear relationships” (Tollenaar and Van der Heijden, 2013).

How can the proofs, simulations and many applications provided by statisticians and computer scientists be so wrong? How can it be that statistical procedures being rapidly adopted by private firms such as Google and Microsoft and by government agencies such as the Department of Homeland security and the Federal Bureau of Investigation are no better than regression methods readily available for over fifty years? Why would the kinds of new analysis procedures being developed for analyzing a variety of datasets with hundreds of thousands of cases (Dumbill, 2013; National Research Council, 2013) not be especially effective for a criminal justice dataset of similar size?

A careful reading of the technical literature and recent criminal justice applications suggests that there can be a substantial disconnect between that technical literature and the applications favored by many criminal justice researchers. Statisticians and computer scientists sometimes do not distinguish between forecasting performance in

¹⁴“LDA” stands for linear discriminant analysis.

principle and forecasting performance in practice. Criminal justice researchers too often proceed as if the new procedures are just minor revisions of the generalized linear model. In fact, the conceptual framework and actual procedures can be very different and require a substantial change in data analysis craft lore. Without a proper appreciation of how the new methods differ from the old, there can be serious operational and interpretative mistakes.

In this chapter, we try to improve the scientific discourse by providing an accessible discussion of some especially visible, modern forecasting tools that can usefully inform criminal justice decision-making. The discussion is an introduction to material addressed far more deeply in Berk (2012). We also try to provide honest, apples-to-apples performance comparisons between the newer forecasting methods and more traditional approaches.

For some readers, it may be useful to make clear what this discussion is not about. As one would expect, there have been jurisprudential concerns about “actuarial methods” dating from at least the time when sentencing guidelines first became popular (Messinger and Berk, 1987; Feeley and Simon, 1994), and more recent discussions about the role of race have introduced an important overlay (Harcourt, 2008; Berk, 2009). The issues are difficult and real. They are also not addressed here. Our concerns are more immediate. Forecasts of future dangerousness are being developed and used. Real decisions are being made affecting real people. At the very least, those decision should be informed by the best information available. And that information depends significantly on the forecasting procedures deployed.

6.2 Proper Criminal Justice Forecasting Comparisons

The conceptual foundation for criminal justice forecasting can easily be misconstrued (Ridgeway, 2013). We begin, therefore, with a fundamental conceptual point that some readers may at first find counterintuitive. As a formal matter, one does not have to understand the future to forecast it with useful accuracy. Accurate forecasting requires that the future be substantially like the past. If this holds, and one has an accurate *description* of the past, one has an accurate forecast of the future. That description does not have to explain why the future takes a particular form and certainly does not require a causal interpretation. Readers comfortable with traditional time series analysis (Box and Jenkins, 1970), should have no problem with this reasoning.

It follows that there is a key distinction between forecasting and explanation that has been badly conflated in some accounts (Andrews et al., 2006). Understanding a phenomena may lead to improved forecasting accuracy, or it may not, but forecasting and explanation are different enterprises that can work at cross-purposes. For example, explanatory models should be relatively simple and provide instructive interpretations. Such models can leave out a large number of weak predictors that one-by-one do not enlighten but *in the aggregate* dramatically improve forecasting accuracy. Common practice implicitly folds such variables into the disturbance term. Alternatively, such predictors, often called “nuisance variables” in limited information structural models, are associated “nuisance parameters” and given “minimal attention” (Cameron and Trivedi, 2005). Similar issues arise if simple, easily interpretable functional forms (e.g., linear) are used when complex functional forms might fit the data somewhat better.

The approach we take is to maximize forecasting accuracy, and that is the premise on which the underlying mathematics depend. We take this approach because it leads to clear performance criteria and various proofs of optimal forecasting accuracy for a given dataset. Such clarity is an undeniable virtue about which more will be said shortly.

Equally important, there are a wide variety of decisions made by criminal justice officials in which a *necessary condition* is the best possible forecasting accuracy. Consider a judge's decision to sentence an offender to either incarceration or probation. A recent Pennsylvania statute states that a "risk assessment instrument may be used as an aide in evaluating the relative risk that an offender will reoffend and be a threat to public safety." Presumably, accuracy really matters. Imagine the ethical and legal implications of using a particular risk tool to justify a long incarceration when there exist more accurate risk tools from which a sentence of probation could be more appropriate. There is also no requirement in the legislation that a judge understand why an individual is high or low risk. Indeed, it is not even clear what a judge would do with such information.¹⁵ Other examples, include pre-trial decisions to release defendants on bail or decisions by parole boards to release under supervision inmates who have not served their full terms. One could also imagine forecasts of future dangerousness helping to determine charging decisions by prosecutors.

Thus, there is no formal concern in this discussion with why certain predictors improve forecasting accuracy and no attempt to interpret them as explanations for the forecasted behavior. For example, if other things equal, shoe size is a useful predictor of recidivism, it can be included as a predictor. Why shoe size matters is immaterial. In short, we are not seeking to identify risk factors that may or may not

¹⁵In the special case when there are clear indications of substance dependency or psychological problems, a judge might order treatment along with the sentence. But such conditions are not necessarily risk factors for many kinds of crime, and indications of need can be sufficient.

make any subject-matter sense. That can be a useful enterprise, but it is a different enterprise.

Indeed, if the enterprise really is explanation, than some form of structural equation modeling may be called for. There is an extensive and largely unrebutted literature highly critical of structural equation modeling in general. An excellent, accessible, and technically sound treatment can be found in Freedman (2009). We cannot rehash the issues here except to stress that machine learning is not a form of structural equation modeling and should never be interpreted as such.¹⁶ Moreover, if the goal is to use one or more risk factors to design and test interventions, many would argue that the only sound approach is randomized experiments or very strong quasi-experiments.

6.2.1 Some Common-Sense Requirements for Fair Forecasting Comparisons

If one intends to compare the forecasting performance of different forecasting tools, there are several basic, common-sense requirements. These provide the ground rules.

- (a) One must be clear on what features of forecasting procedures are being compared. As we explain below, “black box” forecasting methods may forecast with remarkable accuracy and provide decision makers with tools that can be enormously helpful (Breiman, 2001b). But black box forecasting methods may have little to say about which risk factors matter most. If the goal is to compare different procedures by their forecasting accuracy, forecasting accuracy should be the benchmark.

¹⁶A structural equation model is an algebraic theory of how nature generated the data and as such, can be right or wrong. Machine learning employs algorithms that seek some well-defined empirical goal, such as maximizing forecasting accuracy. There is no structural model. Concerns about whether the model is correct are irrelevant. What matters is how well the algorithm performs.

- (b) Forecasting comparisons must be based on data not used to construct the competing forecasting procedures. Such data are often called “test data,” and accuracy is often called “out-of-sample performance.” Data used to build the forecasting procedures can be called “training data.” If training data are also used as test data, all comparisons risk contamination through overfitting (Hastie et al., 2009, p. 219-226). As already noted, this point has been appreciated for well over 50 years, but is often ignored.
- (c) Proper performance criteria must be used that are the same across competing methods. For example, measures of fit are not appropriate if the competition claims to be testing forecasting accuracy. In addition, there are many different measures of forecasting performance (Hastie et al., 2009, chapter 7), and the same measure should be used for all of the competitors. For example, the area under a receiver operating characteristic curve (ROC) provides very different information from that available through direct estimates of generalization (forecasting) error (Hastie et al., 2009, p. 314-317).
- (d) All of the forecasting competitors should be accurately characterized if comparisons are to be properly understood. For example, there are a number of forecasting procedures represented as state-of-the-art that actually are not. There are also forecasting procedures characterized as machine learning that actually are not. Classification trees, for instance, is neither state-of-the-art nor a machine learning technique. AdaBoost (Freund, 1997) is a machine learning procedure, but was state-of-the-art 15 years ago. Algorithms such as RF and SGB are machine learning procedures and state-of-the-art.¹⁷

¹⁷What qualifies as state-of-the-art can certainly be debated, but within sensible boundaries, there can be remarkable consensus. For example, RF is certainly not the newest machine learning procedure, but for a wide range applications nothing else seems to consistently perform better. Likewise, sharp distinctions between machine learning, statistical learning and a variety of other related

- (e) Many of the popular forecasting procedures have tuning parameters that researchers can use to improve forecasting accuracy.¹⁸ In addition, sometimes researchers do not understand that in their effort to maximize forecasting accuracy they are implicitly tuning their procedure. Fair comparisons require that all competitors are tuned in a comparable fashion. This can be difficult because the tuning is often based on principles that can depend on the particular forecasting procedure being used.
- (f) All forecasting competitions are necessarily data dependent and can vary across different applications. Forecasting competitions do not reveal fundamental and invariant forecasting truths. To take a simple example, a procedure that performs poorly in small samples may be a star in large samples because its best properties only materialize asymptotically. Appropriate caveats should be attached to the results of all forecasting comparisons.
- (g) Performance differences across different forecasting procedures must be thoughtfully evaluated. This will often mean a careful consideration of *how a forecasting procedure will be used*. A small difference in forecasting accuracy can translate into a difference of hundreds of crimes. Academic researchers may not care. But stakeholders surely do. There is also the equally important matter of taking uncertainty into account. Some apparent differences wash out in new realizations of the data. They are just chance artifacts.

procedures are increasingly difficult to defend and probably not worth quarreling over (National Research Council, 2013). Nevertheless, within somewhat fuzzy boundaries, there can be widespread agreement.

¹⁸In the estimation of a logistic regression, for instance, the convergence threshold of the iteratively reweighted least squares algorithm is a tuning parameter. It needs to be small enough to produce a close approximation to a maximum likelihood estimate, but not so small that unnecessary iterations are performed. Another example is a decision in stepwise regression to fix the number of predictors that can be included in the final model. In forecasting settings, tuning parameters usually are chosen in service of forecasting accuracy.

- (h) It should go without saying, but all of the forecasting procedures must be implemented correctly. There is ample evidence that too often this is not the case (Berk, 2012).

6.3 Some Conceptual Fundamentals

We turn now to a conceptual overview of classification and forecasting. The intent is to provide a very accessible, didactic overview that can apply to a very broad range of forecasting procedures used previously in criminal justice applications. Readers interested in a technical discussion should consult the references cited.

Consider the decision of whether or not to release an individual on parole. Since the 1920's, such decisions have often been informed by forecasts of whether a given inmate will be arrested for a new crime soon after release. The forecasts are shaped by actuarial procedures applied to information from inmates who had been released in the past. In effect, profiles are developed that can classify inmates by whether they succeeded or failed on parole. These profiles are used to forecast parole outcomes when they are not yet known. In the next few pages, we provide a basic, nontechnical overview of how this can be done. We build on a prior treatment written for criminal justice researchers (Berk, 2012) and on more formal textbook discussions as needed (Bishop, 2006; Hastie et al., 2009).

6.3.1 The Basic Account

Figure 6.1 is a very simplified and initial plot illustrating how classification and forecasting can be undertaken. The red circles represent individuals who have failed on parole in the past. The blue circles represent individuals who have succeeded on

parole in the past. There are two predictors in this illustration. One predictor is the number of prior arrests. The other predictor is the number of rule infractions during the most recent incarceration. Both can be seen as “dynamic” predictors, but “static” predictors would have not materially changed the discussion. Figure 6.1 can be seen as a 3-dimensional scatterplot.¹⁹

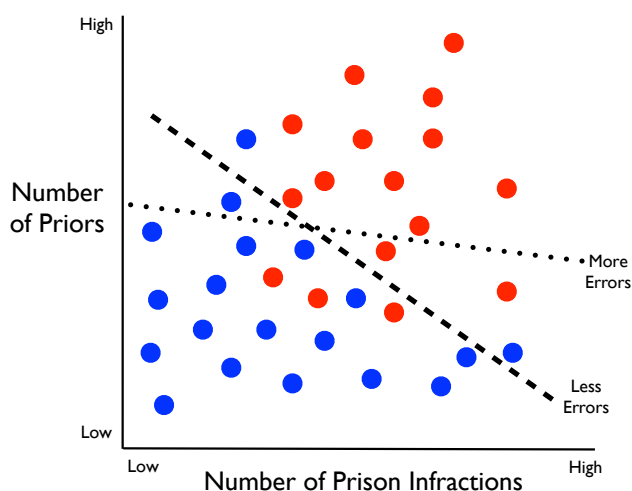


Figure 6.1: Two linear decision boundaries in 2-dimensional predictor space.

The statistical task is to impose a “decision boundary” on the 2-dimensional predictor space that can be used to define two classes: those who fail and those who do not. The term “decision boundary” is used because the intent is to directly inform actual decisions.²⁰ Statistical procedures that partition the data into different

¹⁹The meanings of “dynamic predictors” and “static predictors” can depend on the context and the decision to be informed by the forecast. For example, the difference between static and dynamic predictors plays a key role in the fairness of parole decisions. Is it appropriate to use static predictors already employed at sentencing when later parole decisions are made? Is there a risk of unfair “double counting”? Thus, the crime that sent an individual to prison is static. Should it be also used to help inform parole decisions? In contrast, time in a prison secure housing unit (SHU) is in this context dynamic. There would be no concerns about double counting if it were employed by a parole board.

²⁰The underlying mathematics is shaped by the same goal.

grouping are often called “classifiers.” In this instance, the partitioning should result in the fewest classification errors possible. For Figure 6.1, there will necessarily be two regions defined, one for failures and one for successes. Ideally, the failure region has no successes, and the success region has no failures. Usually, one has to settle for less.

The dotted line is one possible linear decision boundary. In the region above the dotted line, failures predominate by a count of 13 to 2. So, that region is assigned the class of “failure.” In the region below the dotted line, successes predominate by a count of 17 to 5. So, that region is assigned the class of “success.”

The assigned classes can be used for forecasting. When a new case is found for which a forecast is needed, that case is placed in one region or the other depending on its values for the two predictors. For example, a case with a very large number of priors and a very large number of prison infractions would be placed in the “failure” region to the upper right, and a forecast of failure would be made. A decision to impose a stiff prison sentence could follow.

The dotted decision boundary results in several classification errors. There are 2 (blue) successes classified as failures, and 5 five (red) failures classified as successes. Overall, there are 7 errors for 35 cases, which means that the classification procedure is right about 80% of the time. In real applications, this would be considered very good performance.

The dashed line is another attempt to accurately separate the successes from the failures. Above this alternative linear decision boundary, the majority of cases once again are failures. Therefore, the class of “failure” is assigned to that region of the figure. Below the alternative linear decision boundary, the majority of cases are successes. Therefore, the class of “success” is assigned to that region of the figure. Now there are only five misclassified cases: 2 blue circles are in the red region and 3

red circles are in the blue region. The new boundary produces correct classifications about 85% of the time, and on those grounds is likely to be preferred to the old boundary.

As before, any cases with predictor values that place them above the decision boundary, but whose outcomes are not yet known, are forecasted to be failures. Similarly, any cases with predictor values that place them below the decision boundary, but whose outcomes are not known, are forecasted to successes. From a classification exercise comes a forecasting procedure. The forecasts, in turn, are used to inform parole decisions.

How might one arrive at the best linear decision boundary? If the two outcomes are coded as 1 or 0, and conventional linear regression is applied using the two predictors as regressors, one important kind of optimal linear decision boundary can be imposed on the predictor space. That line is defined by fitted values of .50. Cases with regression fitted values greater than .50 are assigned one class and cases with regression fitted values equal to or less than .50 are assigned the other class. By minimizing the sum of squared residuals and imposing a fitted value threshold at .5, one is also minimizing the sum of the classification errors (Hastie et al., 2009).

When the response is represented as the log of the odds of the category coded as 1, there is again a linear decision boundary in “logit” units. The threshold is a logit of 0.0, which in a probability metric is .50. Forecasting accuracy may be better or worse than for linear regression. Linear regression assumes that in the metric of the 1/0 outcome, relationships with the predictors are linear. Logistic regression assumes that in the metric of the 1/0 outcome, relationships with the predictors are S-shaped (i.e., the cumulative logistic function). Which of these leads to better forecasts in a given setting will usually be an empirical matter. Both functions are typically arbitrary because there will rarely be compelling subject-matter theory requiring one

or the other.²¹

6.3.2 Building in Differential Forecasting Error Costs

To this point, all classification errors are given equal weight. A success classified as a failure counts the same as a failure classified as a success. This is why the least squares regression minimizes the number of forecasting errors. In many criminal justice settings, the assumption of equal weights is not responsive to the preferences of stakeholders. For example, the consequences of forecasting a parole success for an individual who will fail can be far more serious than forecasting a parole failure for an individual who will be a success. The parole failure may entail a heinous crime. Failing to release an individual who would be crime-free leads to increased time behind bars. Both forecasting errors are costly, but for many stakeholders, the costs to victims of a heinous crime are far greater than the costs of extra prison time. Whether or not these relative costs generally hold, an assumption that all forecasting errors have equal costs is likely to be unrealistic.²²

And costs matter for forecasts meant to inform real decisions. Figure 6.2 shows why. Using the broken line as the decision boundary, there are two successes that are incorrectly classified as failures. For this illustration, suppose that stakeholders think that the costs of “over-incarceration” are greater than the costs of crimes committed while on parole. There are reasons, therefore, to upweight the blue mistakes relative to the red mistakes. We show this in Figure 6.2 by making the two blue mistakes much larger. A new linear decision boundary results. Least squares regression can be

²¹Linear and quadratic discriminant function analysis has much in common with logistic regression and has been used in criminal justice risk assessments. We do not consider linear or quadratic discriminant function analysis because one must assume that the predictors have a multivariate normal distribution (Hastie et al., 2009, section 4.3). This is unrealistic for most predictors in criminal justice settings, especially when any of the predictors are categorical.

²²A more complete discussion about the role of asymmetric costs is beyond the scope of this paper. An excellent treatment can be found in a special issue of the Albany Law Review (Bushway, 2010).

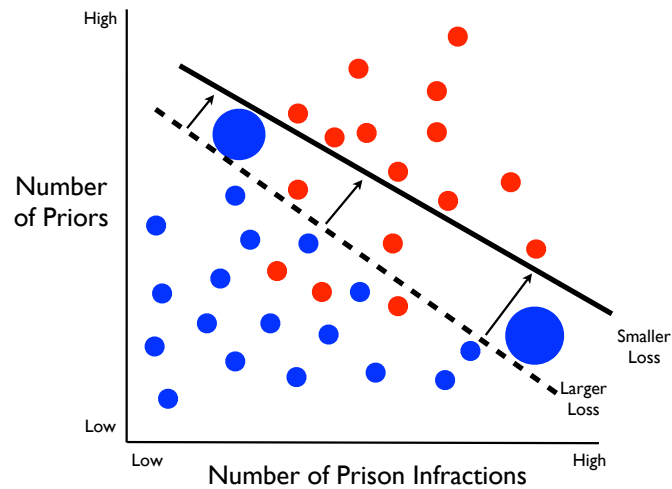


Figure 6.2: Impact of asymmetric costs in 2-dimensional predictor space.

used as before. But the decision boundary shifts toward the upper right with perhaps also a change in the slope.

The two blue mistakes are now accurately classified as successes. They no longer count as errors. But in trade, there are now five rather than three misclassified red circles. It looks like a wash — there are two fewer successes classified as failures, and two more failures classified as successes. But it is not a wash. The new decision boundary is to be preferred because the original two blue mistakes were much more costly than the two new red mistakes.

If the new decision boundary is preferred, many of the forecasts can change. In this example, cases to be forecasted as failures will need a greater number of prior offenses and a greater number of prison infractions than previously. The increase will be larger for the number of prison infractions because the new decision boundary was shifted outward more for the infractions predictor.

The point is that not all forecasting errors are created equal, and the relative costs

of different kinds of forecasting errors should be built into any classification/forecasting procedure. To ignore this issue is to assume equal costs. And if equal costs are not consistent with stakeholder preferences, the forecasts will not be properly responsive. Misleading forecasts can result.

6.3.3 Nonlinear Decision Boundaries

Why be limited to linear decision boundaries? Nonlinear boundaries can in principle perform better. In Figure 6.3, we reproduce much of Figure 6.1, but now with a nonlinear decision boundary shown by the dotted line. There are no red circles falling below the nonlinear decision boundary, and no blue circles falling above the nonlinear decision boundary. Classification is perfect. The prospects for forecasting accuracy look very promising indeed.

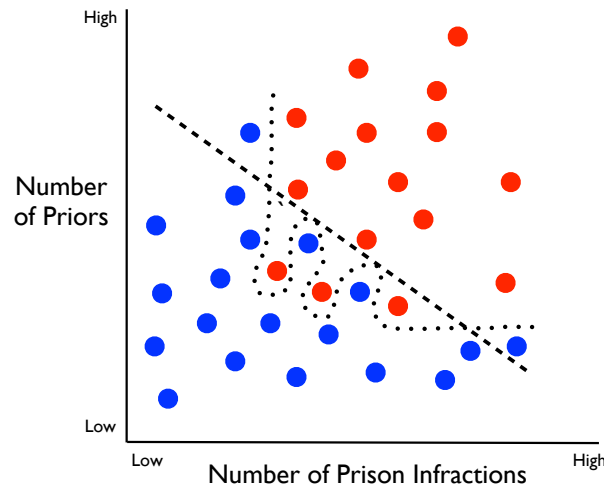


Figure 6.3: A linear and nonlinear decision boundary in 2-dimensional predictor space.

The linear decision boundary is far less complex than the nonlinear decision bound-

ary.²³ The price for greater simplicity is more classification errors. Clearly, one's ability to classify accurately is enhanced when the decision boundary can be more complex. It is easier for the nonlinear decision boundary to respond to complicated data structures.

A sensible statistical aim, therefore, can be to use predictors in a manner that allows for nonlinear decision boundaries as needed. There can be two related approaches (National Research Council, 2013). For parametric procedures such as logistic regression, greater complexity can in principle be addressed by including a larger number of predictors. Transformations of predictors can help. For instance, one might include not just the age of an inmate, but some polynomial function of age. One might even break up age into a set of binary dummy variables. Statistical interactions might also be captured with products of variables. The point is that the capacity to address greater complexity needs to be built in from the beginning or determined later in a set of very effective exploratory procedures. Also required is that the requisite predictors are included in the dataset. Many would argue that these requirements cannot be met in practice.

For nonparametric procedures such as smoothing splines (Hastie et al., 2009), one may include as many predictors as possible, along with promising transformations, but the *procedure* attempts to determine the decision boundary complexity needed. At one extreme, the fitted values are a hyperplane (just as in conventional linear regression). At the other extreme, the fitted values are an interpolation between all data points. The former is much less complex than the latter. In practice, some result between these extremes is typical. In contrast to parametric methods like

²³There seems to be no consensus on how best to define the amount of complexity. One popular approach is the degrees of freedom used to construct the decision boundary. In this example, the nonlinear decision boundary would use many more degrees of freedom than the linear decision boundary. A closely related approach is link complexity to the “effective dimension” of the statistical procedure or in some cases, the data itself (National Research Council, 2013).

logistic regression, an *adaptive* process is used to arrive at a decision boundary — the procedure exploits information in the data to determine both the shape and location of a decision boundary.²⁴ Unless a researcher is close to prescient and has the data rich enough to constructively respond, adaptive procedures start with a substantial forecasting advantage.²⁵

But there is a downside to adaptively determined decision boundaries. As a greater number of degrees of freedom is used up for a given sample size, there is the real risk of increased instability in the results. There is less information available per procedure parameter. In addition, there can be overfitting in which the procedure responds to idiosyncratic features of the data. Because forecasting involves new data, not the data used to develop the decision boundary, forecasting accuracy can be disappointing. The procedure does not generalize well to new data, which is precisely what forecasting entails.

For example, an individual with a large number of priors and a large number of prison misconducts may have a high probability of failure on parole. But a high probability is not a certainty. If that individual does not fail, a complex decision boundary would try to accurately classify that individual as a success. As a result, an anomalous case inconsistent with most of the data would help shape the decision

²⁴Stepwise regression is an example of a very simple adaptive procedure within a conventional regression framework. But again, distinctions may not be sharp. When researchers respecify their models after looking at the results, the final model is shaped by data-informed induction. Some would say that the difference is that the model selection process is not built into the data analysis algorithm itself.

²⁵If resources allow, a parametric brute force approach may help to level the playing field. With thousands of observations and hundreds of predictors, one can in addition construct *a priori* many nonlinear transformations and interaction variables. In effect, the researcher tries to anticipate how a effective adaptive procedure could respond. All of the original predictors and new transformations can then be included in a single “kitchen sink” regression. The regression will likely be uninterpretable. The complexity and multicollinearity alone could be toxic. If model selection procedures are applied to simplify, one is doing a seat-of-the-pants adaptive modeling with all of its attendant problems (Berk et al., 2010). Why settle for a brute force approximation to the desired procedure? An example can be found in the recent paper by (Tollenaar and Van der Heijden, 2013).

boundary. When that decision boundary is then used for forecasting with data in which such anomalous cases were absent, the decision boundary would not perform as well. It would be unnecessarily complex and risk an increase in forecasting errors. Looking back at Figure 6.3, if any one of the 3 red circles had as little as one or two more prison infractions or priors, the red circle would have fallen above the linear decision boundary, and one of the fingers in the nonlinear decision boundary would not have been constructed.

There are useful responses to overfitting, often called “shrinkage” or “regularization.” The intent is to reduce the instability. With smoothing splines, for instance, the fitting function is penalized for increases in complexity (Hastie et al., 2009). In a least squares context, the residual sum of squares is increased based on model complexity so that what might be the smallest sum of squared residuals no longer is the smallest. A residual sum of squares that starts out being larger, but has a smaller penalty because of less complexity, can be the preferred minimizer. In other words, a price is put on complexity that does not substantially improve the fit.

Another approach, “bagging”, capitalizes on a large number of random samples with replacement from the data on hand. A classification procedure is applied to each sample, and the results are averaged across samples. One important consequence is that idiosyncratic results tend to cancel out.

Finally, in this illustration, the two predictors have substantive interpretations. In general, parolees with a great number of prior arrests and a greater number of prison infractions are more likely to fail on parole. However, any substantive insights are a bonus. The primary goal is to classify accurately because that can lead to the most accurate forecasts. With respect to that goal, the two predictors could as well be longitude and latitude. This allows for the possibility of using “black box” classification procedures, for which no apologies need be made. One does not

have to rely a “structural model” when forecasting is the primary motive. Indeed, the requirement of a structural model can undercut forecasting accuracy (Breiman, 2001b). Two different masters are being served.

In summary, when forecasting accuracy is the primary goal, parametric approaches such as logistic regression can in principle perform as well as nonparametric approaches when the best decision boundary is relatively simple, and when the predictors required by the correct model are available in their proper form. When the best decision boundary is complex and/or the requisite predictors are not all available, nonparametric procedures will forecast more accurately, often substantially more accurately.

6.3.4 Enter Machine Learning

Where does machine learning come in? The goal of machine learning can be to find the “right model.” But when machine learning is used strictly as a forecasting procedure, the connections to conventional regression models become very distant indeed as there is no structural model even in principle.

The transition to machine learning can confer a number of important benefits, some of which are not readily available otherwise.

- (a) One is not limited to classifiers able to forecast one of two outcome categories.

In some recent applications, for instance, parole outcomes are forecasted for three classes: an arrest for a violent crime, an arrest for a crime that is not violent, and no arrest (Berk et al., 2010). Increasingly, criminal justice agencies want to forecast more than the binary outcome of any arrest versus no arrest (Berk, 2012). The kind of arrest really matters. In particular, arrests for crimes of violence are distinguished from other kinds of arrests.

- (b) Forecasting errors that do not have equal costs can be introduced into the procedure at the beginning so that all of the results properly represent the preferences of stakeholders (Berk, 2011).
- (c) Regularization is often built directly into the procedure to increase forecasting accuracy (Hastie et al., 2009, chapter 5).
- (d) Highly unbalanced distributions for the classes to be forecasted create no special problems as long as the rare outcomes are important enough to be given extra weight in the analysis. For example, in some recent work for individuals primarily on probation, the outcome classes to be forecasted included a class for homicide or attempted homicide, which represented only about 2% of the outcomes (Berk, 2009; Berk et al., 2009).
- (e) Some procedures work well and in a principled manner with an enormous number of predictors and even when there are more predictors than cases (Hastie et al., 2009, chapter 15).

6.4 The Forecasting Contestants

We will compare the forecasting performance of three different classifiers: logistic regression, **RF**, and stochastic gradient boosting. Logistic regression represents business as usual over the past 50 years. It is a special case of the generalized linear model, and very familiar to criminal justice researchers. **RF** and **SGB**, both ensemble-of-trees methods, will be the machine learning contestants. All of the evidence to date indicates that these ensemble procedures can perform well in criminal justice applications (Berk, 2013). All three algorithms are worthy competitors. We now turn the

discussion to the introduction of asymmetric costs to each of the three competing procedures.

6.4.1 Logistic Regression with Asymmetric Costs

Logistic regression, sometimes called binomial regression, is a special case of the generalized linear model. As such, it is meant to represent how nature generated the data — it is an algebraic translation of subject-matter theory. In that sense it is a “structural model,” and forecasting can be little more than an afterthought. Nevertheless, if the theory is correct and its algebraic representation is consistent with the theory, accurate forecasting can result.

Forecasting is undertaken through the regression’s fitted values. These can either be in logit (i.e., log odds) units or probability units. Researchers typically use the probabilities when forecasting. To get from the probabilities to a forecasted class, a single threshold must be applied. For example, it is common to use a threshold of .50. Probabilities greater than .50 are assigned one outcome class (e.g., failed on parole). Probabilities less than or equal to .50 are assigned the other outcome class (e.g., succeeded on parole).

The threshold of .50 implies that the costs of false negatives and false positives are the same. As already noted, they are usually not the same. Suppose a “positive” is a person who commits a violent crime. Suppose a “negative” is a person who does not commit a violent crime. It follows that if false negatives are three times more costly than false positives, one should use a threshold of .25. Cases with predicted probabilities greater than .25 are forecasted to be violent offenders. Cases with predicted probabilities equal to or less than .25 are forecasted to not be violent offenders. It is three times easier for a person to be forecasted a violent offender than a nonviolent

offender or no offender at all ($.75/.25 = 3$).

Altering the threshold *only affects the step from probabilities to classes*. All of the other logistic regression output is computed under the assumption that false negatives have the same costs as false positives. In particular, the logistic regression coefficients would almost surely be different had the actual relative costs of false negatives and false positives been properly taken into account. It can be a serious error, for instance, to use the regression coefficients as weights for constructing risk assessment instruments.

Finally, logistic regression can only be used for binary outcomes. These days, criminal justice stakeholders often want much more — they want to forecast different kinds of crimes. As already noted, in some applications the intent is to work with three crime categories: arrests for violent crimes, arrests for crimes that are not violent, and no arrest at all. In the context of probation supervision, one motivation is to move supervisory resources from individuals who do not threaten public safety to individuals who do, a strategy that has been shown to work well (Berk et al., 2010). When there are more than two outcome classes, multinomial logistic regression may be an option, but there are a number of unresolved issues about how best to go from predicted probabilities for each class to the classes themselves.

6.4.2 Random Forests with Asymmetric Costs

There are several ways to introduce asymmetric costs into RF. Perhaps the best way is to employ stratified sampling in Step 1a of the RF procedure given in Algorithm 1. There is one stratum for each outcome class. Sample sizes for each stratum are determined so that some outcome classes are oversampled and some are undersampled. In effect, the oversampled classes are given more weight as each tree is grown, which in

turn will affect the balance of false negatives to false positives. That balance captures relative costs. For example, if there are 10 false positives for every false negative, false negatives are necessarily 10 times more costly than false positives.

The approach is similar in spirit to the balanced RF approach (Chen et al., 2004), but does not necessarily use equal sampling sizes for the classes. By relying on this resampling approach, each tree in the RF is constructed with the cost asymmetry taken into account. Not only will the predicted classes reflect this asymmetry, but any additional model output, such as variable importance scores, will as well. Additionally, since each tree draws a unique stratified bootstrap sample based on the desired sampling size values for each class, different observations appear in each tree. Hence, even when a particular class is undersampled, all observations are likely to appear in at least some trees and can contribute to the overall ensemble. Rebalancing the dataset via undersampling before deploying RF, on the other hand, would result in a loss of data and potential predictive information (Chen et al., 2004).

Note that similarly to logistic regression, asymmetric costs can also be accounted for by altering the threshold on the voting rule for classification. For example, rather than using majority vote, $2/3$ of the trees might be required to classify a case as readmission in order to assign a final label of readmission. A major drawback of this approach is that asymmetric costs are only incorporated after the model has been fully constructed. The asymmetric costs are not accounted for in the tree-growing process, which is suboptimal (Hastie et al., 2009).

6.4.3 Asymmetric Costs in Stochastic Gradient Boosting

SGB can be employed for classification by using Bernoulli deviance as the loss function. Unfortunately, unlike in RF, there is no option to oversample or undersample

the different strata of the outcome. `SGB` using Bernoulli deviance outputs probabilities as its fitted values, and asymmetric costs are incorporated are introduced at the end of the procedure by altering the threshold for classification as is done in logistic regression. Experience to date suggests that `SGB` can perform about as well on classification problems as `RF`.

6.4.4 A Simulation

Logistic regression can forecast well when it is able to capture the data structure. However, logistic regression is not adaptive and depends on the researcher to specify an effective model. Important nonlinearities and interaction effects must be anticipated and included using the available predictors. If the researcher lacks the requisite insight or data, logistic regression will necessarily stumble. In contrast, adaptive procedures such as `RF` or `SGB` can shine because both algorithms are designed to search for structure with each pass through the data.

Figure 6.4 shows a fictitious dataset constructed to illustrate when logistic regression will perform poorly and `RF` or `sgb` will perform well.²⁶ It is by intent a worst case scenario for logistic regression and is not meant to represent in general the relative merits of the forecasting competitors. We are trying to address *why* nonparametric methods can forecast better than parametric methods. The exercise is didactic.

There are 100,000 observations. The outcome is binary. Red is coded 1 and blue is coded 0. There are two predictors. The 2-dimensional predictor space contains a blue area that is homogeneously successes and two red areas that are homogeneously failures. The graphical conventions are no different from those used for the earlier figures except that the colored circles for individual observations are replaced by

²⁶The lessons learned can be applied far beyond logistic regression to any parametric regression approach. The lessons also apply to a wide range functions that have clear structures, but are very difficult for parametric regression models to capture.

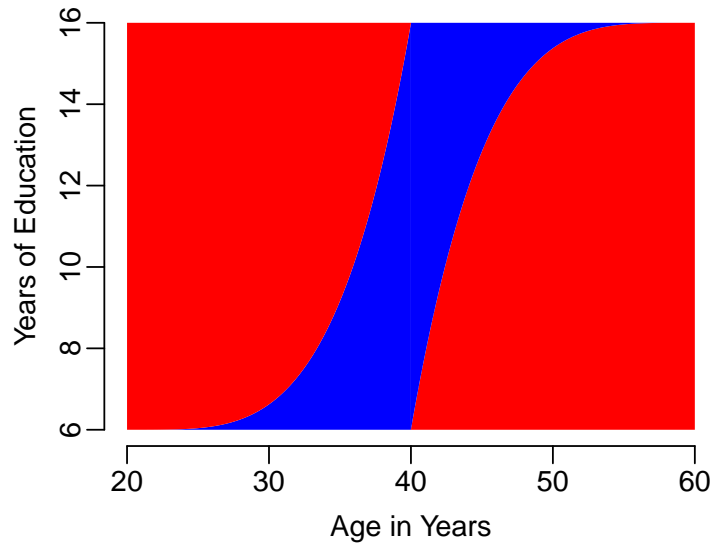


Figure 6.4: A very challenging classification example.

solid colors for different regions. It is as if we have printed a very large number of overlapping red circles and a very large number of overlapping blue circles. However, the data structure is far more complex because the blue region has red regions to its left and its right. Complex data structures of this sort are routinely analyzed in the classification literature (Hastie et al., 2009), but usually with many more than two predictors so that visualizations such as Figure 6.4 are unavailable. Any researcher trying to arrive at the correct parametric model from an examination of a scatter plot would necessarily be flying blind.

The surface was built by first drawing one predictor from a uniform distribution. The second predictor was constructed as a power function of the first. Then the predictor space was partitioned to show an interaction effect: both predictors had to be high or low for the area to be red. That is, there are nonlinear effects and an interaction effect. Because each of the three regions is perfectly homogeneous, the

data provide a clear and compelling signal that a good classifier should be able to accurately detect.

After the fact, one might overlay the following subject-matter account. The outcome is whether or not a parolee finds employment. The blue area contains successes and the red area contains failures. On the horizontal axis is age in years. The young and the old do not do well. The vertical axis is years of education. The association is not strong but parolees with a lot of education or very little do slightly better. In addition, when the educational level is higher, the best ages for finding work are older.

Why might such patterns occur? The kinds of positions for which parolees apply and the kinds of employers who would hire them represent a very limited subset of all jobs. By and large, the positions will involve physical labor for which not much experience or skill is required. Pay will be low and the work will be hard. Younger parolees may not be inclined to seek such positions, and older parolees may be incapable of doing the work. Education may be largely irrelevant for most of the jobs a parolee will seek. But, those who have very little education may correctly target their job search only for menial positions. Those with more education may correctly understand that they have a wider range of employment options. Finally, having more education may give some older workers, who would have difficulty working at demanding menial jobs, the chance to take entry level white collar positions (e.g., taking orders and making change at fast food restaurants).²⁷

This *post hoc* account may well be wrong, perhaps very wrong. The intent is to provide a less abstract setting in which to think about each contestant's performance. By itself, the story has no impact whatsoever on how well a given classifier performs.

²⁷Some analyses we conducted for the program "Ready, Willing & Able" supported by the Doe Fund, are consistent with this account.

Any good classifier should forecast with near perfect accuracy. Unlike in real data, there is no noise.

When logistic regression is used, both regression coefficients are virtually zero.²⁸ Logistic regression is unable to extract any useful information from the two predictors. All that remains is the intercept, which is effectively the logit of the outcome variable's proportion of reds (i.e., .80). The distribution of the predicted probabilities ranges from .7958 to .8022. The predicted probabilities have almost no variability.

For didactic purposes and with no important loss of generality, we assume that the costs of false negatives are the same as the costs of false positives. The corresponding threshold of .50 is applied. It follows that forecasting error is minimized by always predicting red. 20% of the time the forecast would be wrong. The true reds would be forecasted with 100% accuracy, and the true blues would be forecasted with 0% accuracy. Table 6.1 shows the results.²⁹

	Predict Blue	Predict Red	Model Error
Actual Blue	0	20078	1.0
Actual Red	0	79922	0.0

Table 6.1: Logistic regression confusion table using simulated test data.

Suppose a researcher is astute enough to include in advance the product of the two predictors to capture an interaction effect. Our reading of criminal justice forecasting applications is that such interactions are rarely used, but it is useful to see how logistic regression performs when given an especially good opportunity to deliver.

Table 6.2 shows the results. Although there are now nonzero regression coefficients

²⁸The two regression coefficients are -.03 and -.01. Even with 100,000 observations, one cannot reject the null hypothesis of 0.0 for either.

²⁹Other thresholds would not change the performance of logistic regression. A threshold a very little bit below .80 would allow some blues to be correctly forecasted. The price would be a commensurate increase in reds forecasted incorrectly. Virtually no predictive information from the predictors is being used. The predictors might as well be ignored.

for all three regressors, there are still no predicted probabilities smaller than .5. As before, forecasting error is minimized by always forecasting red. Nevertheless, there is some meaningful information in the predicted probabilities, and with cost ratios that weight forecasting errors for blue cases more heavily than for red cases, some blue cases will be correctly predicted.³⁰ For example, if a cost ratio of 4 to 1 is used, actual blues and actual reds are both correctly forecasting about 2/3rds of the time. That may seem quite good, but for these data the appropriate target is perfection.

	Predict Blue	Predict Red	Model Error
Actual Blue	0	20078	1.0
Actual Red	0	79922	0.0

Table 6.2: Logistic regression with interaction confusion table using simulated test data.

How does an adaptive machine learning procedure perform? For illustrative purposes, we take **RF** as our machine learning champion. Table 6.3 shows the results for **RF** assuming equal costs. With respect to the cost ratio, we are comparing apples to apples. The same two predictors are used, but there is *no* product variable for an interaction effect. The researcher using **RF** is not allowed to be as clever as the researcher using logistic regression — **RF** begins with a model specification disadvantage. Still, **RF** is just about perfect. Given either outcome, **RF** forecasts correctly more than 99% of the time. The failure to be literally perfect results from randomness in the **RF** algorithm itself.

The implications of this forecasting contest are clear. When the data structure is complex, machine learning procedures can perform very well. An adaptive process that “learns” from data can be very effective. This is precisely what the large

³⁰The predicted probabilities now range from .5333 to .9384.

	Predict Blue	Predict Red	Model Error
Actual Blue	19975	102	0.005
Actual Red	92	79830	0.001

Table 6.3: Random forests confusion table using simulated test data.

literature in statistics and computer science says. Logistic regression and other parametric forecasting procedures will not perform as well unless the researcher is able to construct a parametric model that captures all of the significant features of the data structure. As already noted, this can be a daunting task.

6.5 An Empirical Example

We turn now to analyses of real data. The dataset was selected to be typical of those recently used in parole or probation settings. Recall, however, that it is very difficult with real data to arrive at results that are broadly generalizable.

6.5.1 Forecasting Arrests for Serious Crimes

The data address how well parolees manage under supervision. There are 20,000 observations in the training data and 5,000 observations in the test data. We consider whether an individual is arrested for a serious crime within 2 years of release on probation. Serious crimes include murder, attempted murder, rape, aggravated assault, and arson. About 13% fail by this definition. Such crimes are of widespread concern. Static and dynamic predictors include:

- (a) Date of Birth;
- (b) Number of Violent Priors as an Adult

- (c) Earliest Age for a Charge as an Adult
- (d) Total Number of Priors as an Adult
- (e) Earliest Age for a Charge as a Juvenile
- (f) Total Number of Priors as a Juvenile
- (g) Number of Charges for Drug Crimes as an Adult
- (h) Number of Sex Crime Priors as an Adult

There is nothing special about these predictors. They represent the usual kinds of information that is routinely available on parolees when they begin their supervision. From past experience, they can make important contributions to forecasting accuracy (Berk, 2012).

We first apply logistic regression to the training data. A threshold of .135 is imposed on the predicted probabilities in order to arrive empirically at a 5 to 1 cost ratio of false negatives to false positives. Table 6.4 is the confusion table that results when the model is applied to test data. From the column on the far right, about 44% of the true failures are misclassified and about 32% of the true successes are misclassified. The forecasting accuracy is within the range of recent studies with similar data (Berk, 2012) and could well be useful for decision-makers.

	Predict Fail	Predict No Fail	Model Error
Actual Fail	378	302	0.444
Actual No Fail	1385	2935	0.321

Table 6.4: Logistic regression test data confusion table for serious crime.

Table 6.5 is the confusion table for RF using the test data. The procedure was tuned to also arrive at a cost ratio of about 5 to 1 for false negatives versus false

positives. From the column on the far right, about 37% of those who actually fail are incorrectly identified and about 28% of those who actually do not fail are incorrectly identified. Forecasting accuracy for **RF** appears to be superior.

	Predict Fail	Predict No Fail	Model Error
Actual Fail	427	253	0.372
Actual No Fail	1196	3124	0.277

Table 6.5: Random forests test data confusion table for serious crime.

Table 6.6 is the confusion table for **SGB** using the test data.

A threshold of .13 was used on the predicted probabilities from the training data to empirically arrive at a cost ratio of about 5 to 1. From the column on the far right, about 42% of those who actually fail are incorrectly identified and about 32% of those who actually do not fail are incorrectly identified. **SGB** does appreciably better than logistic regression when forecasting failures, but only slightly better when forecasting successes.

	Predict Fail	Predict No Fail	Model Error
Actual Fail	396	284	0.418
Actual No Fail	1361	2459	0.315

Table 6.6: Stochastic gradient boosting test data confusion table for serious crime.

It appears that across the three tables, **RF** performs better than logistic regression and **SGB**. This is consistent with published studies (Berk, 2012). But one must not overstate what is learned from the comparisons we report. It is difficult to guarantee that after tuning, one is necessarily comparing apples-to-apples. We have tried to insure that for all practical purposes, the false negative to false positive cost ratios are the same for all three procedures. But the cost ratios are not identical, and

it is essentially impossible to make them so. The test data and training data are *different* random splits of the available dataset. Tuning done on the training data will carry over a bit differently to the test data, depending on the forecasting procedure. Moreover, each procedure was tuned with its own special set of tuning parameters. There is no guarantee that the results are fully comparable. Indeed, it is not even clear how to define such a thing.

Another important issue is whether the differences are large enough to matter. As already explained, that judgment depends on the application. For example, the agency from which these data were obtained supervises about 40,000 individuals on probation each year. About 5000 of these individuals are arrested for a serious crime within 24 months, most within less than a year. For failures, the difference of approximately 7% between the accuracy of logistic regression compared to RF translates into about 350 serious crimes. Roughly 50 of those will be homicides or attempted homicides, the perpetrator of which could be identified in advance by RF, but not by logistic regression. In this instance, stakeholders found the practical difference in forecasting accuracy dramatic.

If one is looking for firm conclusions about forecasting accuracy from our results and others, it is almost certain that properly applied, RF will always do at least as well as logistic regression and much of the time meaningfully better. SGB will do at least as well as logistic regression, but is somewhat less likely to dominate it.

6.6 Conclusions

Complex decision boundaries pose a significant challenge for logistic regression or any other parametric classifier. To forecast well, a researcher must understand the nature of the complexity, be able to properly translate that knowledge into an algebraic

expression, and then have the data to construct an appropriate model. These are daunting requirements for criminal justice applications.

In contrast, adaptive machine learning procedures have the capacity to empirically discover patterns in the data and construct suitably complex decision boundaries. The requirements are a conventional menu of predictors and a large enough sample to exploit them. The tree-based machine learning procedures we have reviewed can then perform well and have several other important assets that logistic regression lacks: the capacity for outcome categories with more than two classes, a natural way to build in the asymmetric costs of forecasting errors, and a variety of instructive output that builds in asymmetric costs.

In practice, performance differences between logistic regression and most machine learning procedures can be small if the true decision boundary is simple. But how would one know? If logistic regression is used because a simple decision boundary is incorrectly assumed, substantial forecasting accuracy can be forfeited. In criminal justice settings where real lives can be at stake, the consequences could be significant. Why take the risk?

Citation

Berk, R. A. and Bleich, J. (2013). Statistical Procedures for forecasting criminal behavior: a comparative assessment. *Criminology and Public Policy*, 12(3):315-544.

Using Random Forests with Asymmetric Costs to Predict Hospital Readmissions

Abstract

Sufficiently accurate predictions of hospital readmissions are necessary for the allocation of scarce clinical resources to reduce preventable readmissions. Here, we describe the use of a data-driven approach that relies on machine learning algorithms to predict readmission risk at the time of discharge. We employ random forests to clinical and administrative electronic health record data available from a cohort of 103,688 patients discharged from the acute inpatient settings of the University of Pennsylvania Health System between June 25th, 2011 and June 30th, 2013. We predict both 30-day all-cause readmissions and 7-day unplanned readmissions using only predictors available by the time of discharge. Using oversampling and undersampling of the different outcome classes of readmission and no readmission, we incorporate into our models the asymmetric costs of a false negative relative to a false positive from the perspective of a hospital. We developed a machine learning-based model using random forests with a 5:1 relative cost ratio for 30-day all-cause readmissions that

achieves a sensitivity of 65% and specificity of 71% on validation data, as well as a random forests model with a 20:1 cost ratio for 7-day unplanned readmissions that achieves a sensitivity of 62% and specificity of 66% on validation data and discover that prior health system utilization, clinical discharging service, and vital sign information were most predictive of readmissions. By modeling the complex relationships between many predictor variables and readmission data for a large health system, we demonstrate successful predictive models that can be used upon discharge to flag patients at high risk of readmission.

7.1 Background and Significance

Hospital readmissions (the return of a patient to the hospital within some specified time after discharge) often reflect poor quality care, and can result in significant patient distress and high health care costs (Jencks et al., 2009; Allaudeen et al., 2011b; Bisharat et al., 2012; Allaudeen et al., 2011a; Bradley et al., 2012, 2013a). Thus, reduction in hospital readmissions is a public health priority. In the United States, the Centers for Medicare & Medicaid Services (CMS) penalizes health care providers with high rates of 30-day all-cause readmissions (Au et al., 2012; Bradley et al., 2013b). Therefore many hospitals have an interest in accurately predicting patients risk of readmission in order to target their limited resources to prevent them (Amarasingham et al., 2013; Allaudeen et al., 2011a; Bowles et al., 2014; Ahmad et al., 2013).

Prior approaches to predict readmissions using regression models and clinical and administrative data have had poor to modest predictive power as measured by the Area Under the Curve (AUC) with values ranging from 0.61 to 0.70, as well as high sensitivity and low specificity (Allaudeen et al., 2011a; Au et al., 2012; Baillie et al.,

2013; Billings et al., 2012; Donzé et al., 2013; Umscheid et al., 2014; de Lissovoy, 2013). Additionally, preexisting models are often disease-specific with potentially limited generalizability (Allaudeen et al., 2011b; Bradley et al., 2013a). Further, all of these approaches have used manual regression modeling to select risk variables (Allaudeen et al., 2011b; Bradley et al., 2013b), a process that is particularly cumbersome when it is estimated that an acutely ill inpatient generates a median of 1,348 individual data points per patient per day (Chandra et al., 2011), ranging from laboratory values to free text data (de Lissovoy, 2013). This has created a need for novel approaches to prediction, such as the application of machine learning algorithms and natural language processing, which take advantage of big data available from electronic health records (EHR).

EHR data present the opportunity to leverage machine learning algorithms and data mining techniques to uncover previously unknown and often complicated patterns and associations that can be used to improve quality of care and inform clinical decision-making in real-time. The advent of affordable data storage and analysis technologies at the health system level has opened new frontiers in institutional evidence-based practice, including the application of predictive modeling techniques such as machine learning to identify patients at high risk of adverse events such as readmission.

Machine learning approaches have been successfully employed to predict adverse clinical outcomes using high-dimensional datasets such as EHR data, often significantly outperforming traditional regression approaches. In particular, the random forests (RF) algorithm (Breiman, 2001a) was developed to provide highly generalizable predictions based on an ensemble of tree-based classifiers that are individually derived from random subsamples of a training dataset. RF has been successfully employed as a predictive classifier across diverse application domains such as criminology (Berk,

2013), ecology (Cutler et al., 2007), and bioinformatics (Strobl et al., 2007).

A perceived limitation on the predictive performance of **RF** (and similar classifiers) appears when one class (or outcome) occurs less frequently than another in a dataset, as is the case with hospital readmissions, which occur relatively infrequently. The perceived limitation arises because the algorithm is designed to favor predicting the majority class (or most frequent outcome) in order to minimize overall misclassification error (Strobl et al., 2007). However, minimization of overall misclassification error implicitly assumes that the different types of misclassification errors (i.e. false positives and false negatives) have equal costs. Hospitals, on the other hand, face asymmetric costs for the different misclassification errors related to readmissions. In other words, falsely misclassifying patients as low risk and thus foregoing opportunities to prevent readmissions is likely more costly given existing financial penalties than falsely misclassifying patients as high risk, and providing transitional care services when they may not have been needed. By default, **RF** assumes equal costs and optimizes the misclassification rate under the assumption of these equal costs. In this scenario, where the costs of the different misclassification errors are assumed equal and the event rate for a particular outcome is low, the algorithm can achieve the lowest total misclassification rate by predicting no readmissions for all hospital discharges. However, this results in sensitivity values of 0. These limitations of **RF** can be overcome by modifying the **RF** procedure to incorporate the appropriate asymmetric costs during model construction. We offer such an approach herein.

We additionally demonstrate models that could be deployed at time of discharge, thereby allowing the health system to direct transitional care interventions and intensive discharge planning to patients at high risk of readmissions. Implementation of these models provides the potential for avoiding adverse outcomes that threaten patient safety and generate high costs for the health care system.

7.2 Objective

In this study, we describe the development and resulting performance of RF modeling approaches to predict both 30-day all-cause and 7-day unplanned readmissions. We utilize two key methods in our approach. First, we restrict our set of patient variables to only employ clinical and administrative data available to the health system by the time of a patient discharge, allowing for real-time prediction. Second, we construct cost-sensitive models that can accurately account for the relative rarity of readmission events and the asymmetric costs associated with misclassification errors related to readmission events.

7.3 Materials and Methods

7.3.1 Setting

The UPHS includes three hospitals with over 1,500 beds and more than 70,000 annual admissions (Baillie et al., 2013). We obtained data on 103,688 adults discharged from the acute inpatient settings of UPHS between June 25th, 2011 and June 30th, 2013. The unit of analysis is each unique hospital visit. Thus, individual patients can have multiple visits within the data set, and for each admission, a prediction is generated.

7.3.2 Outcome Variables

We consider the CMS 30-day all-cause readmission criterion as well as 7-day unplanned readmissions because the discharging hospital may face factors beyond its control that might influence readmission for a 30-day period (Joynt and Jha, 2012). Thus, a shorter window for readmissions may offer a more meaningful metric for hos-

pital quality. In addition, by targeting readmissions that may be more preventable, one can gauge the effectiveness of proposed prevention efforts more accurately (Lavenberg et al., 2014). We define readmissions using ICD-9 and CMS diagnosis-related group codes.

7.3.3 Predictor Variables

We examined 188 variables (e.g. predictors) for each hospital admission. Classes of predictors include information on demographics, previous admissions statistics, lab test values and vital sign recordings. For predictors with more than two values during a given hospital encounter (e.g. lab test and vital sign information), we retained only the first and last values recorded as well as the times of the recording. We excluded predictors that were not available by the time of discharge, such as procedure and diagnostic codes for that admission. We did, however, include for each patient the procedure and diagnostic codes from the most recent admission within the dataset (if applicable) prior to the current admission.

For quantitative predictors with large proportions of missing data such as laboratory tests, we transformed quantitative values into categorical predictors consisting of deciles and added an additional level for missing values. For these recoded quantitative predictors, we additionally relabeled extreme values (defined as being in the top or bottom 0.5 percentile of a given predictors distribution) to a level denoted as extreme.

7.3.4 Random Forests with Asymmetric Costs

As previously discussed, the cost of an incorrect prediction of readmission (false positive) is not likely to be equal to that of an incorrect prediction of no readmission

(false negative). This cost asymmetry may arise from resource considerations and constraints, such as a limited staff to employ readmission reduction interventions, as well as financial penalties associated with hospital readmissions. Here, we consider failing to identify a readmission to be relatively more costly. For the 30-day model, we consider a failure to identify a readmission (i.e. a false negative) as 5 times more costly than incorrectly predicting a readmission for a patient who is not readmitted (i.e. a false positive), generating a 5:1 cost ratio. Given the relative rarity of 7-day unplanned readmissions, there must be a substantial cost incurred by the health system for failing to identify patients with this type of readmission, otherwise these cases are not worth identifying from a cost minimization perspective. Thus for the 7-day unplanned readmissions, we consider a cost ratio of 20:1. Note that these cost ratios are meant to be illustrative and should be appropriately determined based on each hospital’s unique circumstances.

Recalling the discussion from Section 6.4.2, we incorporate these cost ratios into the model by oversampling cases resulting in a readmission and undersampling cases resulting in no readmission until the desired empirical cost ratio is achieved on the OOB estimates obtained from the model.

7.3.5 Variable Selection

In theory, a health system could construct a RF approach using the full set of predictors available by the time of discharge and allow the model to dynamically determine which predictors are most useful. From an implementation perspective, however, it can be difficult to always collect such a complete set of information for each patient in real-time. Therefore, we strive to develop models using a smaller subset of the predictors without sacrificing predictive performance. This is achieved via a backward

selection that iteratively removes predictors with the lowest “variable importance scores” (Breiman, 2001a). For each predictor, the variable importance score we use represents the average increase in prediction error for a given outcome class when the predictor of interest is not allowed to contribute to the current model. Hence, predictors with higher scores (and thereby larger implied performance losses) are more important. Our elimination procedure continues removing predictors until the cost-sensitive performance begins to degrade when compared to performance when using the full set of predictors available. No stopping criterion was employed, but rather the cost-sensitive performance results were manually inspected to determine the final set of predictors. We utilized the variable importance scores computed within the `randomForest` package and describe these scores in detail in Section 7.4.2.

7.3.6 Model Evaluation

In any machine learning application, it is of utmost importance to have insight into how this model will perform on future data before actual implementation into the production environment. It is possible to obtain an accurate assessment of future performance, but proper procedures must be followed. The variable selection procedure and model construction must occur on one subset of the data (the training data) and then evaluated on a different subset of the data (the validation data). The validation data should not be used for model construction, otherwise any estimates of the models future performance using this test data would be invalid. Thus, we performed variable selection and model development on a 75% random subset of the data, reserving the remaining 25% to validate model performance. Note that under this scheme, we are implicitly assuming stationarity with respect to time across the sample period. That is, the relationship between the predictors and readmission risk

does not change over the time period we examine.

7.4 Results

7.4.1 Descriptive Characteristics

Table 7.1 provides a summary of the cohort of patients used in this study.

Characteristic	Value
Mean age, years (SD)	58.5 (17.2)
Female, %	50.1
Race/Ethnicity, %	
White	56.2
Black	36.7
Other	7.1
Median length of stay (IQR), days	3.7 (2.6, 6.5)
Discharging service, %	
Medicine	56.2
Surgery	22.6
Mixed/Other	25.8
Top 5 primary discharge diagnoses, %	
Circulatory system diseases	22.2
Neoplasm	13.4
Injury and poisoning	13.3
Digestive system diseases	9.7
Musculoskeletal system diseases	9.4

Table 7.1: Study cohort characteristics.

Table 7.2 shows the incidence of 30-day all-cause and 7-day unplanned readmissions. Here we characterize all clinical services into either one of Medicine or Surgery. Overall, admissions to medical units result in slightly higher rates of readmissions than those to surgical units.

We observed substantial heterogeneity of readmission rates when different ser-

Readmission Window	Medicine	Surgery	Overall
7-day	4.7%	3.4%	4.2%
30-day	13.8%	8.1%	11.3%

Table 7.2: Incidence rates for 30-day all cause and 7-day readmissions.

vices are compared. For example, categorizing individual clinical services into either Medicine or Surgery services, we observe that the Hematology/Oncology service attains the highest overall readmission rate (24.7% of visits result in 30-day all-cause readmissions and 9.1% of visits result in unplanned 7-day readmissions), and that the Transplant service had the highest readmission rate among surgical services (22.1% of visits result in 30-day all-cause readmissions and 9.9% of visits result in unplanned 7-day readmissions). These results demonstrate the importance of hospital service in prediction of readmission risk.

7.4.2 30-Day All-cause Readmissions Random Forests Model

We used OOB confusion matrix-derived performance metrics to achieve a sampling scheme that resulted in a 5:1 cost ratio as defined by the ratio of false positives to false negatives (Figure 7.1). The cost ratio achieved from OOB estimates was 4.97 (19,794 false positives divided by 3,977 false negatives), which corresponded to the 5:1 assumed stakeholder cost ratio preferences. Since there are about five times more false positives than false negatives, but each false negative is five times more costly, the overall cost of the prediction errors are roughly balanced.

Performance conditional on our given cost ratio is examined via the row and column proportions in the confusion matrix. The column conditional proportions provide sensitivity and specificity measures, given as 63.3% and 70.4%, respectively. The two main rows of the confusion matrix give the positive and negative predictive

		Readmission Outcome		
		Has	Has No	Prevalence
		Readmission	Readmission	0.14
Model Prediction	Predict Readmission	6,856	19,794	PPV = 0.26
	Predict No Readmission	3,977	47,139	NPV = 0.92
		Sens = 0.63	Spec = 0.70	

Figure 7.1: RF confusion matrix for 30-day all-cause readmissions on training data (OOB) using 5:1 cost ratio.

values, given as 25.7% and 92.2% respectively.

The resulting models performance was then evaluated on the 25% validation data, where we observed a cost-ratio of 5.01 (6,528 false positives divided by 1,303 false negatives), demonstrating strong concordance with the OOB values from the training data subset and suggesting that the approach performs well on the validation data with respect to stakeholder preferences (Figure 7.2). Similar to the OOB performance on the training data, the model achieved a sensitivity of 64.7% and a specificity of 70.6%, and positive and negative predictive values of 26.8% and 92.3% respectively.

Given the performance of the model, one may inquire as to which predictors are contributing to the black-box fit constructed by the RF. The variable importance scores computed by RF can provide such insights, and the variable importance plot for the 5:1 30-day all-cause readmission model is shown in Figure 7.3. For each predictor, there is an associated score representing the average increase in forecasting error for the outcome “readmission” (a similar plot is available for the outcome “no

		Readmission Outcome		
		Has	Has No	Prevalence
		Readmission	Readmission	0.14
Model Prediction	Predict Readmission	2,387	6,528	PPV = 0.27
	Predict No Readmission	1,302	15,705	NPV = 0.92
		Sens = 0.65	Spec = 0.71	

Figure 7.2: Random forests confusion matrix for 30-day all-cause readmissions on validation data using 5:1 cost ratio.

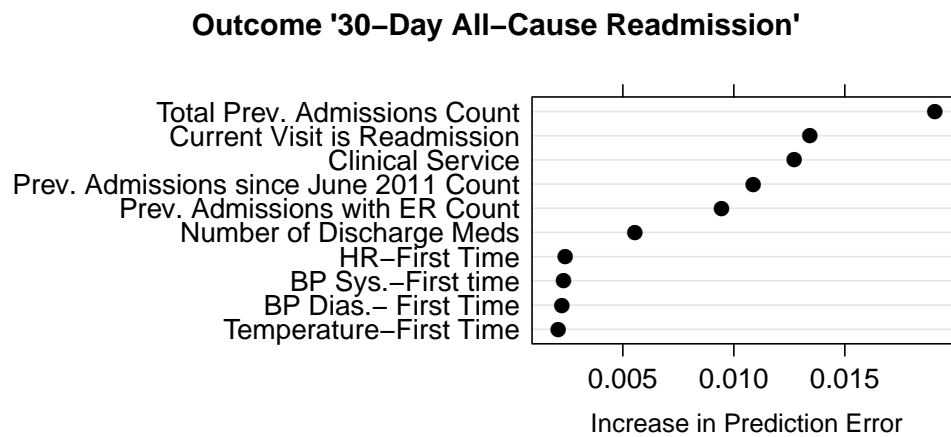


Figure 7.3: Variable importance plot for 30-day all-cause readmissions RF model. The top 10 variables for the class “readmission” are shown.

readmission”, but is not shown). Recall that this score is to be interpreted as the increase in prediction error assuming that the predictor is not allowed to contribute to the current model. Hence, these scores are conditional on the existing model and do not apply to a model refit with a given predictor excluded. For example, Figure 7.3 indicates that if the predictor representing the total count of previous admissions is not allowed to contribute to the current model, predictive accuracy for the outcome “readmission” would decrease by about 2.0%. It is important to note that for variables further down the plot, such as the first heart rate value, despite the variable importance value being near 0, this does not necessarily imply that the variable has no predictive power. Variables in RF models are intricately tied together through the tree construction process and singly weak predictors can have large aggregate contributions. The plot shown only considers contributions to predictive accuracy one predictor at a time.

We included 39 predictors in our selected model, and the top 10 predictors are included in Figure 7.3. Prior health system utilization, the clinical service at discharge, and vital signs were among the most important predictors.

7.4.3 7-Day Unplanned Readmissions Random Forests Model

To provide additional insight into preventable readmissions, we focused on 7-day readmissions that were identified as unplanned by ICD-9 and DRG codes. We used a cost ratio of 20:1, reflective of the decreased prevalence of unplanned 7-day readmissions. The training model had a sensitivity of 61.2%, a specificity of 65.3%, and positive and negative predictive values of 7.1% and 97.4% respectively.(Figure 7.4) The results for the validation data were similar (Figure 7.5). We observed an empirical cost ratio of 19.5, in accordance with the target cost ratio of 20:1. To assess the impact of pre-

dictors on unplanned 7-day readmission, we analyzed the importance of the top 10 predictors as done for the 30-day all-cause model (Figure 7.6). The most important predictors were similar for the two readmission outcome timeframes, except that the serum sodium value at discharge appears in the top predictors of 7-day unplanned readmissions and blood pressure measurements do not.

		Readmission Outcome		
		Has	Has No	Prevalence
		Readmission	Readmission	0.04
Model Prediction	Predict Readmission	1,988	25,857	PPV = 0.07
	Predict No Readmission	1,262	48,659	NPV = 0.97
		Sens = 0.61	Spec = 0.65	

Figure 7.4: RF confusion matrix for 7-day unplanned readmissions on training data (OOB) using 20:1 cost ratio.

7.5 Discussion

In this study, we report that two RF models sensitive to asymmetric costs can accurately predict 30-day all-cause and 7-day unplanned readmissions. We demonstrate that cost-sensitive models can achieve a balance between sensitivity and specificity, thereby overcoming the RF problem of majority class bias that is a hindrance to accurate prediction of a minority class, such as readmission events following discharge. We also demonstrate that the RF approach can utilize information encoded in EHR

		Readmission Outcome		
		Has	Has No	Prevalence
		Readmission	Readmission	0.04
Model Prediction	Predict Readmission	709	8,519	PPV = 0.08
	Predict No Readmission	436	16,258	NPV = 0.97
		Sens = 0.62	Spec = 0.65	

Figure 7.5: RF confusion matrix for 7-day unplanned readmissions on validation data using 20:1 cost ratio.

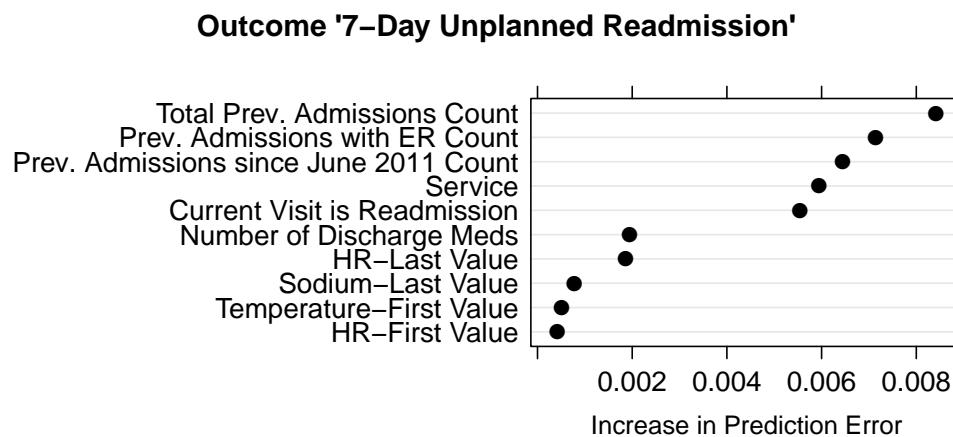


Figure 7.6: Variable importance plot for 7-day unplanned readmissions RF model. The top 10 variables for the class “readmission” are shown.

data to make predictions. Finally, we peek inside the black-box model and highlight the importance of variables such as health care utilization, number of discharge medications, and vital signs in the prediction of readmissions.

Considering the positive and negative predictive values for the 30-day all-cause readmissions model, when the RF predicts a readmission, it makes about 4 mistakes for every correct prediction. However, when it predicts no readmission, it is rarely wrong—only once in 20 predictions. The reason that the model is generally accurate when predicting no readmission, but not when predicting readmission, is a consequence of the 5:1 ratio of costs that were predefined. Given that it is of relatively high cost to misidentify a case that results in a readmission, the model casts a wide net and is more likely to predict a readmission unless it is fairly certain that the case will not result in a readmission. Health system priorities should dictate the width of this net. For example, UPHS administrators should consider the resource constraints or financial costs associated with 4 mistakes for each correct readmission prediction.

A previous modeling effort for 30-day all-cause readmissions for UPHS produced sensitivity and specificity values of 39% and 84%, respectively, and a PPV and NPV of 30% and 89%, respectively, on prospective data (Baillie et al., 2013). We see that our RF approach drastically improves sensitivity with some reduction in specificity and improves the NPV with a slight reduction in the PPV. It is important to note, however, that this is not a fair comparison. The results for the previous model mathematically imply a cost ratio of roughly 1.5:10, as compared to our current cost ratio of 5:1. And an honest comparison of models under asymmetric costs requires proper tuning of all of the models under comparison to the desired cost ratio (Berk, 2013). Given the lower cost of false negatives in the previous model versus ours, it is not surprising to see such a high value of specificity.

While the prototype models presented in this paper show promise for predicting

readmissions, there is still substantial opportunity to improve the predictions. There are almost certainly useful covariates missing from the available data set that could improve predictive accuracy, and other machine learning algorithms such as support vector machines and **SGB** could be explored. Additionally, the target outcomes could be refined to focus on preventable readmissions (Lavenberg et al., 2014), or readmissions for specific diseases or medical conditions. After a model is constructed, providers are then presented with the challenge of integrating the prediction tool into practice, and designing and testing interventions to be used when a patient meets threshold criteria for readmission risk. Finally, the machine learning framework presented herein is highly flexible and can be applied to a broad set of health system prediction tasks, such as predicting length of stay and assessing risk of mortality.

7.6 Conclusion

In summary, we have presented a data-driven machine learning approach to predicting hospital readmissions derived from EHR data obtained at the University of Pennsylvania Health System. We employed a machine learning approach that could process a large set of patient and visit-level information to discover patterns in the data associated with readmissions. Our approach demonstrates that cost asymmetries can be integrated into the construction of **RF** models such that their predictions best reflect user needs and values. Models such as those presented herein could be implemented into an EHR so that a prediction can be produced at time of discharge for each patient. This prediction can help target scarce health system resources to the patients in need of transitional care interventions, resulting in improved patient outcomes and reduced health system costs.

Citation

Bleich J., Cole, B., Kapelner, A., Baillie, C., Gupta, R., Hanish, A., Calgua, E., Umscheid, C.A., and Berk, R.A. (2015). Using random forests with asymmetric costs to predict hospital readmissions. *working paper*.

Bootstrapping for Stable Classification Trees

Abstract

Recent legislation in Pennsylvania mandates that forecasts of "future dangerousness" be provided to judges when sentences are given. Similar requirements already exist in other jurisdictions. Research has shown that machine learning can lead to usefully accurate forecasts of criminal behavior in such setting. But there are settings in which there is insufficient IT infrastructure to support machine learning. The intent of this paper is provide a prototype procedure for making forecasts of future dangerousness that could be used to inform sentencing decisions when machine learning is not practical. We consider how classification trees can be improved so that they may provide an acceptable second choice. We apply a version of classifications tree available in R, with some technical enhancements to improve tree stability. Our approach is illustrated with real data that could be used to inform sentencing decisions. We find that modest sized trees grown from large samples can forecast well and in a stable fashion, especially if the small fraction of indecisive classifications are found and accounted for in a systematic manner. But machine learning is still to be preferred when practical. We conclude that our enhanced version of classification trees may provide a viable

alternative to machine learning when machine learning is beyond local IT capabilities.

8.1 Introduction

Behavioral forecasts have informed parole decisions in the United States since the 1920's (Burgess, 1928; Borden, 1928). Over the decades, these forecasts have increasingly relied on quantitative methods that some would call actuarial (Messinger and Berk, 1987; Feeley and Simon, 1994). Despite jurisprudential concerns and forecasting accuracy that has been difficult to evaluate (Farrington and Tarling, 1985; Gottfredson and Moriarty, 2006; Harcourt, 2008; Berk, 2008, 2009, 2012), there is little doubt that these methods are here to stay. Forecasts using even very simple statistical procedures have been shown to consistently perform better than clinical judgments (Monahan, 1981; Hastie and Dawes, 2001), and there is growing support for forecasts of criminal behavior across a range of criminal justice settings in addition to parole hearings: bail determinations, charging, sentencing, and probation/parole supervision.

In this chapter, we focus on sentencing decisions. There are already jurisdictions that provide judges with quantitative forecasts of risk (Kleiman et al., 2007), and some are heading down the same path. For example, a recent Pennsylvania statute authorizes the Pennsylvania Commission on Sentencing to develop a risk forecasting instrument to help inform sentencing decisions under the state's sentencing guidelines. The history and politics are complicated (Hyatt et al., 2011), but for purposes of this discussion, the key section reads as follows:

42 Pa.C.S.A. §2154.7. Adoption of risk assessment instrument.

(a) General rule. – The commission shall adopt a sentence risk assessment

instrument for the sentencing court to use to help determine the appropriate sentence within the limits established by law for defendants who plead guilty or nolo contendere to, or who were found guilty of, felonies and misdemeanors. The risk assessment instrument may be used as an aide in evaluating the relative risk that an offender will reoffend and be a threat to public safety.

(b) Sentencing guidelines. – The risk assessment instrument may be incorporated into the sentencing guidelines under section 2154 (relating to adoption of guidelines for sentencing).

(c) Pre-sentencing investigation report. – Subject to the provisions of the Pennsylvania Rules of Criminal Procedure, the sentencing court may use the risk assessment instrument to determine whether a more thorough assessment is necessary and to order a pre-sentence investigation report.

(d) Alternative sentencing. – Subject to the eligibility requirements of each program, the risk assessment instrument may be an aide to help determine appropriate candidates for alternative sentencing, including the recidivism risk reduction incentive, State and county intermediate punishment programs and State motivational boot camps.

(e) Definition. – As used in this section, the term risk assessment instrument means an empirically based worksheet which uses factors that are relevant in predicting recidivism.

We address efforts to construct a forecasting prototype that might be used to inform sentencing under the Pennsylvania statute. One might ordinarily apply a tree-based form of machine learning already shown to be effective in related circumstances (Barnes et al., 2010; Berk et al., 2005, 2006, 2009) . However, these methods lead to forecasting procedures that are best implemented in real time by computers linked

to large criminal justice databases. For now at least, those capacities do not exist in many courtrooms. Moreover, insofar as the forecasts are to be integrated into a sentencing guidelines grid, there are software challenges.

As a fallback position, we suggest that classification trees can be improved in specific ways so that they perform well, and perhaps even well enough, to inform sentencing decisions. One would not ordinarily expect them to forecast as accurately as tree-based machine learning approaches, but the gains in ease of use may offer a sensible tradeoff. We consider these issues conceptually and then provide an illustration with real data. One can think of our results as not yet ready for prime time, but perhaps as an interesting pilot episode. In particular, we focus on the **CART** algorithm although other classification tree methods exist.

8.2 Modifying Classification Trees for Criminal Justice Settings

While we formally introduced the construction of classification trees in Chapter 1, we note that the resulting set of splits from a classification tree effectively constructs profiles of individuals. Hopefully these profiles result in similar outcomes. For example, men under 21, who are gang members with a long history of crime beginning at an early age, represent one simple and short profile likely to be high risk. Such a profile would be effective insofar as most (ideally, all) of the individuals with that profile have the same outcome class. In this instance, that outcome class might be an arrest for a violent crime while on probation. At the other extreme, women over 35 with no gang affiliation and who are first offenders would probably not be arrested for a violent crime while on probation. This profile would be effective in identifying individuals posing little threat to public safety. In both of these simple examples,

forecasts could naturally follow. Any individual with the first profile could be projected as high risk. Any individual with the second profile could be projected as low risk.

One benefit of classification trees is their ability to incorporate asymmetric costs directly into their growing procedure (Breiman et al., 1984). In our context, false positives are incorrect predictions of crime perpetration. False negatives are incorrect predictions that a crime will not be committed. From a law enforcement perspective, the costs of false negatives are often higher than the costs of false positives. A failure to identify a high risk individual is relatively more costly than falsely identifying a high risk individual, and these greater relative costs should be, and can be, taken formally into account. For example, if a profile conveys even a hint of an undesirable outcome, a felony arrest can justifiably be projected. One is prepared to live with an increase in the number of false positives if the number of false negatives can be meaningfully reduced.

Identifying the relevant consequences of different kinds of forecasting errors and specifying their relative costs usually depends heavily on subjective judgments. Hence, different stakeholders legitimately can have different relative cost assessments. How these are properly framed and reconciled can raise difficult issues that are beyond the scope of this paper. An excellent treatment can be found in (Bushway, 2010).

Classification trees are well-known for their ability to construct profiles that can classify individuals quite accurately under asymmetric costs. Unfortunately, the results can also be unstable (Hastie et al., 2009, section 9.2). With new data, such as those one would use to make forecasts, predictive accuracy can be disappointing. In other words, with new data, rather different profiles will sometimes be constructed with different implications for forecasts.

We address instability here by creating classification profiles from a very large

data set. Such data sets are increasingly common in sentencing settings. More stable results can follow. For similar reasons, we construct profiles with relatively few predictors. In addition, we introduce a special class of forecasted outcomes which one might call “can’t tell.” If even a relatively small number of “can’t tell” cases can be identified and properly isolated, stability is greatly increased, and forecasts for the vast majority of cases are substantially improved. We see this approach as a useful, practical advance.

8.2.1 Asymmetric Costs and Tuning

In practice, using a simple majority vote in terminal nodes as described in Section 1.2.1 is unsatisfactory because false positives and false negatives are treated as if they have the same costs; their relative costs are 1 to 1. Symmetric costs are the default in most software so that if the arguments in the loss function are ignored, symmetric costs are necessarily implemented. The California Static Risk Assessment Instrument, for instance, implicitly adopts the default of equal costs with no apparent rationale (Turner et al., 2009).

When symmetric costs are inconsistent with stakeholder preferences, the votes in each terminal node can be weighted to take asymmetric relative costs into account. For our later illustrative analysis, we make false negatives 5 times more costly than false positives. Thus, failing to identify a high risk individual is five times more costly than incorrectly labeling an individual as high risk.³¹ This weighting is broadly consistent with work we have done in the past (Berk, 2012).³²

³¹There is nothing special about the 5 to 1 cost ratio, and for the methodological issues raised in the paper, most any reasonable cost ratio would suffice. The cost ratio just could not be so extreme that the same class is assigned to essentially all cases. In that instance, the role of either false positives or false negatives would be obscured.

³²Cost ratios have ranged from 20 to 1 to 3 to 1. The relative costs of false negatives are more dear. Anecdotally, we have found that criminal justice officials and representatives from a variety

However, as discussed for **RF** in Section 6.4.2, the entire partitioning procedure should respond to differential costs, and we proceed in this fashion below. In other words, the profiles themselves are altered to account for the relative cost of false negatives to false positives. Hence, the entire classification tree is tuned to the 5 to 1 cost ratio. The procedure for incorporating asymmetric costs into **CART** is slightly different than that for **RF**. Asymmetric costs can be incorporated by either altering the prior distribution of the outcome or by explicitly introducing a loss matrix into the classification tree algorithm. We generally favor the latter because the relative costs are explicitly represented.

Consider a $K \times K$ loss matrix \mathbf{W} . For simplicity, and with no important loss of generality, suppose that $K = 2$. The response outcomes are “fail” or “not fail.” When the forecasting procedure misses a failure, one has a false negative. When the forecasting procedure incorrectly identifies a failure, one has a false positive. Then \mathbf{W} is

$$\begin{bmatrix} 0 & R_{fn} \\ R_{fp} & 0 \end{bmatrix}$$

where the entries along the main diagonal are zero, and the off-diagonal elements contain the relative costs of false positives (i.e., R_{fp}) and false negatives (i.e., R_{fn}). The units do not matter. What matters is the ratio of the two. For example, R_{fn} could be 10, and R_{fp} could be 1. False negatives are 10 times more costly than false positives. Put another way, 10 false positives have the same cost as 1 false negative.

Asymmetric costs, introduced so that the forecasts properly respond to stakeholder preferences, can affect not just the recursive partitioning and the classes assigned to terminal nodes, but evaluations of forecasting performance as well. Standard evaluation tools such as the ROC curve take the forecasting output as is. If that

of citizen groups broadly agree that at least for crimes of violence, false negatives are substantially more costly than false positives.

output is based on equal costs, so is the ROC. If the costs are not equal, performance measured by the ROC does not capture what decision-makers want (Hastie et al., 2009, section 9.2.5). For example, risk forecasts that score well by the ROC may implicitly give too little weight to false positives. This may inadvertently increase prison populations and diminish public confidence in the sentencing process.

Finally, for our illustrative analysis we chose tuning parameter values that can help improve stability. The intent was to prevent the partitioning from proceeding too far; smaller trees tend to be more stable because the number of observations in each partition is larger on the average. Precisely how one proceeds will depend on the software used. To build our classification trees, we applied the `rpart` package in R with which the number of terminal nodes can be controlled directly (Therneau and Atkinson, 1997). One happy consequence of smaller trees is that they can be easier to interpret.

8.2.2 Stability Analysis

Working with large data sets and smaller trees is unlikely to fully eliminate potential instability. Had the data been even slightly different, it is possible that the splits chosen could have differed substantially, leading to a new set of terminal nodes and perhaps new forecasts.

To consider how successful our strategy actually was, we explicitly explored prediction instability. We generated a substantial number of bootstrap samples, drawing from the training data with replacement. For each sample, a classification tree was grown using tuning parameters at their previously determined values. The classifications from each tree were stored. Cross tabulations were then undertaken for all possible pairs of trees. From each cross tabulation, the proportion of times the clas-

sifications were the same was computed. A very instructive analysis of tree stability followed, and some important refinements of the forecasts were implemented. Our approach draws heavily on the work of (Kuhnert and Mengersen, 2003). Details are provided below.

In this context, it is important to appreciate that different tree structures imply different substantive interpretations, and it is the structures that can be especially unstable. Predictions derived from terminal nodes are typically more robust because all that matters is the forecasted class. Many different tree structures can lead to the same forecast for a given case. For example, if males under 21 are at high risk, it does not matter for forecasting whether gender defines the first partition and age the second or whether that order is reversed. But it does matter if the variable used in the first partition is taken to be more important. It will also not matter if an age of 20 is chosen instead of 21 as long as 20 year olds and 21 year olds are assigned the same class as before through the impact of other predictors.

8.3 Data

The data used for the illustrative analysis include individuals on probation in Philadelphia between 2002 and 2005. There are 48,923 observations with each observation representing a *case*. An individual can appear in the data more than once as different cases. Because sentencing decisions are made on a case basis, using cases as the unit of analysis is consistent with the needs of decision makers and how one might understand the social processes responsible for the data. In fact, repeat offenders are not unusual, even over relatively short time intervals, but decisions are made case by case. In short, the case/individual distinction has no important implications for the forecasting procedures we use, but there are some related issues for the stability

analysis that will be addressed later.

A random subset of the data were used as training data. These were the data employed to construct the forecasting procedure. The remaining data served as test data from which proper assessments of forecasting accuracy could be obtained. Having both training data and test data is consistent with honest performance assessments and with recommended practice (Hastie et al., 2009, chapter 7).

8.3.1 Variables

Drawing on prior discussions with local stakeholders, the response variable for the analysis is whether an individual once placed on probation was arrested within two years for a violent crime: murder, attempted murder, manslaughter, robbery, assault, kidnapping, rape, and arson. These are “failures.” Of the 48,923 cases, 6,284 cases “fail.” Thankfully, failures are relatively rare. Roughly 15% of all cases fail. All other outcome are treated as the absence of “failure.”

Stakeholder rationale was this: a key consideration after most felony convictions is whether to incarcerate. That decision should be informed at least in part by a forecast of “future dangerousness.” Presumably, judges would be less inclined to release an individual on probation if they are projected to commit a violent crime.

There are, of course, other behavioral outcomes that could in principle be predicted (Berk, 2012). One example is whether an individual is arrested for a murder (Berk et al., 2009). Another example is whether an individual is arrested for any felony. Yet another example allows for three outcome classes: an arrest for a violent crime, an arrest for a crime that is not violent, and no arrest for any felony. A similar three-class outcome has proved to be useful for supervisory decisions of individuals on probation or parole (Barnes et al., 2010).

The usual collection of covariates was available as potential predictors. They can be grouped as follows:

- **Demographic Information:** gender, date of birth
- **Juvenile Priors:** total number of priors, number of sex offense priors, number of drug priors, etc.
- **Adult Priors:** total number of priors, number of sex offense priors, number of murder priors, etc.
- **Other Criminal Record Information:** prior number of days in jail, number of prior probation sentences, bail types, age at earliest prior, etc.

Race and ethnicity were excluded in deference to stakeholder sensitivities. Predictors that should be excluded on other than statistical grounds can raise subtle technical issues because individual, neighborhood, and behavioral attributes can be strongly related. In addition, because offenders tend to victimize individuals who share their “demographics,” controversial predictors may be just those needed to help protect the most likely victims. We have discussed these matters in some detail elsewhere (Berk, 2009).

To further complicate matters, the set of predictors was not limited to the covariates specified. A very important feature of classification trees is that step functions and higher order interaction effects are automatically constructed as needed. In effect, a large number of empirically derived basis functions can be determined. For example, young males from high crime neighborhoods may be projected as high risk. The three constituent covariates (age, gender, and neighborhood) are combined as a three-way interaction that may serve as a surrogate for gang membership, which is not directly measured. Age may serve as a splitting variable several times so that

	Predict ‘Fail’	Predict ‘No Fail’	Model Error
Actual ‘Fail’	3331	1697	0.34
Actual ‘No Fail’	8837	25100	0.26
Use Error	0.73	0.06	Overall Error=0.27

Table 8.1: RF confusion table (OOB) using a 5 to 1 cost ratio of false negatives to false positives.

a nonlinear function is approximated in several steps. In short, the set of potential predictors can far exceed the set of explicit predictors in ways that are not likely to be apparent. We refer interested readers to a more thorough treatment that is beyond the scope of this paper (Berk, 2012).

8.4 Random Forest Results

As a benchmark, we first employed RF to the data.

Table 8.1 shows the confusion table that results. The confusion table is a cross-tabulation of the actual outcome and the forecasted outcome, and for RF, these confusion table is computed on the OOB data.

We begin by considering model error addressed by the rows of the table. When the actual outcome is a violent crime, the model correctly forecasts that outcome 66% of the time ($1.0 - .34$). When the actual outcome is no violent crime, the model correctly forecasts that outcome 74% of the time ($1.0 - .26$). This is respectable performance compared to past research (Berk, 2012), especially when one recalls that failures are relatively rare in these data. Although about 15% fail in the overall pool of individuals, RF is able to correctly forecast nearly two-thirds of all failures. Moreover, forecasting accuracy should improve as additional predictors become available

in future work. Note also that the intended cost ratio of 5 to 1 is well approximated in the results: $8837/1697 = 5.21$.

Now consider use error. If RF were used to make actual forecasts, performance would then be conditioned on the forecast, not the outcome. The columns in Table 8.1 show the result, but have to be interpreted with some care. The 5 to 1 cost ratio imposed on the analysis means that a substantial number of false positives will be tolerated so that false negatives are reduced. This tradeoff is explicitly built into the column calculations. Thus, when a violent crime is forecasted, that forecast is correct only 27% of the time (1.0 -.73). The preference toward false positives has a large impact, just as intended.

When the absence of a violent crime is forecasted, that forecast is correct 94% of the time (1.0-.06). A happy outcome of the preference for false positives is that when a violent crime is not forecasted, it has a very good chance of being correct. Individuals not forecasted to fail could be very good probation risks if one is mainly concerned about future dangerousness.³³

8.5 Classification Tree Results

How well does a classification tree perform on the same data? Proper evaluation of a classification tree requires that the tree be evaluated with test data. Using training data to grow and evaluate a tree can lead to results that are too optimistic. For a given tree grown from the training data, test observations are placed in their appropriate terminal nodes. Classes previously assigned to those nodes provide the forecasts. Table 8.2 shows the results.

³³Overall model error is reported at the lower right corner of the table. However, when the loss function is asymmetric, the overall proportion correct (or in error) is misleading. Symmetric costs are being assumed.

	Predict ‘Fail’	Predict ‘No Fail’	Model Error
Actual ‘Fail’	719	515	0.42
Actual ‘No Fail’	2829	5722	0.33
Use Error	0.80	0.08	Overall Error=0.342

Table 8.2: Classification tree confusion table using test data.

For a given observed outcome, as before, each row shows the proportion of cases in which the classification tree gets it wrong. When an individual actually fails, the tree makes the correct forecast about 58% of the time (i.e., $1.0 - .42$). When an individual actually does not fail, the tree makes the correct forecast about 67% of the time (i.e., $1.0 - .33$). The empirical cost ratio is a little farther off ($2829/515 = 5.5$), but not enough to matter in practice. Nevertheless, the classification tree does not perform as well as RF. This is to be expected.

How much the reduction in accuracy matters in practice would be for stakeholders to determine. In this instance, for every 100 cases, the RF procedure would fail to predict 34 violent crimes whereas the classification tree would fail to predict 42 violent crimes. Stakeholders would need to decide how much a differential of 8 violent crimes per 100 cases matters. In Philadelphia, thousands of individuals are sentenced each year so that the number of violent crimes missed by the classification tree could be well over several hundred.

As before, for each column, for a given outcome predicted by the tree, the proportion of cases in which the forecast is correct is shown. When the tree forecasts a case to be a failure, it is right 20% of the time ($1.0 - .80$). The large number of false positives, which as before were favored by the 5 to 1 cost ratio, are a key factor in the disappointing performance. But again, there is good news. When a tree forecasts that an individual will not fail, it is right 92% of the time ($1.0 - .08$). As with RF,

when an individual is not predicted to commit a violent crime, it is a pretty good bet. Indeed, forecasting accuracy for such predictions is nearly the same as for RF.³⁴

But why settle for a classification tree when RF will usually forecast more accurately, especially if the associations between the predictors and the outcome are complicated? Perhaps the main reason is that the results of a classification tree can be easily translated into practice without the help of a computer.

8.5.1 A Classification Tree Visualization

Figure 8.1 shows a rendering of the classification tree built from the training data. The figure provides some information about which predictors are driving the forecasts and can be a tool for implementing the forecasts without the aid of a computer.

At the top of the tree (i.e., the root node) there are no partitions. Each successive split is shown as branches that lead to internal nodes. At the bottom of the tree are the terminal nodes (also sometimes called “leaves”) to which outcome classes are assigned. In Figure 8.1, terminal nodes are shaded. The classes assigned to each terminal node are shown in the rectangular boxes at the bottom of the figure, with number of failures on the left and the number of non-failures on the right. The assigned class can be used as the forecasted class.

The classification tree is simple by design. Any observation for which a forecast is needed can simply be “dropped down” the tree until a terminal node is reached. For example, if the offender in question has an adult first charge age of less than 21 years, move left. Then, if as a juvenile there are any charges, move left again. The observation then lands in a terminal node and would be forecasted to “fail” because of the cost-weighted vote: $(5 \times 2038) > 6774$. If the second split sends a case to the

³⁴One important reason stems from the large imbalance in the outcome distribution favoring the absence of violent crime. The absence of violent crime is far more common and, therefore, easier to predict correctly. RF earns its keep by forecasting rare events.

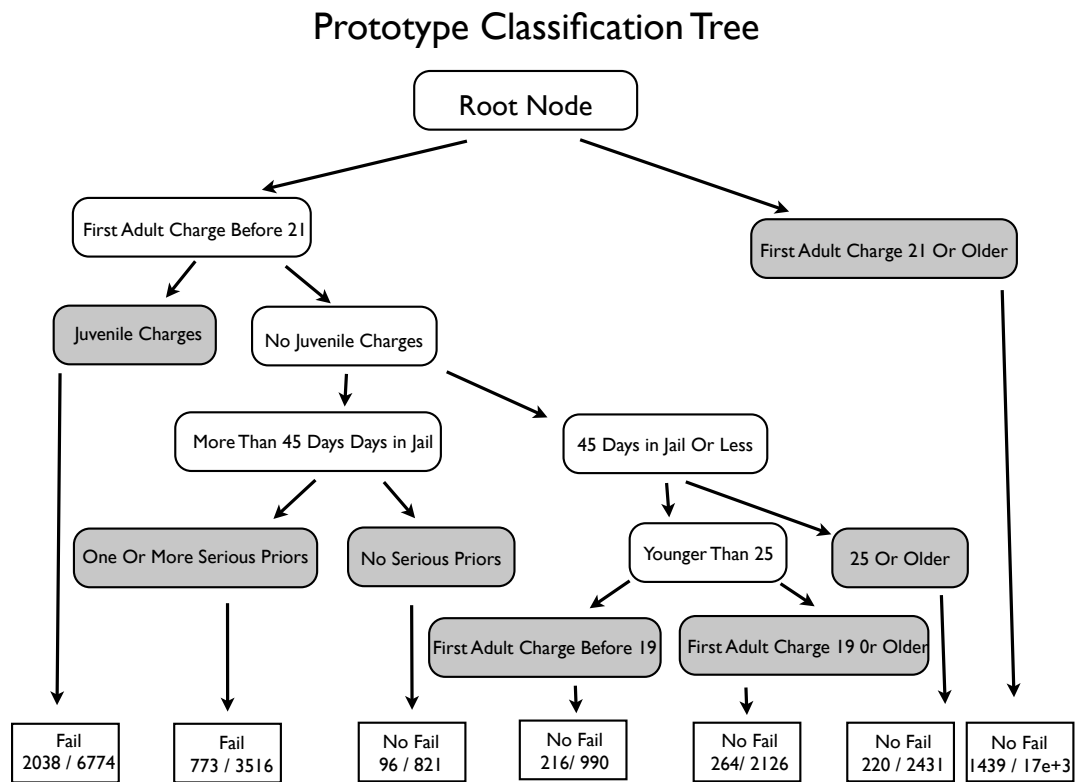


Figure 8.1: Classification tree for whether an individual is arrested for a violent crime.

right, there are additional partitions before one of several terminal nodes is reached.

All observations in need of a forecast can be treated in a similar fashion and will fall into one of the terminal nodes. As a result, the tree structure creates a straightforward set of selection criteria that can aid in sentencing decisions. One just follows that path appropriate for a given offender to its terminal node for which the predicted outcome is provided. In short, the classification tree results can be used in a paper-and-pencil fashion.

The structure of the tree can also provide subject-matter insights. For example, the terminal node on the far right contains all individuals whose first criminal charge occurred after the age of 21. They are the best risks. For those arrested and charged before they were 21, the class assigned depends upon interactions with other variables. For example, one relatively high risk group is composed of individuals whose first adult criminal charge was before the age of 21, who had a criminal charge as a juvenile, who spent at least 46 days in jail previously, and who had at least one serious charge as an adult. In linear model terms, this is a high order interaction effect that would probably not have been specified *a priori* or discovered with conventional regression diagnostics. At best, each constituent of the interaction would have been included as a main effect.

It is important to stress, however, that forecasting is the goal. Even if there are substantive insights in the tree structure, a classification tree is not a causal model. At best, there can be provocative associations.

8.5.2 Stability Analysis

There is a relatively small literature on uncertainty in classification trees, and it is segmented into somewhat different estimation problems and applications. Holmes (2003)

provides an excellent review for phylogenetic applications that has broader implications. A very different tradition can be found in computer science (Tóth, 2008). The work of (Kuhnert and Mengersen, 2003) seems most relevant to a sentencing setting.

Kuhnert and Mengersen (2003) consider several kinds of reliability questions but for us, the following question seems most germane. For a *given* classification tree, how reliable are terminal node classifications over realizations of the data? Here are the steps.

- (a) Generate B bootstrap “test” samples, sampling with replacement from the training data.
- (b) Using the original classification tree, drop each test sample down the tree and determine the class label to be attached to each terminal node using the Bayes classifier (i.e., the class assigned is the class with the most cases).
- (c) Compute the proportion of times the class assigned to the terminal nodes is the same as the class assigned with the original data.

All of the uncertainty is driven by sampling variation in the test samples. There is no uncertainty in the model itself. Moreover, stability is defined over terminal nodes, not the cases that fall within them. So a node containing 10 cases is treated the same as a node containing 100 cases even though a change in how the node is labeled affects 10 times more cases for the second node.

We find this formulation insufficiently responsive to our application. First, we view our classification tree as illustrative, not definitive. It could well change with new data from sampling variation alone. It is important, therefore, to capture *model* uncertainty. Second, judges sentence cases. It is the stability of case classifications that matters. Stability should be represented at the level of cases, not the level of terminal nodes.

As a result, we implement the following procedure.

- (a) We generated 50 “new” samples with replacement from the training data using the same number of observations as in the training data.
- (b) For each new sample, a classification tree was grown as before with the same values for the tuning parameters. The result was a total of 50 trees.³⁵
- (c) The classifications from each tree were stored.
- (d) For each possible pair of the 50 trees (i.e., 1225), we computed the proportion of times the classifications were the same for common cases.³⁶

8.5.2.1 Some Technical Issues

Recall that the unit of analysis is the case, not the individual. As a result, some individuals are included in the data more than once. Substantively, this is not a problem because judges make sentencing decisions on cases as they come up and repeat offenders, even within short time intervals, are not unusual.

If there are problems, they are technical. First, our stability analysis appropriately sampled cases independently *with* replacement. If sampling were not with replacement, each of the samples would necessarily be identical to the original data and to each other. However, the data can be seen as a random realization of the social processes that generate cases in need of sentences, for which there is, in effect, simple

³⁵We used the same tuning parameter values so that the trees were comparable. We were trying to isolate the impact of new data for a tree of a given complexity. Had we altered the tuning parameter values, changes in how cases were classified could result from either the new data or differing tuning parameters. The cost ratio was also unchanged.

³⁶One reason Kuhnert and Mengersen (2003) suggest additional measures of stability is that they apparently do not find sufficiently strong justification for using a Bayes classifier. In our context, where the relative costs of false negatives and false positives are elicited from stakeholders, their concerns seem far less troubling.

random sampling of cases. Because a case cannot be sentenced more than once, the sampling is *without* replacement.

Second, sampling without replacement *also* is an imperfect formalization for how the data were perhaps generated. It is possible that once a given individual has been selected, there is an increased probability that that individual will be selected again. Perhaps criminal behavior becomes more likely and/or the criminal justice system is primed to arrest and punish that individual. Alternatively, it is possible that the experience of being arrested, convicted, sentenced and sanctioned reduces the probability that an individual will be selected again because criminal behavior is deterred. If the sanction involves effective probation supervision, behavior may be changed for the better. If the sanction involves incarceration, time on the streets (at least in the short to medium term) is reduced.

The first problem implies that we may be underestimating instability, at least a bit. Our effective sample size is smaller than our nominal sample size. The second problem could lead to some underestimation or overestimation of instability depending on how the two competing forces balance for a large number of cases. We will revisit these issues after some results are summarized.

8.5.2.2 Results

Figure 8.2 is a histogram of common classification proportions across pairs of trees. Agreement over trees is very high. The distribution is centered around 90% agreement. The proportions range from around 75% agreement to 100% agreement. One can properly conclude that when our provisional tree is used to arrive at real forecasts, there is relatively little instability. Put another way, the forecasts were on the average about 90% reliable over new realizations of the data.

But it is possible to do even better. Recall that classifications and forecasts are

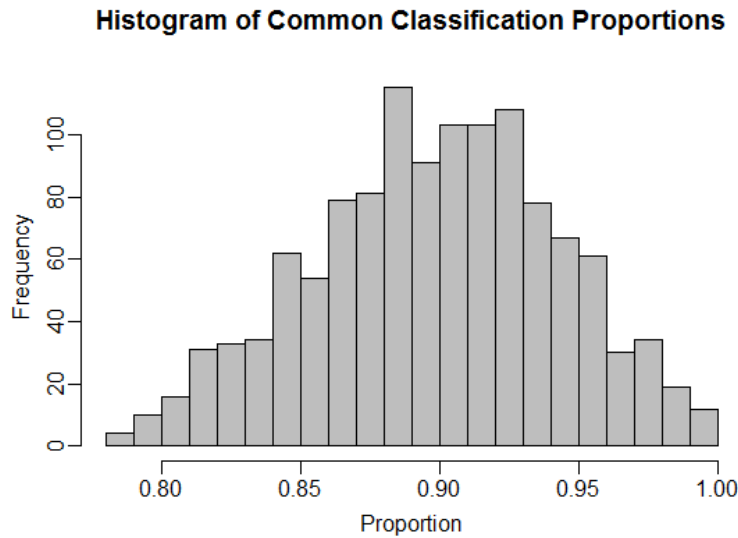


Figure 8.2: Histogram of proportions of common classifications between tree pairs.

derived from the equivalent of votes in each terminal node. It follows that when that vote is close (e.g., 52% v. 48%), there will likely be more instability than when the vote is decisive (e.g., 75% v. 25%). When the vote is close, altering just a few observations can often be enough to swing the vote in the other direction. A close vote indicates that the model is not highly “confident” in the class assigned. It might be sensible, therefore, to flag cases in such terminal nodes as “can’t tell” and perhaps discard them when forecasts are made.

The forecasts from two terminal nodes were discarded as “too close to call.” We began with a “too-close-to-call” definition of .05. Votes within .05 of the .50 threshold were discarded. But because entire nodes had to be either retained or discarded, the .05 rule did not in this instance remove enough unreliable forecasts. When the definition was changed to .07, the results were satisfactory. Thus, forecasts based on votes closer than .43 to .57 were discarded. All other forecasts were retained. With

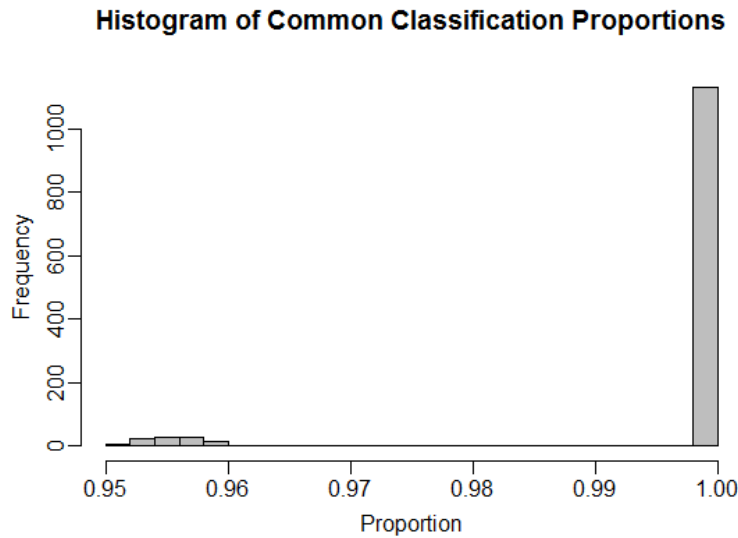


Figure 8.3: Histogram of proportions of common classifications between tree pairs with 14% of data discarded.

other data and different stakeholders, the definition of “unreliable” could well have been quite different. The question for decision makers is how reliable a forecast has to be before it is used to inform sentencing, and that is not our call.

Figure 8.3 shows what happens when the two nodes with votes “too close to call” are not used to generate forecasts. There is nearly 100% agreement. Average agreement is now 99.9%. Average reliability was improved to nearly 100%. The price is that there are no forecasts for about 14% of the cases. But had those forecasts been reported, they could have been deemed too unreliable to use anyway. Overall, these results would have been about the same had the 14% been only 10%, but because unreliable nodes had to be removed in their entirety, all of the cases in them had to be removed as well.

In summary, one might envision providing judges with two kinds of information.

For those profiles leading to highly reliable forecasts, a forecast is made. For those profiles leading to forecasts that are insufficiently reliable, no forecast is made, but those cases would be flagged as such. For the second group, a proper inference is that one cannot decide from the available data and statistical procedures how the individual will perform.³⁷ Therefore, sentences would be determined without quantitative forecasts of risk.

More broadly framed, our stability analysis addresses an important kind of uncertainty in the tree-based, quantitative forecasts. How likely is it that a different forecast would be made with new realizations for the data? With our large sample and small classification tree, there is almost no instability uncertainty in the forecasts for about 85% of the cases. Instability uncertainty would likely be greater in other analyses with substantially smaller samples and/or substantially larger trees. Instability uncertainty would also be affected by the strength of relationships between the predictors and the outcome. As best we can tell, uncertainty in classification tree forecasts has not been addressed in this simple fashion before. We re-emphasize, however, that our concern is with instability in forecasts, not instability in tree structure. The latter can be far more unstable than the former.

One could also imagine providing judges with the votes for each forecast so that the reliability of each forecast could be taken into account. As a practical matter, we thought this was far too heavy a burden to place on the sentencing process, especially because in our illustration at least, most forecasts were highly reliable. But as a technical matter, that information would be available and easily accessed. An outline of the procedure code used for the stability analysis is provided in Appendix A.4.

³⁷Even using the same data, it is likely that a machine learning procedure would do better. Recall, however, that in this paper we are assuming that machine learning is not a practical option.

8.6 Conclusions

The failure outcome for this population is relatively rare: about 15% of the cases fail through an arrest for a violent crime. Nevertheless, even a very simple classification tree is able to correctly identify about 60% of those who fail in the test data. Of those who do not fail, about 67% are correctly identified.

Foretelling accuracy depends especially heavily and directly on the 5 to 1 cost ratio of false negatives to false positives. By design, this opens the door to a substantial number of false positives so that forecasts of failure are correct only about 21% of the time. But another consequence is that forecasting accuracy for individuals who do not fail is over 90%. That is, if a forecast of not failing is made, it is quite likely to be correct. If this kind of information were to be used in sentencing, there would be a large pool of convicted offenders who might be good probation risks. But one must be clear that by “good” risks, we mean with respect to a violent crime only.

It might be more useful in the future to redefine the response, as we have in other applications, to measure three outcomes: an arrest for a violent crime, an arrest for a crime that is not violent, and no arrest at all. Or perhaps some other three-way break would be more appropriate. The point is that by using three outcome classes instead of two, less ham-fisted distinctions can be made. An individual would be forecasted into one of the three outcomes, not one of two. The procedures discussed above would still apply, although the algorithmic output would be a bit more complicated.

We found that the forecasts are quite reliable. Instability does not seem to be a problem for about 85% of the cases. The small tree grown from a large sample no doubt helps a lot. If one discards the relatively small number of cases whose forecasts are deemed unreliable, instability virtually disappears. More generally, the stability analysis provides information on the uncertainty in tree-based forecasts that judges

might find helpful.

Whatever one thinks of the results, it is possible to do substantially better. A high priority is obtaining better data from an appropriate population of offenders. That process has begun. Finally, cost ratios of false negatives to false positives should be obtained from stakeholders. Indeed, several different cost ratios might be relevant. These alone could significantly change the results.

Citation

Berk, R. A. and Bleich, J. (2014). Forecasts of violence to inform sentencing decisions. *Journal of Quantitative Criminology*, 30(1):79-96.

Conclusion

In this thesis, we have presented both methodological and applied advances in ensemble-of-trees methods in machine learning.

In the methodological part, we have provided a number of extensions to the **BART** algorithm. In particular, we introduced a new **R** package **bartMachine** which we hope will greatly increase the accessibility of **BART** to both academics and practitioners alike. Using **bartMachine** as our research engine, we have presented three novel features for the algorithm. First, we introduced a procedure for nonparametric variable selection for **BART** using permutation testing on the variable inclusion proportions. We also presented a simple alteration of **BART**'s prior on splitting rules that allows for informed prior information to be incorporated into the model. Our proposed methodology showed promise in both simulation studies against competing variable selection methods as well as on the application to the yeast regulatory network. We then relaxed the assumption of homoskedasticity in the **BART** model to incorporate parametric models of heteroskedasticity into the model. We found in our simulations that **HBART** was better able to recover the mean function and appropriate credible intervals versus the original implementation. Additionally, **HBART** fared well versus other tree-based Bayesian methods in the motorcycle data example. Finally, we developed a means for incorporating missing data into both the training and test-

ing phases of statistical learning. Our proposed method, **BARTm**, was able to perform well on the MAR, NMAR and pattern-mixture model examples we explored.

As a relatively new algorithm, there is still ample opportunity to further extend **BART**. Given that **BART** has an underlying probability model, it is well-equipped versus other algorithms to explore handling correlated errors across observations, such as in repeated measurement design or longitudinal studies. To date, dealing with correlation structures across observations has been a challenging problem for machine learning. Additional areas to explore for **BART** include examining its performance in extremely high dimensional settings as well as considering **BART**'s performance with respect to probability estimation.

The second part of this thesis explored applications of ensemble-of-trees methods under asymmetric costs. We examined the performance of **RF** and **sgb** in criminology applications. We additionally developed **RF** models that could be used for real-time forecasting of unplanned hospital readmissions in the asymmetric cost framework. Last, we developed stable classification trees for forecasts of violence during probation hearings, a setting where the technology to deploy more sophisticated algorithms may not be readily available. Important future work includes further developing additional ensemble algorithms that incorporate asymmetric costs throughout the entire procedure and not just when applying a threshold to probabilities output from the trained models. In fact, introducing asymmetric costs throughout the entire **BART** fitting procedure is an open question.

Despite having designed highly sophisticated algorithms for prediction, it is ultimately hard to forecast precisely what the future holds for machine learning. At the current juncture in time, both ensemble-of-trees models and deep learning algorithms seem to be at the forefront of the field. As time progresses, it will be exciting to see how the development of future ensemble-of-trees methods unfolds.

A.1 Supplement for Chapter 1

A.1.1 Sampling New Trees

This section provides details on the implementation of Equation 1.13 (steps 1, 3, ..., $2m - 1$), the Metropolis-Hastings step for sampling new trees. Recall from Section 1.4.2 that trees can be altered via growing new child nodes from an existing terminal node, pruning two terminal nodes such that their parent becomes terminal, or changing the splitting rule in a node.

Below is the Metropolis ratio (Gelman et al., 2004, p.291) where the parameter sampled is the tree and the data is the responses unexplained by other trees denoted by \mathbf{R} . We denote the new, proposal tree with an asterisk and the original tree without the asterisk.

$$r = \frac{\mathbb{P}(\mathfrak{T}_* \rightarrow \mathfrak{T}) \mathbb{P}(\mathfrak{T}_* \mid \mathbf{R}, \sigma^2)}{\mathbb{P}(\mathfrak{T} \rightarrow \mathfrak{T}_*) \mathbb{P}(\mathfrak{T} \mid \mathbf{R}, \sigma^2)} \quad (\text{A.1})$$

We accept a draw from the posterior distribution of trees if a draw from a standard uniform distribution is less than the value of r . Immediately we note that it is difficult (if not impossible) to calculate the posterior probabilities for the trees themselves. Instead, we employ Bayes' Rule,

$$\mathbb{P}(\mathfrak{T} \mid \mathbf{R}, \sigma^2) = \frac{\mathbb{P}(\mathbf{R} \mid \mathfrak{T}, \sigma^2) \mathbb{P}(\mathfrak{T} \mid \sigma^2)}{\mathbb{P}(\mathbf{R} \mid \sigma^2)}, \quad (\text{A.2})$$

and plug the result into Equation A.1 to obtain:

$$r = \underbrace{\frac{\mathbb{P}(\mathfrak{T}_* \rightarrow \mathfrak{T})}{\mathbb{P}(\mathfrak{T} \rightarrow \mathfrak{T}_*)}}_{\text{transition ratio}} \times \underbrace{\frac{\mathbb{P}(\mathbf{R} \mid \mathfrak{T}_*, \sigma^2)}{\mathbb{P}(\mathbf{R} \mid \mathfrak{T}, \sigma^2)}}_{\text{likelihood ratio}} \times \underbrace{\frac{\mathbb{P}(\mathfrak{T}_*)}{\mathbb{P}(\mathfrak{T})}}_{\text{tree structure ratio}}. \quad (\text{A.3})$$

Note that the probability of the tree structure is independent of σ^2 .

The goal of this section is to explicitly calculate r for all possible tree proposals — GROW, PRUNE and CHANGE. For each proposal, the calculations are organized into separate sections detailing each of the three ratios — transition, likelihood and tree structure. Note that our actual implementation uses the following expressions in log form for numerical accuracy.

A.1.2 Grow Proposal

Transition Ratio

Transitioning from the original tree to a new tree involves growing two child nodes from a current terminal node:

$$\begin{aligned}
\mathbb{P}\left(\mathfrak{T} \rightarrow \mathfrak{T}_*\right) &= \mathbb{P}(\text{GROW}) \mathbb{P}(\text{selecting } \eta \text{ to grow from}) \times \\
&\quad \mathbb{P}(\text{selecting the } j\text{th attribute to split on}) \times \\
&\quad \mathbb{P}(\text{selecting the } i\text{th value to split on}) \\
&= \mathbb{P}(\text{GROW}) \frac{1}{b} \frac{1}{p_{\text{adj}}(\eta)} \frac{1}{n_{j \cdot \text{adj}}(\eta)}.
\end{aligned} \tag{A.4}$$

We chose one of the current b terminal nodes which we denote the η th node, and then we pick an attribute and split point. $p_{\text{adj}}(\eta)$ denotes the number of predictors left available to split on. This can be less than p if certain predictors do not have two or more unique values once the data reaches the η th node. For example, this regularly occurs if a dummy variable was split on in some node higher up in the lineage. $n_{j \cdot \text{adj}}(\eta)$ denotes the number of *unique* values left in the p th attribute after adjusting for parents' splits.

Transitioning from the new tree back to the original tree involves pruning that node:

$$\mathbb{P}\left(\mathfrak{T}_* \rightarrow \mathfrak{T}\right) = \mathbb{P}(\text{PRUNE}) \mathbb{P}(\text{selecting } \eta \text{ to prune from}) = \mathbb{P}(\text{PRUNE}) \frac{1}{w_2^*} \tag{A.5}$$

where w_2^* denotes the number of second generation internal nodes (nodes with two terminal child nodes) in the new tree. Thus, the full transition ratio is:

$$\frac{\mathbb{P}\left(\mathfrak{T}_* \rightarrow \mathfrak{T}\right)}{\mathbb{P}\left(\mathfrak{T} \rightarrow \mathfrak{T}_*\right)} = \frac{\mathbb{P}(\text{PRUNE})}{\mathbb{P}(\text{GROW})} \frac{b p_{\text{adj}}(\eta) n_{j \cdot \text{adj}}(\eta)}{w_2^*}. \tag{A.6}$$

Note that when there are no variables with more two or more unique values, the

probability of GROW is set to zero and the step will be automatically rejected.

Likelihood Ratio

To calculate the likelihood, the tree structure determines which responses fall into which of the b terminal nodes. Thus,

$$\mathbb{P}\left(R_1, \dots, R_n \mid \mathbb{F}, \sigma^2\right) = \prod_{\ell=1}^b \mathbb{P}\left(R_{\ell_1}, \dots, R_{\ell_{n_\ell}} \mid \sigma^2\right) \quad (\text{A.7})$$

where each term on the right hand side is the probability of responses in one of the b terminal nodes, which are independent by assumption. The R_ℓ 's denote the data in the ℓ th terminal node and where n_ℓ denotes how many observations are in each terminal node and $n = \sum_{\ell=1}^b n_\ell$.

We now find an analytic expression for the node likelihood term. Remember, if the mean in each terminal node, which we denote μ_ℓ , was known, then we would have $R_{\ell_1}, \dots, R_{\ell_{n_\ell}} \mid \mu_\ell, \sigma^2 \stackrel{iid}{\sim} \mathcal{N}(\mu_\ell, \sigma^2)$. BART requires μ_ℓ to be margined out, allowing the Gibbs sampler in Equation 1.13 to avoid dealing with jumping between continuous spaces of varying dimensions (Chipman et al., 2010, page 275). Recall that one of the BART model assumptions is a prior on the average value of $\mu \sim \mathcal{N}(0, \sigma_\mu^2)$ and thus,

$$\mathbb{P}\left(R_{\ell_1}, \dots, R_{\ell_{n_\ell}} \mid \sigma^2\right) = \int_{\mathbb{R}} \mathbb{P}\left(R_{\ell_1}, \dots, R_{\ell_{n_\ell}} \mid \mu_\ell, \sigma^2\right) \mathbb{P}\left(\mu_\ell; \sigma_\mu^2\right) d\mu_\ell \quad (\text{A.8})$$

which can be shown via completion of the square or convolution to be

$$\mathbb{P}\left(R_{\ell_1}, \dots, R_{\ell_{n_\ell}} \mid \sigma^2\right) = \frac{1}{(2\pi\sigma^2)^{n_\ell/2}} \sqrt{\frac{\sigma^2}{\sigma^2 + n_\ell\sigma_\mu^2}} \times \quad (\text{A.9})$$

$$\exp \left(-\frac{1}{2\sigma^2} \left(\sum_{i=1}^{n_\ell} (R_{\ell_i} - \bar{R}_\ell)^2 - \frac{\bar{R}_\ell^2 n_\ell^2}{n_\ell + \frac{\sigma^2}{\sigma_\mu^2}} + n_\ell \bar{R}_\ell^2 \right) \right)$$

where \bar{R}_ℓ denotes the mean response in the node and R_{ℓ_i} denotes the observations $i = 1 \dots n_\ell$ in the node.

Since the likelihoods are solely determined by the terminal nodes, the proposal tree differs from the original tree by only the selected node to be grown, denoted by ℓ , which becomes two children after the GROW step denoted by ℓ_L and ℓ_R . Hence, the likelihood ratio becomes:

$$\frac{\mathbb{P}(\mathbf{R} \mid \mathfrak{F}_*, \sigma^2)}{\mathbb{P}(\mathbf{R} \mid \mathfrak{F}, \sigma^2)} = \frac{\mathbb{P}(R_{\ell_{L,1}}, \dots, R_{\ell_{L,n_{\ell,L}}} \mid \sigma^2) \mathbb{P}(R_{\ell_{R,1}}, \dots, R_{\ell_{R,n_{\ell,R}}} \mid \sigma^2)}{\mathbb{P}(R_{\ell_1}, \dots, R_{\ell_{n_\ell}} \mid \sigma^2)} \quad (\text{A.10})$$

Plugging Equation A.9 into Equation A.10 three times yields the ratio for the GROW step:

$$\begin{aligned} & \sqrt{\frac{\sigma^2 (\sigma^2 + n_\ell \sigma_\mu^2)}{(\sigma^2 + n_{\ell_L} \sigma_\mu^2) (\sigma^2 + n_{\ell_R} \sigma_\mu^2)}} \\ & \times \exp \left(\frac{\sigma_\mu^2}{2\sigma^2} \left(\frac{(\sum_{i=1}^{n_{\ell_L}} R_{\ell_{L,i}})^2}{\sigma^2 + n_{\ell_L} \sigma_\mu^2} + \frac{(\sum_{i=1}^{n_{\ell_R}} R_{\ell_{R,i}})^2}{\sigma^2 + n_{\ell_R} \sigma_\mu^2} - \frac{(\sum_{i=1}^{n_\ell} R_{\ell,i})^2}{\sigma^2 + n_\ell \sigma_\mu^2} \right) \right) \end{aligned} \quad (\text{A.11})$$

where n_{ℓ_L} and n_{ℓ_R} denote the number of data points in the newly grown left and right child nodes.

Tree Structure Ratio

In Section 1.4.1 we discussed the prior on the tree structure (where the splits occur) as well as the tree rules. For the entire tree,

$$\mathbb{P}\left(\frac{\mathfrak{F}}{\mathfrak{F}}\right) = \prod_{\eta \in H_{\text{terminals}}} (1 - \mathbb{P}_{\text{SPLIT}}(\eta)) \prod_{\eta \in H_{\text{internals}}} \mathbb{P}_{\text{SPLIT}}(\eta) \prod_{\eta \in H_{\text{internals}}} \mathbb{P}_{\text{RULE}}(\eta) \quad (\text{A.12})$$

where $H_{\text{terminals}}$ denotes the set of terminal nodes and $H_{\text{internals}}$ denotes the internal nodes.

Recall that the probability of splitting on a given node η is $\mathbb{P}_{\text{SPLIT}}(\eta) = \alpha / (1 + d_\eta)^\beta$. The probability is controlled by two hyperparameters, α and β , and d_η is the depth (number of parent generations) of node η . When assigning a rule, recall that **BART** picks from all available attributes and then from all available unique split points. Using the notation from the transition ratio section, $\mathbb{P}_{\text{RULE}}(\eta) = 1/p_{\text{adj}}(\eta) \times 1/n_{j \cdot \text{adj}}(\eta)$.

Once again, the original tree features a node η that was selected to be grown. The proposal tree differs with two child nodes denoted η_L and η_R . We can now form the ratio:

$$\begin{aligned} \frac{\mathbb{P}\left(\frac{\mathfrak{F}_*}{\mathfrak{F}_*}\right)}{\mathbb{P}\left(\frac{\mathfrak{F}}{\mathfrak{F}}\right)} &= \frac{(1 - \mathbb{P}_{\text{SPLIT}}(\eta_L))(1 - \mathbb{P}_{\text{SPLIT}}(\eta_R)) \mathbb{P}_{\text{SPLIT}}(\eta) \mathbb{P}_{\text{RULE}}(\eta)}{(1 - \mathbb{P}_{\text{SPLIT}}(\eta))} \quad (\text{A.13}) \\ &= \frac{\left(1 - \frac{\alpha}{(1 + d_{\eta_L})^\beta}\right) \left(1 - \frac{\alpha}{(1 + d_{\eta_R})^\beta}\right) \frac{\alpha}{(1 + d_\eta)^\beta} \frac{1}{p_{\text{adj}}(\eta)} \frac{1}{n_{j \cdot \text{adj}}(\eta)}}{1 - \frac{\alpha}{(1 + d_\eta)^\beta}} \\ &= \alpha \frac{\left(1 - \frac{\alpha}{(2 + d_\eta)^\beta}\right)^2}{\left((1 + d_\eta)^\beta - \alpha\right) p_{\text{adj}}(\eta) n_{j \cdot \text{adj}}(\eta)} \end{aligned}$$

The last line follows from algebra and using the fact that the depth of the grown nodes is the depth of the parent node incremented by one ($d_{\eta_L} = d_{\eta_R} = d_\eta + 1$).

A.1.3 Prune Proposal

A prune proposal is the “opposite” of a grow proposal. Prune selects a node with two children and removes them. Thus, each ratio will be approximately the inverse of the ratios found in the previous section concerning the grow proposal. Note also that prune steps are not considered in trees that consist of a single root node.

Transition Ratio

We begin with transitioning from the original tree to the proposal tree:

$$\mathbb{P}\left(\mathfrak{T} \rightarrow \mathfrak{T}_*\right) = \mathbb{P}(\text{PRUNE}) \mathbb{P}(\text{selecting } \eta \text{ to prune from}) = \mathbb{P}(\text{PRUNE}) \frac{1}{w_2} \quad (\text{A.14})$$

where w_2 denotes the number of parent nodes that have two children but no grandchildren. To transition in the opposite direction, we are obligated to grow from node η . This is similar to Equation A.4 except the proposed tree has one less terminal node due to the pruning of the original tree, resulting in a $1/(b-1)$ term:

$$\mathbb{P}\left(\mathfrak{T}_* \rightarrow \mathfrak{T}\right) = \mathbb{P}(\text{GROW}) \frac{1}{b-1} \frac{1}{p_{\text{adj}}(\eta^*)} \frac{1}{n_{j^* \cdot \text{adj}}(\eta^*)}. \quad (\text{A.15})$$

Thus, the transition ratio is:

$$\frac{\mathbb{P}\left(\mathfrak{T}_* \rightarrow \mathfrak{T}\right)}{\mathbb{P}\left(\mathfrak{T} \rightarrow \mathfrak{T}_*\right)} = \frac{\mathbb{P}(\text{GROW})}{\mathbb{P}(\text{PRUNE})} \frac{w_2}{(b-1)p_{\text{adj}}(\eta^*)n_{j^* \cdot \text{adj}}(\eta^*)}. \quad (\text{A.16})$$

Likelihood Ratio

This is simply the inverse of the likelihood ratio for the grow proposal:

$$\frac{\mathbb{P}(\mathbf{R} \mid \mathfrak{F}_*, \sigma^2)}{\mathbb{P}(\mathbf{R} \mid \mathfrak{F}, \sigma^2)} = \sqrt{\frac{(\sigma^2 + n_{\ell_L} \sigma_\mu^2)(\sigma^2 + n_{\ell_R} \sigma_\mu^2)}{\sigma^2(\sigma^2 + n_\ell \sigma_\mu^2)}} \times \exp\left(\frac{\sigma_\mu^2}{2\sigma^2} \left(\frac{(\sum_{i=1}^{n_\ell} R_{\ell,i})^2}{\sigma^2 + n_\ell \sigma_\mu^2} - \frac{(\sum_{i=1}^{n_{\ell_L}} R_{\ell_L,i})^2}{\sigma^2 + n_{\ell_L} \sigma_\mu^2} - \frac{(\sum_{i=1}^{n_{\ell_R}} R_{\ell_R,i})^2}{\sigma^2 + n_{\ell_R} \sigma_\mu^2} \right)\right). \quad (\text{A.17})$$

Tree Structure Ratio

This is also simply the inverse of the tree structure ratio for the grow proposal:

$$\frac{\mathbb{P}(\mathfrak{F}_*)}{\mathbb{P}(\mathfrak{F})} = \frac{\left((1 + d_\eta)^\beta - \alpha\right) p_{\text{adj}}(\eta^*) n_{j^* \cdot \text{adj}}(\eta^*)}{\alpha \left(1 - \frac{\alpha}{(2 + d_\eta)^\beta}\right)^2}. \quad (\text{A.18})$$

A.1.4 Change

A change proposal involves picking an internal node and changing its rule by picking both a new available predictor to split on and a new valid split value among values of the selected predictor. Although this could be implemented for use in any internal node in the tree, for simplicity we limit our implementation to *singly* internal nodes: those that have two terminal child nodes and thus, no grand-children.

Transition Ratio

The transition to a proposal tree is below:

$$\begin{aligned}
\mathbb{P}\left(\mathfrak{F} \rightarrow \mathfrak{F}_*\right) &= \mathbb{P}(\text{CHANGE}) \mathbb{P}(\text{selecting node } \eta \text{ to change}) \times \quad (\text{A.19}) \\
&\quad \mathbb{P}(\text{selecting the new attribute to split on}) \times \\
&\quad \mathbb{P}(\text{selecting the new value to split on})
\end{aligned}$$

When calculating the ratio, the first three terms are shared in both numerator and denominator. The probability of selecting the new value to split on will differ as different split features have different numbers of unique values available. Thus we are left with

$$\frac{\mathbb{P}\left(\mathfrak{F}_* \rightarrow \mathfrak{F}\right)}{\mathbb{P}\left(\mathfrak{F} \rightarrow \mathfrak{F}_*\right)} = \frac{n_{j^* \cdot \text{adj}}(\eta^*)}{n_{j \cdot \text{adj}}(\eta)} \quad (\text{A.20})$$

where $n_{j^* \cdot \text{adj}}(\eta^*)$ is the number of split values available under the proposal tree's splitting rule and $n_{j \cdot \text{adj}}(\eta)$ is the number of split values available under the original tree's splitting rule.

Likelihood Ratio

The proposal tree differs from the original tree only in the two child nodes of the selected change node. These two terminal nodes have the unexplained responses apportioned differently. Denote R_1 as the residuals of the first child node and R_2 as the unexplained responses in the second daughter node. Thus we begin with:

$$\frac{\mathbb{P}\left(\mathbf{R} \mid \mathfrak{F}_*, \sigma^2\right)}{\mathbb{P}\left(\mathbf{R} \mid \mathfrak{F}, \sigma^2\right)} = \frac{\mathbb{P}\left(R_{1^*,1}, \dots, R_{1^*,n_{1^*}} \mid \sigma^2\right) \mathbb{P}\left(R_{2^*,1}, \dots, R_{2^*,n_{2^*}} \mid \sigma^2\right)}{\mathbb{P}\left(R_{1,1}, \dots, R_{1,n_1} \mid \sigma^2\right) \mathbb{P}\left(R_{2,1}, \dots, R_{2,n_2} \mid \sigma^2\right)} \quad (\text{A.21})$$

where the responses denoted with an asterisk are the responses in the proposal tree's child nodes.

Substituting Equation A.9 four times and using algebra, the following expression is obtained for the ratio:

$$\sqrt{\frac{\left(\frac{\sigma_\mu^2}{\sigma_\mu^2} + n_1\right) \left(\frac{\sigma_\mu^2}{\sigma_\mu^2} + n_2\right)}{\left(\frac{\sigma_\mu^2}{\sigma_\mu^2} + n_1^*\right) \left(\frac{\sigma_\mu^2}{\sigma_\mu^2} + n_2^*\right)}} \times \exp\left(\frac{1}{2\sigma^2} \left(\frac{(\sum_{i=1}^{n_1^*} R_{1^*,i})^2}{n_1^* + \frac{\sigma_\mu^2}{\sigma_\mu^2}} + \frac{(\sum_{i=1}^{n_2^*} R_{2^*,i})^2}{n_2^* + \frac{\sigma_\mu^2}{\sigma_\mu^2}} - \frac{(\sum_{i=1}^{n_1} R_{1,i})^2}{n_1 + \frac{\sigma_\mu^2}{\sigma_\mu^2}} - \frac{(\sum_{i=1}^{n_2} R_{2,i})^2}{n_2 + \frac{\sigma_\mu^2}{\sigma_\mu^2}} \right)\right) \quad (\text{A.22})$$

which simplifies to

$$\exp\left(\frac{1}{2\sigma^2} \left(\frac{(\sum_{i=1}^{n_1^*} R_{1^*,i})^2 - (\sum_{i=1}^{n_1^*} R_{1,i})^2}{n_1 + \frac{\sigma_\mu^2}{\sigma_\mu^2}} + \frac{(\sum_{i=1}^{n_2^*} R_{2^*,i})^2 - (\sum_{i=1}^{n_2^*} R_{2,i})^2}{n_2 + \frac{\sigma_\mu^2}{\sigma_\mu^2}} \right)\right) \quad (\text{A.23})$$

if the number of responses in the children do not change in the proposal ($n_1 = n_1^*$ and $n_2 = n_2^*$).

Tree Structure Ratio

The proposal tree has the same structure as the original tree. Thus we only need to take into account the changed node's children:

$$\frac{\mathbb{P}\left(\frac{\mathfrak{P}}{\mathfrak{F}}\right)}{\mathbb{P}\left(\frac{\mathfrak{P}}{\mathfrak{F}}\right)} = \frac{(1 - \mathbb{P}_{\text{SPLIT}}(\eta_{1^*})) (1 - \mathbb{P}_{\text{SPLIT}}(\eta_{2^*})) \mathbb{P}_{\text{SPLIT}}(\eta_*) \mathbb{P}_{\text{RULE}}(\eta_*)}{(1 - \mathbb{P}_{\text{SPLIT}}(\eta_1)) (1 - \mathbb{P}_{\text{SPLIT}}(\eta_2)) \mathbb{P}_{\text{SPLIT}}(\eta) \mathbb{P}_{\text{RULE}}(\eta)}. \quad (\text{A.24})$$

The probability of splits remain the same because the child nodes are at the same depths. Thus we only need to consider the ratio of the probability of the rules. Once

again, the probability of selecting the new value to split on will differ as different split features have different numbers of unique values available. We are left with

$$\frac{\mathbb{P}\left(\frac{\mathfrak{P}}{\mathfrak{I}}_*\right)}{\mathbb{P}\left(\frac{\mathfrak{P}}{\mathfrak{I}}\right)} = \frac{n_{j \cdot \text{adj}}(\eta)}{n_{j^* \cdot \text{adj}}(\eta^*)}. \quad (\text{A.25})$$

Note that this is the inverse of the transition ratio. Hence, for the change step, only the likelihood ratio needs to be computed to determine the Metropolis-Hastings ratio r .

A.2 Supplement for Chapter 3

A.2.1 Pseudo-code for Variable Selection Procedures

Algorithm 4 Local Threshold Procedure

```
Compute  $p_1, \dots, p_K$  ▷ Inclusion proportions from original data  
for  $i \leftarrow \{1, \dots, P\}$  do ▷  $P$  is the number of null permutations  
     $\mathbf{y}^* \leftarrow \text{Permute}(\mathbf{y})$   
    Run BART using  $\mathbf{y}^*$  as response  
    Compute  $p_{i1}^*, \dots, p_{iK}^*$  ▷ Inclusion proportions from permuted data  
end for  
for  $j \leftarrow \{1, \dots, K\}$  do  
     $q_j^* \leftarrow \text{Quantile}(p_{1j}^*, \dots, p_{Pj}^*, 1 - \alpha)$  ▷  $1 - \alpha$  quantile of  $\mathbf{x}_j$  permutation  
distribution  
    if  $p_j > q_j^*$  then Include  $p_j$  in Vars end if  
end for  
return Vars
```

Algorithm 5 Global Maximum Threshold Procedure

Compute p_1, \dots, p_K ▷ Inclusion proportions from original data

for $i \leftarrow \{1, \dots, P\}$ **do** ▷ P is the number of null permutations

$\mathbf{y}^* \leftarrow \text{Permute}(\mathbf{y})$

Run BART using \mathbf{y}^* as response

Compute $p_{i1}^*, \dots, p_{iK}^*$ ▷ Inclusion proportions from permuted data

$g_i \leftarrow \text{Max}(p_{i1}^*, \dots, p_{iK}^*)$ ▷ Maximum of proportions from permuted data

end for

$g^* \leftarrow \text{Quantile}(g_1, \dots, g_P, 1 - \alpha)$ ▷ $1 - \alpha$ Quantile of maxima

for $j \leftarrow \{1, \dots, K\}$ **do**

if $p_j > g^*$ **then** Include p_j in Vars **end if**

end for

return Vars

Algorithm 6 Global Standard Error Threshold Procedure

Compute p_1, \dots, p_K ▷ Inclusion proportions from original data

for $i \leftarrow \{1, \dots, P\}$ **do** ▷ P is the number of null permutations

$\mathbf{y}^* \leftarrow \text{Permute}(\mathbf{y})$

Run BART using \mathbf{y}^* as response

Compute $p_{i1}^*, \dots, p_{iK}^*$ ▷ Inclusion proportions from permuted data

end for

for $j \leftarrow \{1, \dots, K\}$ **do**

$m_j \leftarrow \text{Avg}(p_{1j}^*, \dots, p_{Pj}^*)$ ▷ Sample average of \mathbf{x}_j permutation distribution

$s_j \leftarrow \text{SD}(p_{1j}^*, \dots, p_{Pj}^*)$ ▷ Sample sd of \mathbf{x}_j permutation distribution

end for

$C^* \leftarrow \inf_{C \in \mathbb{R}^+} \left\{ \forall j, \quad \frac{1}{P} \sum_{i=1}^P \mathbb{1}_{p_{ij}^* \leq m_j + C \cdot s_j} > 1 - \alpha \right\}$ ▷ Simultaneous coverage

for $j \leftarrow \{1, \dots, K\}$ **do**

if $p_j > m_j + C^* \cdot s_j$ **then** Include p_j in Vars **end if**

end for

return Vars

Algorithm 7 Cross-Validated Comparison of Threshold Procedures

Divide the data into K training-test splits

for $k \leftarrow \{1, \dots, K\}$ **do**

for method $\leftarrow \{\text{Local, Global Maximum, Global SE}\}$ **do**

$\text{Var}_{\text{method}} \leftarrow$ Selected variables using method on BART

$\text{BART}_{\text{method}} \leftarrow$ BART built from k^{th} training set using only $\text{Var}_{\text{method}}$

$L2_{k,\text{method}} \leftarrow$ $L2$ error from $\text{BART}_{\text{method}}$ on k^{th} test set

end for

end for

for method $\leftarrow \{\text{Local, Global Maximum, Global SE}\}$ **do**

$L2_{\text{method}} \leftarrow \sum_{k=1}^K L2_{k,\text{method}}$ \triangleright Aggregate $L2$ error over entire training set

end for

method* $\leftarrow \arg \min_{\text{method}} \{L2_{\text{method}}\}$ \triangleright Choose the best method from the three

return Selected variables using method* on BART on full training set

A.3 Supplement for Chapter 5

A.3.0.1 Gibbs Sampling Details for HBART

In the following sections, we provide implementation details for each step of the HBART Gibbs sampler (Equation 5.5).

Drawing σ^2 (step $2m + 1$)

Drawing σ^2 in HBART requires a slight modification to the scheme used in the original homoskedastic BART. With the errors distributed multivariate normal (Equation 5.2) and the prior on σ^2 being distributed inverse-gamma, conjugacy yields the posterior as an inverse-gamma distribution just as in BART. The quadratic form of the errors and their covariance matrix carries into the scale parameter. Hence, we sample σ^2 as

$$\begin{aligned}\sigma^2 &\sim \text{InvGamma}\left(\frac{\nu + n}{2}, \frac{\nu\lambda + \boldsymbol{\mathcal{E}}^T \mathbf{D}^{-1} \boldsymbol{\mathcal{E}}}{2}\right) \\ &= \text{InvGamma}\left(\frac{\nu + n}{2}, \frac{\nu\lambda}{2} + \frac{1}{2} \left(\sum_{i=1}^n \exp(-\mathbf{z}_i^\top \boldsymbol{\gamma}) \mathcal{E}_i^2 \right)\right).\end{aligned}\tag{A.26}$$

The default values of hyperparameters ν and q are set to be the same as the defaults of the original homoskedastic BART algorithm.

Drawing $\boldsymbol{\gamma}$ (step $2m + 2$)

The posterior of $\boldsymbol{\gamma} \mid \mathcal{E}_1, \dots, \mathcal{E}_n, \sigma^2$ is proportional to:

$$\mathbb{P}(\boldsymbol{\gamma} \mid \mathcal{E}_1, \dots, \mathcal{E}_n, \sigma^2) \propto \mathbb{P}(\mathcal{E}_1, \dots, \mathcal{E}_n \mid \boldsymbol{\gamma}, \sigma^2) \mathbb{P}(\boldsymbol{\gamma})\tag{A.27}$$

$$\begin{aligned}
&= \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2\exp(\mathbf{z}_i^\top \boldsymbol{\gamma})}} \exp\left(-\frac{1}{2\sigma^2\exp(\mathbf{z}_i^\top \boldsymbol{\gamma})} \mathcal{E}_i^2\right) \times \\
&\quad \prod_{j=1}^p \frac{1}{\sqrt{2\pi\Sigma_{jj}}} \exp\left(-\frac{1}{2\Sigma_{jj}} (\gamma_j - \gamma_{0j})^2\right) \\
&\propto \frac{1}{\sqrt{\sigma^2\exp(\sum_{i=1}^n \mathbf{z}_i^\top \boldsymbol{\gamma})}} \times \\
&\quad \exp\left(-\frac{1}{2} \left(\sum_{i=1}^n \frac{\mathcal{E}_i^2}{\sigma^2\exp(\mathbf{z}_i^\top \boldsymbol{\gamma})} + \sum_{j=1}^p \frac{(\gamma_j - \gamma_{0j})^2}{\Sigma_{jj}} \right)\right)
\end{aligned}$$

Since it is not tractable to draw directly from the above distribution, a proposal distribution is required. A suitable proposal distribution can be obtained using Gamerman (1997), which is based on a single step of an iteratively reweighted least squares algorithm. Letting $\boldsymbol{\gamma}$ represent the current value of the parameter vector, the recommended proposal density is $\mathbb{P}(\boldsymbol{\gamma}^*|\boldsymbol{\gamma}) = \mathcal{N}_p(\mathbf{a}(\boldsymbol{\gamma}), \mathbf{B})$ where

$$\begin{aligned}
\mathbf{a} &= \mathbf{B} \left(\boldsymbol{\Sigma}^{-1} \boldsymbol{\gamma}_0 + \frac{1}{2} \mathbf{Z}^T \mathbf{w} \right), \quad \mathbf{B} = \left(\boldsymbol{\Sigma}^{-1} + \frac{1}{2} \mathbf{Z}^T \mathbf{Z} \right)^{-1} \text{ and} \\
\mathbf{w} &= \left[\mathbf{z}_1^\top \boldsymbol{\gamma} + \frac{\mathcal{E}_1^2}{\sigma^2 \exp(\mathbf{z}_1^\top \boldsymbol{\gamma})} - 1 \dots \mathbf{z}_n^\top \boldsymbol{\gamma} + \frac{\mathcal{E}_n^2}{\sigma^2 \exp(\mathbf{z}_n^\top \boldsymbol{\gamma})} - 1 \right]^\top.
\end{aligned} \tag{A.28}$$

Then, $\boldsymbol{\gamma}^*$ is accepted if a draw from a standard uniform distribution is less than the Metropolis-Hastings ratio (Gelman et al., 2004, p.291),

$$r = \underbrace{\frac{\mathbb{P}(\boldsymbol{\gamma}^*|\boldsymbol{\gamma})}{\mathbb{P}(\boldsymbol{\gamma}|\boldsymbol{\gamma}^*)}}_{\text{jump ratio}} \underbrace{\frac{\mathbb{P}(\boldsymbol{\gamma}^* | \sigma^2, \mathcal{E}_1, \dots, \mathcal{E}_n)}{\mathbb{P}(\boldsymbol{\gamma} | \sigma^2, \mathcal{E}_1, \dots, \mathcal{E}_n)}}_{\text{likelihood ratio}}. \tag{A.29}$$

The *jump ratio* in Equation A.29 becomes

$$\begin{aligned}
\frac{\mathbb{P}(\gamma^*|\gamma)}{\mathbb{P}(\gamma|\gamma^*)} &= \frac{(2\pi)^{-\frac{p}{2}}|\mathbf{B}|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\gamma^* - a(\gamma))^{\top}\mathbf{B}^{-1}(\gamma^* - a(\gamma))\right)}{(2\pi)^{-\frac{p}{2}}|\mathbf{B}|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\gamma - a(\gamma^*))^{\top}\mathbf{B}^{-1}(\gamma - a(\gamma^*))\right)} \\
&= \exp\left(\frac{1}{2}(\gamma - \gamma^* + a(\gamma) - a(\gamma^*))^{\top}\mathbf{B}^{-1}(\gamma - \gamma^* + a(\gamma) - a(\gamma^*))\right),
\end{aligned} \tag{A.30}$$

the *likelihood ratio* in Equation A.29 is calculated to be

$$\begin{aligned}
\frac{\mathbb{P}(\gamma^* | \sigma^2, \mathcal{E}_1, \dots, \mathcal{E}_n)}{\mathbb{P}(\gamma | \sigma^2, \mathcal{E}_1, \dots, \mathcal{E}_n)} &= \frac{\prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2\exp(\mathbf{z}_i^{\top}\gamma^*)}}\exp\left(-\frac{1}{2\sigma^2\exp(\mathbf{z}_i^{\top}\gamma^*)}\mathcal{E}_i^2\right)}{\prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2\exp(\mathbf{z}_i^{\top}\gamma)}}\exp\left(-\frac{1}{2\sigma^2\exp(\mathbf{z}_i^{\top}\gamma)}\mathcal{E}_i^2\right)} \\
&\quad \times \frac{\prod_{j=1}^p \frac{1}{\sqrt{2\pi\Sigma_{jj}}}\exp\left(-\frac{1}{2\Sigma_{jj}}(\gamma_j^* - \gamma_{0j})^2\right)}{\prod_{j=1}^p \frac{1}{\sqrt{2\pi\Sigma_{jj}}}\exp\left(-\frac{1}{2\Sigma_{jj}}(\gamma_j - \gamma_{0j})^2\right)} \\
&= \exp\left(\frac{1}{2}\left(\sum_{i=1}^n \mathbf{z}_i^{\top}(\gamma - \gamma^*)\right.\right. \\
&\quad \left.+\frac{1}{\sigma^2}\left(\sum_{i=1}^n \mathcal{E}_i^2\left(\exp(-\mathbf{z}_i^{\top}\gamma) - \exp(-\mathbf{z}_i^{\top}\gamma^*)\right)\right)\right) \\
&\quad \left.+\sum_{j=1}^p \frac{1}{\Sigma_{jj}}\left(\gamma_j^2 - \gamma_j^{*2} + 2\gamma_{0j}(\gamma_j^* - \gamma_j)\right)\right)
\end{aligned} \tag{A.31}$$

which results in the Metropolis-Hastings ratio of Equation A.29 of

$$\begin{aligned}
r &= \exp\left(\frac{1}{2}\left((\gamma - \gamma^* + a(\gamma) - a(\gamma^*))^{\top}\mathbf{B}^{-1}(\gamma - \gamma^* + a(\gamma) - a(\gamma^*))\right.\right. \\
&\quad \left.+\sum_{i=1}^n \mathbf{z}_i^{\top}(\gamma - \gamma^*) + \frac{1}{\sigma^2}\sum_{i=1}^n \mathcal{E}_i^2\left(\exp(-\mathbf{z}_i^{\top}\gamma) - \exp(-\mathbf{z}_i^{\top}\gamma^*)\right)\right) \\
&\quad \left.+\sum_{j=1}^p \frac{1}{\Sigma_{jj}}\left(\gamma_j^2 - \gamma_j^{*2} + 2\gamma_{0j}(\gamma_j^* - \gamma_j)\right)\right).
\end{aligned} \tag{A.32}$$

We choose default values for the hyperparameters $\gamma_{0,1}, \dots, \gamma_{0,p}$ and $\Sigma_{11}, \dots, \Sigma_{pp}$. All $\gamma_{0,j}$ are set to 0. This choice centers the prior distribution of the linear model coefficients at zero. For the variance hyperparameters, we choose the Σ_{jj} 's to be 1000 for all j which is sufficiently large so our model will not impose shrinkage of the coefficients towards 0. Investigating this parameter's role in our algorithm we view as fruitful future work.

Drawing the \mathcal{L} 's (steps 2, 4, \dots , $2m$)

Sampling the leaf parameters must be adjusted to reflect the heteroskedasticity in the model. Observations considered highly variable will now *be downweighted* when constructing an estimate of a leaf node's prediction. Recall that $\mathcal{L}_t \mid \mathfrak{F}_1, \mathbf{R}_1, \sigma_1^2, \dots, \sigma_n^2$ is the sampling for all leaves where each leaf is considered independent,

$$\begin{aligned} \mu_{t1} & \mid \mathfrak{F}_t, \mathbf{R}_{t_1}, \sigma_1^2, \dots, \sigma_n^2 \\ \mu_{t2} & \mid \mathfrak{F}_t, \mathbf{R}_{t_2}, \sigma_1^2, \dots, \sigma_n^2 \\ & \vdots \\ \mu_{tb_t} & \mid \mathfrak{F}_t, \mathbf{R}_{t_{b_t}}, \sigma_1^2, \dots, \sigma_n^2 \end{aligned} \tag{A.33}$$

where b_t denotes the number of terminal nodes in the t th tree and the subscripts on the R_{t_i} terms denote only the data that is apportioned to the specific terminal node. Recall that the prior on the leaf parameters, μ 's, are normal and the likelihood of the responses, R_t 's, are assumed normal as well.

Given the normal-normal conjugacy, we derive the normal posterior distribution for a given leaf's prediction which is necessarily a function of the heterogeneous variances. We drop the subscripts on the R term for convenience and denote k as the

number of data records that fell into the given leaf. Note that we drop double subscripting on the variances $\{\sigma_1^2, \dots, \sigma_k^2\}$ which are a subset of $\{\sigma_1^2, \dots, \sigma_n^2\}$ for the data records in this leaf.

$$\begin{aligned}
\mathbb{P}(\mu \mid \mathbf{R}, \sigma_1^2, \dots, \sigma_k^2, \gamma) &\propto \mathbb{P}(R \mid \mu, \sigma_1^2, \dots, \sigma_k^2, \gamma) \mathbb{P}(\mu \mid \sigma_1^2, \dots, \sigma_k^2) \\
&= \mathcal{N}_k(\mu \mathbf{1}, \sigma^2 \mathbf{D}) \mathcal{N}(0, \sigma_\mu^2) = \left(\prod_{i=1}^k \mathcal{N}(\mu, \sigma_i^2) \right) \mathcal{N}(0, \sigma_\mu^2) \\
&= \mathcal{N} \left(\frac{\sum_{i=1}^k \frac{R_i}{\sigma_i^2}}{\frac{1}{\sigma_\mu^2} + \sum_{i=1}^k \frac{1}{\sigma_i^2}}, \frac{1}{\frac{1}{\sigma_\mu^2} + \sum_{i=1}^k \frac{1}{\sigma_i^2}} \right)
\end{aligned} \tag{A.34}$$

Drawing the \mathfrak{T} 's (steps 1, 3, \dots , $2m - 1$)

As described in Kapelner and Bleich (2015, Appendix A), the draw of a new tree structure relies on a Metropolis-Hastings step where trees can be altered via growing new child nodes from an existing terminal node (**GROW**), pruning two terminal nodes such that their parent becomes terminal (**PRUNE**), or changing the splitting rule in a node (**CHANGE**).

Below is the Metropolis-Hastings ratio where the parameter sampled is the tree and the data is the responses unexplained by other trees denoted by \mathbf{R} . We denote the new, proposal tree with an asterisk and the original tree without the asterisk.

$$r = \frac{\mathbb{P}(\mathfrak{T}_* \rightarrow \mathfrak{T}) \mathbb{P}(\mathfrak{T}_* \mid \mathbf{R}, \sigma^2, \gamma)}{\mathbb{P}(\mathfrak{T} \rightarrow \mathfrak{T}_*) \mathbb{P}(\mathfrak{T} \mid \mathbf{R}, \sigma^2, \gamma)} \tag{A.35}$$

This can be reformulated using repeated applications of Bayes' Rule to be a prod-

uct of three ratios.

$$r = \underbrace{\frac{\mathbb{P}\left(\mathfrak{T}_* \rightarrow \mathfrak{T}\right)}{\mathbb{P}\left(\mathfrak{T} \rightarrow \mathfrak{T}_*\right)}}_{\text{transition ratio}} \times \underbrace{\frac{\mathbb{P}\left(\mathbf{R} \mid \mathfrak{T}_*, \sigma^2, \gamma\right)}{\mathbb{P}\left(\mathbf{R} \mid \mathfrak{T}, \sigma^2, \gamma\right)}}_{\text{likelihood ratio}} \times \underbrace{\frac{\mathbb{P}\left(\mathfrak{T}_*\right)}{\mathbb{P}\left(\mathfrak{T}\right)}}_{\text{tree structure ratio}} \quad (\text{A.36})$$

Note that the probability of the tree structure is independent of σ^2 and γ .

The transition ratio and the tree structure ratio remain the same as in the original BART as they do not depend on the variance parameters. The likelihood ratio now must take into account the heterogeneity in variances. The PRUNE likelihood ratio is the inverse of the GROW likelihood ratio. Thus, we only need to compute likelihood ratios for GROW and CHANGE.

The Likelihood Ratio for the GROW proposal

Since the likelihoods are solely determined by the terminal nodes, the grown proposal tree differs from the original tree by only the selected node to be grown. We denote the node to be grown by ℓ , the left child by ℓ_L and the right child by ℓ_R .

$$\frac{\mathbb{P}\left(\mathbf{R} \mid \mathfrak{T}_*, \sigma^2, \gamma\right)}{\mathbb{P}\left(\mathbf{R} \mid \mathfrak{T}, \sigma^2, \gamma\right)} = \frac{\int_{\mathbb{R}} \mathbb{P}\left(\mathbf{R}_{\ell_L} \mid \mu, \sigma^2, \gamma\right) \mathbb{P}(\mu) \, \mathrm{d}\mu \int_{\mathbb{R}} \mathbb{P}\left(\mathbf{R}_{\ell_R} \mid \mu, \sigma^2, \gamma\right) \mathbb{P}(\mu) \, \mathrm{d}\mu}{\int_{\mathbb{R}} \mathbb{P}\left(\mathbf{R}_{\ell} \mid \mu, \sigma^2, \gamma\right) \mathbb{P}(\mu) \, \mathrm{d}\mu} \quad (\text{A.37})$$

Each of these three integrals are the same with regards to marginalizing the μ term:

$$\begin{aligned} & \int_{\mathbb{R}} \mathbb{P}\left(\mathbf{R} \mid \mu, \sigma^2, \gamma\right) \mathbb{P}(\mu) \, \mathrm{d}\mu \\ &= \int_{\mathbb{R}} \frac{1}{(2\pi)^{\frac{n}{2}} |\mathbf{D}|^{-\frac{1}{2}}} \exp\left(-\frac{1}{2} (\mathbf{R} - \mu \mathbf{1})^\top \mathbf{D}^{-1} (\mathbf{R} - \mu \mathbf{1})\right) \frac{1}{\sqrt{2\pi\sigma_\mu^2}} \exp\left(-\frac{1}{2\sigma_\mu^2} \mu^2\right) \, \mathrm{d}\mu \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{(2\pi)^{\frac{n}{2}+1} \sqrt{\sigma_\mu^2 \prod_{i=1}^n \sigma_i^2}} \exp \left(-\frac{1}{2} \sum_{i=1}^n \frac{R_i^2}{\sigma_i^2} \right) \\
&\quad \times \int_{\mathbb{R}} \exp \left(-\frac{1}{2} \left(\frac{\mu^2}{\sigma_\mu^2} - 2\mu \sum_{i=1}^n \frac{R_i}{\sigma_i^2} + \mu^2 \sum_{i=1}^n \frac{1}{\sigma_i^2} \right) \right) d\mu \\
&= \left((2\pi)^n \left(1 + \sigma_\mu^2 \sum_{i=1}^n \frac{1}{\sigma_i^2} \right) \prod_{i=1}^n \sigma_i^2 \right)^{-\frac{1}{2}} \exp \left(\frac{1}{2} \left(\frac{\sigma_\mu^2 \left(\sum_{i=1}^n \frac{R_i}{\sigma_i^2} \right)^2}{1 + \sigma_\mu^2 \sum_{i=1}^n \frac{1}{\sigma_i^2}} - \sum_{i=1}^n \frac{R_i^2}{\sigma_i^2} \right) \right) \quad (\text{A.38})
\end{aligned}$$

The likelihood ratio can now be computed by substituting Equation A.38 into Equation A.37 three times to arrive at:

$$\begin{aligned}
\frac{\mathbb{P}(\mathbf{R} \mid \mathfrak{F}_* \sigma^2, \gamma)}{\mathbb{P}(\mathbf{R} \mid \mathfrak{F}, \sigma^2, \gamma)} &= \sqrt{\frac{1 + \sigma_\mu^2 \sum_{i=1}^{n_\ell} \frac{1}{\sigma_i^2}}{\left(1 + \sigma_\mu^2 \sum_{i=1}^{n_{\ell_L}} \frac{1}{\sigma_i^2} \right) \left(1 + \sigma_\mu^2 \sum_{i=1}^{n_{\ell_R}} \frac{1}{\sigma_i^2} \right)}} \quad (\text{A.39}) \\
&\quad \times \exp \left(\frac{\sigma_\mu^2}{2} \left(\frac{\left(\sum_{i=1}^{n_{\ell_L}} \frac{R_i}{\sigma_i^2} \right)^2}{1 + \sigma_\mu^2 \sum_{i=1}^{n_{\ell_L}} \frac{1}{\sigma_i^2}} + \frac{\left(\sum_{i=1}^{n_{\ell_R}} \frac{R_i}{\sigma_i^2} \right)^2}{1 + \sigma_\mu^2 \sum_{i=1}^{n_{\ell_R}} \frac{1}{\sigma_i^2}} - \frac{\left(\sum_{i=1}^{n_\ell} \frac{R_i}{\sigma_i^2} \right)^2}{1 + \sigma_\mu^2 \sum_{i=1}^{n_\ell} \frac{1}{\sigma_i^2}} \right) \right).
\end{aligned}$$

The Likelihood Ratio for the CHANGE proposal

The homoskedastic BART implementation of Kapelner and Bleich (2015) considered change proposals for singly internal nodes only (i.e., both child nodes must be terminal nodes). The likelihood ratio in this case simplifies to the likelihood of these two leaves before and after the change:

$$\frac{\mathbb{P}(R_{\ell^*,1}, \dots, R_{\ell^*,n_{\ell^*}} \mid \sigma^2, \gamma) \mathbb{P}(R_{r^*,1}, \dots, R_{r^*,n_{r^*}} \mid \sigma^2, \gamma)}{\mathbb{P}(R_{\ell,1}, \dots, R_{\ell,n_{\ell}} \mid \sigma^2, \gamma) \mathbb{P}(R_{r,1}, \dots, R_{r,n_r} \mid \sigma^2, \gamma)}. \quad (\text{A.40})$$

The ℓ refers to the left terminal node and r refers to the terminal right node. The ℓ^* and the r^* denote these same two nodes in the proposal tree, i.e after the parent's split rule was changed.

The likelihood with μ margined out has been calculated in Equation A.38 and we express it here with a convenient factorization:

$$\underbrace{\left((2\pi)^{n_{\ell}} \left(1 + \sigma_{\mu}^2 \sum_{i=1}^{n_{\ell}} \frac{1}{\sigma_i^2} \right) \prod_{i=1}^{n_{\ell}} \sigma_i^2 \right)^{-\frac{1}{2}}}_{A} \quad (\text{A.41})$$

$$\times \underbrace{\exp \left(-\frac{1}{2} \sum_{i=1}^{n_{\ell}} \frac{R_i^2}{\sigma_i^2} \right)}_B \underbrace{\exp \left(\frac{\sigma_{\mu}^2}{2} \left(\frac{\left(\sum_{i=1}^{n_{\ell}} \frac{R_i}{\sigma_i^2} \right)^2}{1 + \sigma_{\mu}^2 \sum_{i=1}^{n_{\ell}} \frac{1}{\sigma_i^2}} \right) \right)}_C. \quad (\text{A.42})$$

To find the ratio, we must substitute this expression into equation A.40 four times.

We begin by substituting only the term marked A above to arrive at

$$\frac{(2\pi)^{\frac{n_{\ell}}{2}} \sqrt{\left(\prod_{i=1}^{n_{\ell}} \sigma_i^2 \right) \left(1 + \sigma_{\mu}^2 \sum_{i=1}^{n_{\ell}} \frac{1}{\sigma_i^2} \right)} (2\pi)^{\frac{n_r}{2}} \sqrt{\left(\prod_{i=1}^{n_r} \sigma_i^2 \right) \left(1 + \sigma_{\mu}^2 \sum_{i=1}^{n_r} \frac{1}{\sigma_i^2} \right)}}{(2\pi)^{\frac{n_{\ell^*}}{2}} \sqrt{\left(\prod_{i=1}^{n_{\ell^*}} \sigma_i^2 \right) \left(1 + \sigma_{\mu}^2 \sum_{i=1}^{n_{\ell^*}} \frac{1}{\sigma_i^2} \right)} (2\pi)^{\frac{n_{r^*}}{2}} \sqrt{\left(\prod_{i=1}^{n_{r^*}} \sigma_i^2 \right) \left(1 + \sigma_{\mu}^2 \sum_{i=1}^{n_{r^*}} \frac{1}{\sigma_i^2} \right)}}. \quad (\text{A.43})$$

Of course $n_{\ell^*} + n_{r^*} = n_{\ell} + n_r = n$ and the σ^2 's are the same since they aren't drawn

until later on in the Gibbs sampling scheme. Thus, the above reduces to

$$\sqrt{\frac{\left(1 + \sigma_\mu^2 \sum_{i=1}^{n_\ell} \frac{1}{\sigma_i^2}\right) \left(1 + \sigma_\mu^2 \sum_{i=1}^{n_r} \frac{1}{\sigma_i^2}\right)}{\left(1 + \sigma_\mu^2 \sum_{i=1}^{n_{\ell^*}} \frac{1}{\sigma_i^2}\right) \left(1 + \sigma_\mu^2 \sum_{i=1}^{n_{r^*}} \frac{1}{\sigma_i^2}\right)}}. \quad (\text{A.44})$$

In term B , upon making all four substitutions, it is clear all the parents' observations must be summed in the numerator as well as the denominator. Thus, this term cancels.

Now we examine term C . Due to the exponentiation, multiplication becomes addition and division becomes subtraction and all four substitutions yield

$$\exp \left(\frac{\sigma_\mu^2}{2} \left(\frac{\left(\sum_{i=1}^{n_{\ell^*}} \frac{R_i}{\sigma_i^2} \right)^2}{1 + \sigma_\mu^2 \sum_{i=1}^{n_{\ell^*}} \frac{1}{\sigma_i^2}} + \frac{\left(\sum_{i=1}^{n_{r^*}} \frac{R_i}{\sigma_i^2} \right)^2}{1 + \sigma_\mu^2 \sum_{i=1}^{n_{r^*}} \frac{1}{\sigma_i^2}} - \frac{\left(\sum_{i=1}^{n_\ell} \frac{R_i}{\sigma_i^2} \right)^2}{1 + \sigma_\mu^2 \sum_{i=1}^{n_\ell} \frac{1}{\sigma_i^2}} - \frac{\left(\sum_{i=1}^{n_r} \frac{R_i}{\sigma_i^2} \right)^2}{1 + \sigma_\mu^2 \sum_{i=1}^{n_r} \frac{1}{\sigma_i^2}} \right) \right). \quad (\text{A.45})$$

Multiplying terms A.44 and A.45 yields the likelihood ratio for the **CHANGE** proposal.

A.4 Supplement for Chapter 8

The steps that follow summarize the procedure used to discard unreliable forecasts.

- (a) Construct a classification tree using a loss-matrix with the desired costs.
- (b) For each case, find the terminal node where it was classified.
- (c) Create a table that shows for each terminal node, how many observations in that node were labelled “Fail” vs. “No Fail.”
- (d) To understand how close the vote in each node was, re-weight the “Fails” in each terminal node by their weight as if they were false negatives (e.g., assign a cost of 5.0 to each failure). This now allows one to check the majority vote (using weighted fails) to see which classification won.
- (e) Let F be the weighted sum of all of the “Fails” (e.g., 400 fails \times 5.0 cost = 2000) and NF be total number of non-failures, each with a weight of 1.0. Let $p = F/(F + NF)$, the proportion of the weighted total number of votes that are “Fail” in a given terminal node. This can be done symmetrically with $(1 - p)$ as well.
- (f) Set a desired margin. Check if $|p - .5| < c$ is too close to call. How close is too close will require some trial and error.
- (g) Store which nodes are too close and which are not. Use the output from the tree with the additional label that some nodes are now too close too call.
- (h) One can exclude these too-close observations and run a cross-check against the bootstrapped trees to see how the stability improves.

Bibliography

- Ahmad, F. S., Metlay, J. P., Barg, F. K., Henderson, R. R., and Werner, R. M. (2013). Identifying hospital organizational strategies to reduce readmissions. *American Journal of Medical Quality*, 28(4):278–285.
- Albert, J. H. and Chib, S. (1993). Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, 88(422):669–679.
- Allaudeen, N., Schnipper, J. L., Orav, E. J., Wachter, R. M., and Vidyarthi, A. R. (2011a). Inability of providers to predict unplanned readmissions. *Journal of general internal medicine*, 26(7):771–776.
- Allaudeen, N., Vidyarthi, A., Maselli, J., and Auerbach, A. (2011b). Redefining readmission risk factors for general medicine patients. *Journal of Hospital Medicine*, 6(2):54–60.
- Amarasingham, R., Patel, P. C., Toto, K., Nelson, L. L., Swanson, T. S., Moore, B. J., Xie, B., Zhang, S., Alvarez, K. S., Ma, Y., et al. (2013). Allocating scarce resources in real-time to reduce heart failure readmissions: a prospective, controlled study. *BMJ quality & safety*, 22(12):998–1005.
- Andrews, D. A., Bonta, J., and Wormith, J. S. (2006). The recent past and near future of risk and/or need assessment. *Crime and delinquency*, 52(1):7.
- Au, A. G., McAlister, F. A., Bakal, J. A., Ezekowitz, J., Kaul, P., and van Walraven, C. (2012). Predicting the risk of unplanned readmission or death within 30 days of discharge after a heart failure hospitalization. *American heart journal*, 164(3):365–372.
- Bache, K. and Lichman, M. (2013). UCI machine learning repository.
- Baillie, C. A., VanZandbergen, C., Tait, G., Hanish, A., Leas, B., French, B., C, W. H., Behta, M., and Umscheid, C. A. (2013). The readmission risk flag: Using the electronic health record to automatically identify patients at risk for 30-day readmission. *Journal of Hospital Medicine*, 8(12):689–695.
- Barnes, G. C., Ahlman, L., Gill, C., Sherman, L. W., Kurtz, E., and Malvestuto, R. (2010). Low-intensity community supervision for low-risk offenders: a randomized, controlled trial. *Journal of Experimental Criminology*, 6(2):159–189.

- Berk, R. A. (2007). Statistical inference and meta-analysis. *Journal of Experimental Criminology*, 3(3):247–270.
- Berk, R. A. (2008). Forecasting methods in crime and justice. *Annual review of law and social science*, 4:219–238.
- Berk, R. A. (2009). The role of race in forecasts of violent crime. *Race and social problems*, 1(4):231–242.
- Berk, R. A. (2011). Asymmetric loss functions for forecasting in criminal justice settings. *Journal of Quantitative Criminology*, 27(1):107–123.
- Berk, R. A. (2012). *Criminal Justice Forecasts of Risk: a Machine Learning Approach*. Springer Science & Business Media.
- Berk, R. A. (2013). Algorithmic criminology. *Security Informatics*, 2(1):1–14.
- Berk, R. A., Brown, L., and Zhao, L. (2010). Statistical inference after model selection. *Journal of Quantitative Criminology*, 26(2):217–236.
- Berk, R. A., He, Y., and Sorenson, S. B. (2005). Developing a practical forecasting screener for domestic violence incidents. *Evaluation Review*, 29(4):358–383.
- Berk, R. A., Kriegler, B., and Baek, J. H. (2006). Forecasting dangerous inmate misconduct: An application of ensemble statistical procedures. *Journal of Quantitative Criminology*, 22(2):131–145.
- Berk, R. A., Sherman, L. W., Barnes, G., Kurtz, E., and Ahlman, L. (2009). Forecasting murder within a population of probationers and parolees: a high stakes application of statistical learning. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 172(1):191–211.
- Billings, J., Blunt, I., Steventon, A., Georghiou, T., Lewis, G., and Bardsley, M. (2012). Development of a predictive model to identify inpatients at risk of re-admission within 30 days of discharge (parr-30). *BMJ open*, 2(4).
- Bisharat, N., Handler, C., and Schwartz, N. (2012). Readmissions to medical wards: Analysis of demographic and socio-medical factors. *European journal of internal medicine*, 23(5):457–460.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*, volume 4. springer New York.
- Bleich, J., Kapelner, A., George, E. I., and Jensen, S. T. (2014). Variable selection for BART: An application to gene regulation. *Annals of Applied Statistics*, 8(3):1750–1781.
- Borden, H. G. (1928). Factors for predicting parole success. *Journal of the American Institute of Criminal Law and Criminology*, pages 328–336.
- Boscardin, W. J. and Gelman, A. (1994). Bayesian computation for parametric models of heteroscedasticity in the linear model. *University of California, Berkeley*.
- Bottolo, L. and Richardson, S. (2010). Evolutionary stochastic search for bayesian model exploration. *Bayesian Analysis*, 5:583–618.
- Bowles, K. H., Hanlon, A., Holland, D., Potashnik, S. L., and Topaz, M. (2014). Impact of discharge planning decision support on time to readmission among older adult medical patients. *Professional case management*, 19(1):29–38.

- Box, G. E. P. and Jenkins, G. M. (1970). Time Series Analysis: Forecasting and Control. *Holden-Day, San Francisco*.
- Bradley, E. H., Curry, L., Horwitz, L. I., Sipsma, H., Thompson, J. W., Elma, M., Walsh, M. N., and Krumholz, H. M. (2012). Contemporary evidence about hospital strategies for reducing 30-day readmissions: a national study. *Journal of the American College of Cardiology*, 60(7):607–614.
- Bradley, E. H., Curry, L., Horwitz, L. I., Sipsma, H., Wang, Y., Walsh, M. N., Goldmann, D., White, N., Piña, I. L., and Krumholz, H. M. (2013a). Hospital strategies associated with 30-day readmission rates for patients with heart failure. *Circulation: Cardiovascular Quality and Outcomes*, 6(4):444–450.
- Bradley, E. H., Yakusheva, O., Horwitz, L. I., Sipsma, H., and Fletcher, J. (2013b). Identifying patients at increased risk for unplanned readmission. *Medical care*, 51(9):761–766.
- Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.
- Breiman, L. (2001a). Random forests. *Machine learning*, 45(1):5–32.
- Breiman, L. (2001b). Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–231.
- Breiman, L. and Cutler, A. (2013). Online manual for random forests. *Website*, www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm.
- Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and Regression Trees*. CRC press.
- Burgess, E. W. (1928). Factors determining success or failure on parole. *The workings of the indeterminate sentence law and the parole system in Illinois*, pages 221–234.
- Bushway, S. D. (2010). Estimating empirical blackstone ratios in two settings: Murder cases and hiring. *Alb. L. Rev.*, 74:1087.
- Cameron, A. C. and Trivedi, P. K. (2005). *Microeconometrics: Methods and Applications*. Cambridge university press.
- Carroll, R. J. and Ruppert, D. (1988). *Transformation and Weighting in Regression*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis.
- Casey, P. M., Warren, R. K., and Elek, J. K. (2011). *Using offender risk and needs assessment information at sentencing: Guidance for courts from a national working group*. National Center for State Courts.
- Cepeda, E. and Gamerman, D. (2001). Bayesian modeling of variance heterogeneity in normal regression models. *Brazilian Journal of Probability and Statistics*, pages 207–221.
- Chan, D., Kohn, R., Nott, D. J., and Kirby, C. (2006). Locally adaptive semiparametric estimation of the mean and variance functions in regression models. *Journal of Computational and Graphical Statistics*, pages 0–26.
- Chandra, S., Agarwal, D., Hanson, A., Farmer, J., Pickering, B. W., Gajic, O., and Herasevich, V. (2011). The use of an electronic medical record based automatic calculation tool to quantify risk of unplanned readmission to the intensive care unit: A validation study. *Journal of critical care*, 26(6):634–e9.

- Chen, C., Liaw, A., and Breiman, L. (2004). Using random forest to learn imbalanced data. Technical report, Department of Statistics, UC Berkeley.
- Chen, G., Jensen, S. T., and Stoeckert, C. J. (2007). Clustering of genes into regulons using integrated modeling - COGRIM. *Genome Biology*, 8:R4.
- Chipman, H., George, E., and McCulloch, R. (1998). Bayesian CART Model Search. *Journal of the American Statistical Association*, 93(443):935.
- Chipman, H. A., George, E. I., and McCulloch, R. E. (2010). BART: Bayesian Additive Regressive Trees. *The Annals of Applied Statistics*, 4(1):266–298.
- Cutler, D. R., Edwards Jr, T. C., Beard, K. H., Cutler, A., Hess, K. T., Gibson, J., and Lawler, J. J. (2007). Random forests for classification in ecology. *Ecology*, 88(11):2783–2792.
- Damien, P., Dellaportas, P., Polson, N. G., and Stephens, D. A. (2013). *Bayesian Theory and Applications*, pages 455–464. Oxford University Press, first edition.
- de Lissovoy, G. (2013). Big data meets the electronic medical record: A commentary on” identifying patients at increased risk for unplanned readmission”. *Medical care*, 51(9):759–760.
- Deng, H. and Runger, G. (2012). Feature selection via regularized trees. *The 2012 International Joint Conference on Neural Networks (IJCNN)*.
- Díaz-Uriarte, R. and Alvarez de Andrés, S. (2006). Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, 7:3.
- Ding, Y. and Simonoff, J. S. (2010). An investigation of missing data methods for classification trees applied to binary response data. *The Journal of Machine Learning Research*, 11:131–170.
- Donzé, J., Aujesky, D., Williams, D., and Schnipper, J. L. (2013). Potentially avoidable 30-day hospital readmissions in medical patients: derivation and validation of a prediction model. *JAMA internal medicine*, 173(8):632–638.
- Dumbill, E. (2013). Making sense of big data. *Big Data*, 1(1):1–2.
- Farrington, D. P. and Tarling, R. (1985). *Prediction in Criminology*. SUNY Press.
- Feeley, M. and Simon, J. (1994). Actuarial justice: the emerging new criminal law. *The futures of criminology*. London: Sage, pages 173–201.
- Freedman, D. A. (2009). *Statistical Models: Theory and Practice*. cambridge university press.
- Freund, Y. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.
- Friedman, J. H. (1991). Multivariate adaptive regression splines (with discussion and a rejoinder by the author). *The annals of statistics*, 19:1–67.
- Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232.
- Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378.

- Friedman, J. H., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1).
- Gamerman, D. (1997). Sampling from the posterior distribution in generalized linear mixed models. *Statistics and Computing*.
- Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2004). *Bayesian Data Analysis*. Chapman & Hall / CRC, second edition.
- Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 6:721–741.
- George, E. and McCulloch, R. (1993). Variable selection via Gibbs sampling. *Journal of the American Statistical Association*, 88:881–889.
- Gottfredson, S. D. and Moriarty, L. J. (2006). Statistical risk assessment: Old problems and new applications. *Crime & Delinquency*, 52(1):178–200.
- Gramacy, R. B. (2007). tgp: An R package for Bayesian nonstationary, semiparametric nonlinear regression and design by treed Gaussian process models. *Journal Of Statistical Software*, 19(9).
- Gramacy, R. B., Taddy, M., and Wild, S. (2013). Variable selection and sensitivity analysis using dynamic trees, with an application to computer code performance tuning. *The Annals of Applied Statistics*, 7(1):51–80.
- Hans, C. (2009). Bayesian lasso regression. *Biometrika*, 96:835–845.
- Hans, C., Dobra, A., and West, M. (2007). Shotgun stochastic search for “large p” regression. *Journal of the American Statistical Association*, 102:507–516.
- Harcourt, B. E. (2008). *Against Prediction: Profiling, Policing, and Punishing in an Actuarial Age*. University of Chicago Press.
- Hastie, R. and Dawes, R. M. (2001). *Rational Choice in an Uncertain World: The Psychology of Judgment and Decision Making*. Sage.
- Hastie, T. and Tibshirani, R. (2000). Bayesian backfitting. *Statistical Science*, 15(3):196–213.
- Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer Science, second edition.
- Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.
- Hocking, R. (1976). The analysis and selection of variables in linear regression. *Biometrics*, 32:1–49.
- Holmes, S. (2003). Bootstrapping phylogenetic trees: theory and methods. *Statistical Science*, pages 241–255.
- Hyatt, J., Chanenson, S. L., and Bergstrom, M. H. (2011). Reform in motion: The promise and perils of incorporating risk assessments and cost-benefit analysis into pennsylvania sentencing.
- Ishwaran, H. and Rao, J. S. (2005). Spike and slab variable selection: Frequentist and Bayesian strategies. *The Annals of Statistics*, 33(2):730–773.

- Ishwaran, H. and Rao, J. S. (2010). Generalized ridge regression: geometry and computational solutions when p is larger than n . *Technical Report*.
- Ishwaran, H., Rao, J. S., and Kogalur, U. B. (2013). *spikeslab : Prediction and variable selection using spike and slab regression*. R package version 1.1.5.
- Jencks, S. F., Williams, M. V., and Coleman, E. A. (2009). Rehospitalizations among patients in the medicare fee-for-service program. *New England Journal of Medicine*, 360(14):1418–1428.
- Jensen, S. T., Chen, G., and Stoeckert, C. J. (2007). Bayesian variable selection and data integration for biological regulatory networks. *Annals of Applied Statistics*, 1:612–633.
- Joynt, K. E. and Jha, A. K. (2012). Thirty-day readmissionstruth and consequences. *New England Journal of Medicine*, 366(15):1366–1369.
- Kapelner, A. and Bleich, J. (2015). bartMachine: Machine learning with Bayesian Additive Regression Trees. *Journal of Statistical Software, forthcoming*.
- Kibler, D. F., Aha, D. W., and Albert, M. K. (1989). Instance-based prediction of real-valued attributes. *Computational Intelligence*, 5:51.
- Kindo, B., Wang, H., and Pe, E. (2013). MBACT - Multiclass Bayesian Additive Classification Trees. *arXiv*, pages 1–29.
- Kleiman, M., Ostrom, B. J., and Cheesman, F. L. (2007). Using risk assessment to inform sentencing decisions for nonviolent offenders in virginia. *Crime & Delinquency*, 53(1):106–132.
- Kuhnert, P. M. and Mengersen, K. (2003). Reliability measures for local nodes assessment in classification trees. *Journal of Computational and Graphical Statistics*, 12(2):398–416.
- Lavenberg, J. G., Leas, B., Umscheid, C. A., Williams, K., Goldmann, D. R., and Kripalani, S. (2014). Assessing preventability in the quest to reduce hospital readmissions. *Journal of Hospital Medicine*, 9(9):598–603.
- Lee, T., Rinaldi, N., Robert, F., Odom, D., Bar-Joseph, Z., Gerber, G., Hannett, N., Harbison, C., Thompson, C., Simon, I., Zeitlinger, J., Jennings, E., Murray, H., Gordon, D., Ren, B., Wyrick, J., Tagne, J., Volkert, T., Fraenkel, E., Gifford, D., and Young, R. (2002). Transcriptional regulatory networks in *Saccharomyces cerevisiae*. *Science*, 298:763–764.
- Leslie, D. S., Kohn, R., and Nott, D. J. (2007). A general approach to heteroscedastic linear regression. *Statistics and Computing*, pages 1–29.
- Li, F. and Zhang, N. (2010). Bayesian variable selection in structured high-dimensional covariate spaces with applications in genomics. *Journal of the American Statistical Association*, 105:1978–2002.
- Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R news*, 2:18–22.
- Little, R. J. A. (1993). Pattern-mixture models for multivariate incomplete data. *Journal of the American Statistical Association*, 88(421):125–134.
- Little, R. J. A. and Rubin, D. B. (2002). *Statistical Analysis with Missing Data*. Wiley, second edition.

- Liu, Y. Y., Yang, M., Ramsay, M., Li, X. S., and Coid, J. W. (2011). A comparison of logistic regression, classification and regression tree, and neural networks models in predicting violent re-offending. *Journal of Quantitative Criminology*, 27(4):547–573.
- Messinger, S. L. and Berk, R. A. (1987). Dangerous people: a review of the NAS report on career criminals. *Criminology*, 25(3):767–781.
- Miller, A. J. (2002). *Subset Selection in Regression*. New York: Chapman and Hall, 2nd edition edition.
- Mitchell, T. and Beauchamp, J. (1988). Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404):1023–1032.
- Monahan, J. (1981). *Predicting Violent Behavior: An Assessment of Clinical Techniques*. Sage Publications Beverly Hills, CA.
- National Research Council (2013). *Frontiers in Massive Data Analysis*. The National Academies Press, Washington, DC.
- Ohlin, L. E. and Duncan, O. D. (1949). The efficiency of prediction in criminology. *American Journal of Sociology*, pages 441–452.
- Ohlin, L. E. and Lawrence, R. A. (1952). A comparison of alternative methods of parole prediction. *American Sociological Review*, pages 268–274.
- Oregon Youth Authority (2011). *OYA Recidivism Risk Assessment, Violent Crime (ORRA-V): Modeling Risk to Recidivate with a Violent Crime*. Oregon Youth Authority, Research and Evaluation.
- Park, T. and Casella, G. (2008). The bayesian lasso. *Journal of the American Statistical Association*, 103(482):681–686.
- Pew Center of the States (2011). Risk/needs assessment 101: science reveals new tools to manage offenders. Technical report, The Pew Center of the States.
- Pratola, M. T., Chipman, H., Higdon, D., McCulloch, R., and Rust, W. (2013). Parallel Bayesian Additive Regression Trees. Technical report, University of Chicago.
- Reiss Jr, A. J. (1951). The accuracy, efficiency, and validity of a prediction instrument. *American Journal of Sociology*, pages 552–561.
- Ridgeway, G. (2006). Generalized boosted regression models. *Documentation on the R Package gbm, version 1. 5, 7*.
- Ridgeway, G. (2013). The pitfalls of prediction. *NIJ Journal*, 271:34–40.
- Rockova, V. and George, E. (2013). EMVS: The EM approach to variable selection. *Journal of the American Statistical Association*, page (in press).
- Rokach, L. (2009). Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39.
- Rubin, D. B. (1978). Multiple imputations in sample surveys-a phenomenological Bayesian approach to nonresponse. Technical report, Proceedings of the Survey Research Methods Section, American Statistical Association.

- Rubin, D. B. (1981). The Bayesian bootstrap. *The annals of statistics*, 9(1):130–134.
- Ruppert, D., Wand, P., and Carroll, R. J. (2003). *Semiparametric Regression*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.
- Schafer, J. L. (1997). *Analysis of Incomplete Multivariate Data*. Chapman & Hall / CRC.
- Schmidt, G., Mattern, R., and Schüler, F. (1981). Biomechanical investigation to determine physical and traumatological differentiation criteria for the maximum load capacity of head and vertebral column with and without protective helmet under effect. Technical report, EFC Research Program on Biomechanics of Impacts. Final report Phase III, Project G5. Institute für Rechtsmedizin, Universität Heidelberg.
- Skeem, J. L. and Monahan, J. (2011). Current directions in violence risk assessment. *Current Directions in Psychological Science*, 20(1):38–42.
- Stekhoven, D. J. and Bühlmann, P. (2012). MissForest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28:112–8.
- Stingo, F. and Vannucci, M. (2011). Variable selection for discriminant analysis with markov random field priors for the analysis of microarray data. *Bioinformatics*, 27:495–501.
- Strobl, C., Boulesteix, A., Zeileis, A., and Hothorn, T. (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC bioinformatics*, 8(1):25.
- Taddy, M. A., Gramacy, R. B., and Polson, N. G. (2011). Dynamic trees for learning and design. *Journal of the American Statistical Association*, 106(493):109–123.
- Therneau, T. M. and Atkinson, E. J. (1997). An introduction to recursive partitioning using the rpart routines. Technical report, Mayo Foundation.
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.
- Tollenaar, N. and Van der Heijden, P. G. M. (2013). Which method predicts recidivism best?: a comparison of statistical, machine learning and data mining predictive models. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 176(2):565–584.
- Tóth, N. (2008). *Handling classification uncertainty with decision trees in biomedical diagnostic systems*. PhD thesis, Department of Measurement and Information Systems, Budapest University of Technology and Economics.
- Troyanskaya, O., Cantor, M., and Sherlock, G. (2001). Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525.
- Turner, S., Hess, J., and Jannetta, J. (2009). Development of california risk assessment instrument. Technical report, Center for Evidence Based Corrections, University of California, Irvine.
- Twala, B. (2009). An empirical comparison of techniques for handling incomplete data using decision trees. *Applied Artificial Intelligence*, 23(5):1–35.
- Twala, B., Jones, M., and Hand, D. J. (2008). Good methods for coping with missing data in decision trees. *Pattern Recognition Letters*, 29(7):950–956.

- Umscheid, C. A., Betesh, J., VanZandbergen, C., Hanish, A., Tait, G., Mikkelsen, M. E., French, B., and Fuchs, B. D. (2014). Development, implementation, and impact of an automated early warning and response system for sepsis. *Journal of Hospital Medicine*.
- Urbanek, S. (2011). *rJava: Low-level R to Java interface*. R package version 0.9-3 available on CRAN.
- van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworth, 2nd edition edition.
- Vapnik, V. (1998). *Statistical Learning Theory*, volume 1. Wiley New York.
- White, H. (1980). A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica*, pages 817–838.
- Yang, M., Liu, Y. Y., and Coid, J. W. (2010). Applying neural networks and other statistics models to classification of serious offenders and the prediction of recidivism. *Vol 6/10. London: Ministry of Justice*.
- Yau, P. and Kohn, R. (2003). Estimation and variable selection in nonparametric heteroscedastic regression. *Statistics and Computing*, pages 191–208.
- Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.