

Inference for Random Forests

A Thesis

Presented to

The Division of Mathematical and Natural Sciences

Reed College

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Arts

Paul-Hieu Vinh Nguyen

May 2022

Approved for the Division
(Mathematics)

Jonathan Wells

Acknowledgements

I would like to thank some people for their support that made this thesis possible:

Nate and Kelly, for your guidance, support, and more

Friends at Home, Xander, Elton, Chris, Bradley, Habin, Leo, Amy, Ava, and Aristotle for your companionship

Reedie Friends and Mentors, Riley, Andy, Alex, Sarah, David, Olek, Sam, Ben, Zeki, and Simon, for the fun, help, and advice at Reed

Kim Bui and Justin Iverson, Co Mai Ly and Bac Tai, for the Portland welcome, and for the holiday dinners away from home

The Tse Family, for opening up your home and filling it with fond memories for the past six years and more

My Grandparents, Aunts, Uncles, and Cousins, for forming my support system, especially as a child

Noelle, for your love and support

Table of Contents

Chapter 1: Introduction	1
1.1 Random Forests	2
1.2 Recent Work	5
1.2.1 Subbagged Random Forests	5
1.2.2 Causal Forests	5
Chapter 2: Theory	9
2.1 Ensemble Methods and U-Statistics	9
2.1.1 U-Statistics	9
2.1.2 Subbagging	10
2.1.3 Random Forests	13
2.2 Estimating Variance	15
2.3 Inference Procedures	16
2.3.1 Setting the scene: Simple Linear Regression	16
2.3.2 Confidence Intervals and Hypothesis Tests for Ensemble Methods	18
2.3.3 Tests of Significance	19
Chapter 3: Algorithms and Code	23
3.1 Subbagging Algorithm	23
3.2 Subsampled Random Forest Algorithm	25
3.3 Variance Estimation Algorithms	26
3.4 Building Confidence Intervals	29
3.5 Performing Significance Tests	35
3.5.1 Variance Estimation Algorithms for Multiple Test Points	38
Chapter 4: Significance Test Simulation	43
4.1 Significance Test: SIMPLE	43
4.2 Significance Test: POLY	47

Chapter 5: Conclusion	51
5.1 Findings and Results	51
5.2 Limitations	51
5.3 Future Extensions	52
Appendix A: Code	53
References	55

List of Tables

3.1	Coverage Probabilities for Simple/MARS	33
4.1	Summary Statistics for SIMPLE simulation	46

List of Figures

1.1	Regression Tree: Palmers Penguins	2
3.1	Confidence Intervals for Simple Scenario	34
3.2	Confidence Intervals for MARS Scenario	34
4.1	Histogram of Test Statistics Testing the Significance of X_1	44
4.2	Histogram of Test Statistics Testing the Significance of X_2	45
4.3	Simple Regression Tree for Flipper Length	49
4.4	Partition of Predictor Space According to Flipper Length Regression Tree	49

Abstract

This thesis follows Mentch & Hooker (2016) to investigate inference procedures for random forests. We present subbagged ensemble methods, combinations of basic algorithms built on subsamples of the training set, as U-statistics. U-statistics were shown to have minimum variance by Halmos (1946) and to be asymptotically normal from Hoeffding (1992). We study and recreate inference procedures from Mentch & Hooker (2016) for the subbagged ensemble methods, namely confidence intervals for predictions and significance tests for predictors. We transcribe the pseudo-code for the inference procedures and provide functions to perform them in R. Finally, we present the results from simulation studies performed on generated data and discuss our findings.

Dedication

Dedicated to Mom, Dad, and Nicolas.

Chapter 1

Introduction

In a controversial 2001 paper, Leo Breiman describes two cultures in the statistical modeling world, one based on traditional data models, and he advocates for a shift to the second, which emphasizes prediction (Breiman, 2001b). The first culture is a data modeling culture, which assumes a stochastic data model for nature. The values of parameters are estimated from the data, and then are used for inference and prediction. The second culture is the algorithmic modeling culture. This approach focuses on finding an algorithm that prioritizes predictive accuracy. Breiman declares that statisticians' commitment to the data modeling culture has kept them away from new, interesting methods and problems. However, although Breiman advocates for a shift to the algorithmic approach, he notes that data models can be useful: they offer qualitative understanding of the data and can explain relationships within the data.

Later, Breiman also developed the random forest algorithmic model, an ensemble of tree classifiers, where each tree uses a random selection of predictors at each node to split upon. The random forest gives great accuracy and is robust to outliers and noise (Breiman, 2001a). This thesis will thread the line between Breiman's two cultures: we will study the theoretical properties of random forests in order to further develop inference techniques and increase its interpretability, giving further insight to understand data.

The thesis is organized as follows:

- the remainder of Chapter 1 discusses the random forest algorithm in greater depth, and then explores recently developed inference techniques for random forests proposed by researchers. In particular, we discuss Mentch & Hooker (2016)'s confidence intervals and hypothesis tests on subbagged random forests, and Wager & Athey (2018)'s inference techniques on causal forests.

- Chapter 2 describes in greater detail the theory behind the inference procedures in Mentch & Hooker (2016). We discuss subbagging and random forests as U-statistics, report the limiting distributions of their predictions, and describe the procedures to procure confidence intervals and to perform significance tests.
- Chapter 3 provides functions, with comments, to perform the inference procedures in R, as well as showcases the results of the confidence intervals.
- Chapter 4 present the results of simulations using the significance test mentioned above on two different data generation models.
- Chapter 5 describes the key findings of this thesis and concludes the thesis with comments on potential extensions of this work.

1.1 Random Forests

To build an ensemble of tree predictors, one must build an individual tree first. Let's build a single regression tree to predict the flipper length of a penguin in the Palmer's Penguins dataset (Horst, Hill, & Gorman, 2020).

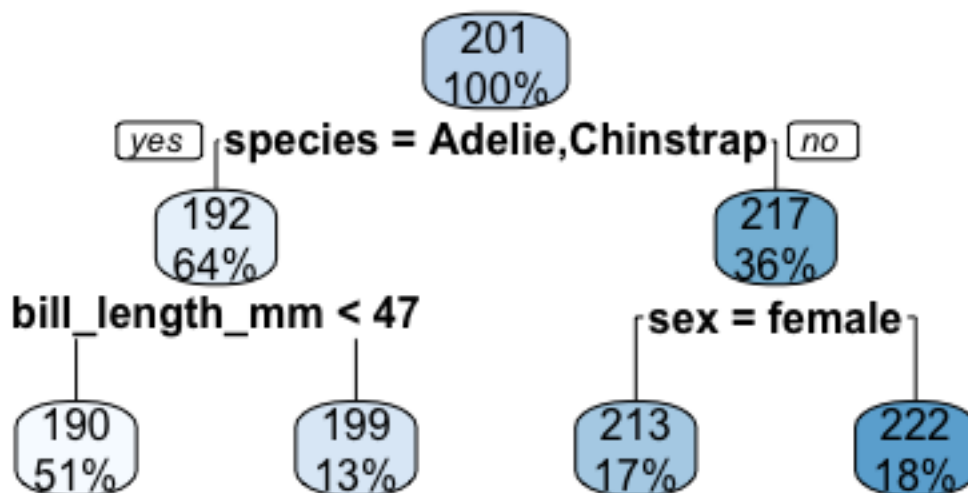


Figure 1.1: Regression Tree: Palmers Penguins

This tree consists of a series of splitting rules. We first split our dataset of penguins into two categories, one with the species Adelie and Chinstrap, and the other species. Among the Adelie and Chinstrap penguins, then consider each penguin's bill length,

and if it is less than 47 mm, the predicted flipper length is 190. The predicted flipper length of a penguin with a bill longer than 47 mm is 199mm. On the other hand, for species besides the Adelie and Chinstrap, consider if the penguin is female. If so, the tree predicts a flipper length of 213 mm, and for males, 222 mm.

To build a regression tree, we divide the predictor spaces into distinct regions, for example Adelie and Chinstrap penguins, and non-Adelie and Chinstrap penguins. We divide the predictor space into regions R_1, \dots, R_j in order to minimize the residual sum of squares, $\sum_{j=1}^J \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$. For every observation that falls into a region, make the same prediction, which is the mean of the response values for the training observations that fall within this particular region. A classification tree is similar to a regression tree, only we predict a qualitative response rather than a numerical one.

Breiman’s original random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all the trees in the forest. In essence, some randomness is used to grow these trees. Early renditions of random forests include bagging, the process of using bootstrapping to produce a training set for individual trees. Likewise, an ensemble made from trees grown by selecting splits at random (random split selection) is a random forest (Dietterich, 1998). Another random forest building mechanism is the “random subspace” method, which randomly selects of a subset of predictors to use to grow each tree (Ho, 1998). The random forest we discuss in this thesis, and commonly used, is one where the random vector determines which predictors are used at each node within a tree. Breiman (2001a) finds that this type of random forest yields similar predictive results as Adaboost, another tree-based algorithm that has great predictive accuracy, but is more robust than Adaboost.

Random forests are constructed by averaging the predictions of individual trees, where at each node in an individual tree, the algorithm uses only a random subset of available predictors to grow the tree. While this process may seem unintuitive, the approach mitigates the high variance and overfitting that follows from the individual classification and regression trees (CART). Additionally, the random forest is used in tandem with bagging. In bagging, each tree is built with a random bootstrap sample of the training data. We simulate a bootstrap sample by drawing n observations from our training data *with replacement*. Each bootstrap sample will contain roughly 63% of the training points; since we draw with replacement, many training observations will be included in the bootstrap set multiple times. This leaves 37% of the original training observations to be used as test points, our “out-of-bag” estimates. Breiman notes that the use of bagging enhances accuracy and gives ongoing estimates of the

errors of the overall ensemble of trees. Bagging also provides estimates for variable importance, providing some interpretability to this algorithmic method.

Bagging helps alleviate the high variance produced by standard decision trees, as standard decision trees often overfit to the training data. Random forests improve the bagged trees by decorrelating the trees. If a data set includes one very strong predictor, each bagged tree will use this strong predictor early in the tree-building process, and the bagged trees will look very similar and produce very similar predictions. In contrast, some trees within the random forest will not include this predictor by design, decorrelating the trees, and making the average of the resulting trees less variable.

Procedure for constructing a random forest from a training set. Given training set T , form bootstrap training sets T_k , construct classifiers (trees) $h(\mathbf{x}, T_k)$, and let these vote to form the random forest prediction. Note that each bootstrapped training set results in the growth of one tree, and that the random forest prediction is either the average prediction of all the trees, or a majority vote in the classification context. Now, we can make use of the randomness in which observations are involved in the bootstrap to generate useful out-of-bag error estimates. For each (y, \mathbf{x}) in the training set, aggregate the votes only over the classifiers for which T_k does not contain (y, \mathbf{x}) , call this the out-of-bag classifier. The out-of-bag estimate for the generalization error is the error rate of the out-of-bag classifier on the training set. Breiman (2001a) notes that the out-of-bag estimate is as accurate as using a test set the same size of the training set, eliminating the need to set aside a test set. Breiman’s method of determining “variable importance” compares the percent increase in misclassification rate for internal out-of-bag (observations not in a tree’s bootstrap sample) estimates with a randomized predictor to the rate with trees using all variables normally.

To get this misclassification rate, one can perform the following procedure: With M input variables, after constructing each tree, randomly permute the values of the m th variable in the out-of-bag examples, and use said tree to predict with the out-of-bag data. Then, save the classification given for each data point from the out-of-bag examples, and repeat for all m input variables. Finally, the plurality of the out-of-bag class votes for each observation with the m th variable randomized is compared to the true class label of this observation, giving our misclassification rate (Breiman, 2001a).

If a particular predictor has a large percent increase in misclassification rate, then the randomized predictor is considered more important. Although this is currently the widely used metric for variable importance, much work has been done since then to further the interpretability of random forests (Mentch & Hooker, 2016, 2017; Wager

& Athey, 2018; Wager, Hastie, & Efron, 2014).

1.2 Recent Work

We focus on two main branches from the recent inference on random forests literature. The first is Mentch & Hooker (2016)’s work on subbagged random forests. The second branch is Wager & Athey (2018)’s work on causal forests.

1.2.1 Subbagged Random Forests

Mentch & Hooker (2016) introduces tools for performing formal statistical inference for predictions from supervised learning ensembles, specifically for a variant of bagging/random forests. By considering only a subsample of the data during the bagging process, the authors demonstrate this fits into the statistical framework of U-statistics, which were shown to have minimum variance by Halmos (1946) and later demonstrated to be asymptotically normal by Hoeffding (1948). Mentch and Hooker show that under weak regularity conditions, predictions from the subsample ensemble methods are asymptotically normal. They then provide a method to consistently estimate the variance in the limiting distribution, making producing confidence intervals and testing feature significance possible.

To estimate the approximate distribution of the predictions at any given feature vector of interest \mathbf{x}^* and produce confidence intervals, we estimate variance parameters and take quantiles from the appropriate limiting normal distribution. From there, we can use the standard confidence interval procedure to test hypotheses for an expected ensemble prediction, θ_{k_n} , made with subsamples of size k_n , of the form $\theta_{k_n} = c$. However, note that these confidence intervals are only for the expected prediction θ_{k_n} , not for the underlying regression function. This procedure tells us additional information when used with the standard approach of looking at the Mean Squared Error (MSE) of the random forests. The MSE approach provides information regarding the accuracy of the model, and these confidence intervals supplies information regarding the stability and reasonable values of the predictions.

1.2.2 Causal Forests

In contrast, Wager & Athey (2018) develop a non-parametric causal forest for estimating heterogeneous treatment effects, and they show that causal forests are pointwise consistent for the true treatment effect and have an asymptotically Gaussian and

centered sampling distribution. While their work is focused on causality and treatment effects, Wager and Athey perform significant work in asymptotic theory for random forests. They develop asymptotic normality theory enabling statistical inference with random forest predictions. Wager and Athey provide a set of conditions under which predictions made by random forests are both asymptotically unbiased and Gaussian, thus allowing for classical statistical inference.

Given the following conditions for the base learner trees, Wager and Athey note that random forest predictions are asymptotically Normal:

The first condition is an “honest” tree. A tree is honest if for each training example i , the tree does not use the response to place its splits. For example, grow a tree with one subsample, and then use a different subsample to estimate the predictions at the leaves of the tree. Next, we need random-split trees, trees that incorporate randomness in the way trees choose the subset of variables to split on. Additionally, we need regularity, evenly split and deep trees, and symmetry, where the order of training observations does not impact tree structure.

This paper could be another great topic to base a future thesis on, especially for a student interested in causal inference, but we choose to focus on Mentch & Hooker (2016), and consider ensembles based on more general base learners that do not require “honesty”.

Both Mentch & Hooker (2016) and Wager & Athey (2018) focus on asymptotic normality theory in order to perform statistical inference on random forest predictions. Both papers utilize subsampling in their procedures, as estimators based on bagging can exhibit unexpected properties even in simple cases (Friedman & Hall, 2007). Mentch & Hooker (2016) and Wager & Athey (2018) both achieve asymptotic normality through their subbagged random forest procedures; however, their necessary conditions and generalizability differ. Wager & Athey (2018) focus their efforts on estimating heterogeneous treatment effects, and require an “honest” tree building mechanism, while Mentch & Hooker (2016) focus on a more general class of subbagged ensemble predictors. This generalizability does come with a cost however: random forests built under this method will not necessarily be asymptotically unbiased, so does not directly translate to statistical inference on $\mu(x) = E[Y|X = x]$ (Wager & Athey, 2018). However, the results from Mentch & Hooker (2016) are valid for and centered at the expected prediction from a random forest, which can still provide much value to the research community. For example, linear models and ordinary least squares assume linear relationships between Y and predictors, yet researchers still use regression as an inference tool to learn about the relationships between Y

and its predictors. Thus, given their superiority in prediction over established methods and flexibility, developing and exploring procedures for random forests remain valuable for statistical inference (McAlexander & Mentch, 2020).

Chapter 2

Theory

This chapter focuses on the inference procedures described in Mentch & Hooker (2016). Specifically, we re-introduce the subbagging and subsampled random forest procedures that result in estimators in the form of U-statistics. Then, we describe the inference procedures developed by Mentch and Hooker to produce confidence intervals and tests of significance.

2.1 Ensemble Methods and U-Statistics

2.1.1 U-Statistics

First, we provide the definition of a U-statistic and provide some examples of U-statistics which the reader may find helpful. Consider n independent random vectors, $Z_1, \dots, Z_n \stackrel{iid}{\sim} F_{Z,\theta}$, where θ , the parameter of interest, is a function of $k \leq n$ arguments, and $F_{Z,\theta}$ is a marginal cumulative distribution function. Then, suppose there exists an unbiased estimator h of θ , a function of $k \leq n$ arguments. Then, we can write

$$\theta = \mathbb{E}h(Z_1, \dots, Z_k)$$

Without loss of generality, we may assume that h is permutation symmetric in its arguments, since it can be replaced with an equivalent permutation symmetric h . The minimum variance unbiased estimator for θ is given by

$$U_n = \frac{1}{\binom{n}{k}} \sum_{(i)} h(Z_{i_1}, \dots, Z_{i_k}) \tag{2.1}$$

where the sum is taken over all possible $\binom{n}{k}$ subsamples of size k from the training set with size n (Halmos, 1946). This is referred to as a U-statistic with kernel h of rank k .

Some examples of U-statistics include the sample mean to estimate μ . For $k = 1$, note that $h(X_1) = x_1$ is an unbiased estimator of μ , and the corresponding U-statistic is $U_n = \bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i$ is a U-statistic that many have encountered before. Similarly, we can also estimate variance as a U-statistic that takes $k = 2$ arguments: $\sigma^2 = E[(X - E[X])^2]$, estimated as $E(X_1^2 - X_1X_2)$. Although $f(X_1, X_2) = x_1^2 - x_1x_2$ is not symmetric in x_1 and x_2 , we find the corresponding symmetric form by taking the average: $h(x_1, x_2) = \frac{1}{2}(f(x_1, x_2) + f(x_2, x_1)) = \frac{x_1^2 - 2x_1x_2 + x_2^2}{2} = \frac{(x_1 - x_2)^2}{2}$. In this formulation of the U-statistic, note that the order in which x_1, x_2 appears does not affect the result. This leads to the U-statistic:

$$\begin{aligned} U_n &= \frac{2}{n(n-1)} \sum_{i < j} \frac{(X_i - X_j)^2}{2} \\ &= \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2 = s^2 \end{aligned}$$

which takes the common form of sample variance.

Hoeffding (1992) demonstrates, with fixed kernel and rank, these statistics are asymptotically normal with limiting variance $\frac{k^2}{n} \zeta_{1,k}$ where

$$\zeta_{1,k} = \text{cov}(h(Z_1, \dots, Z_k), h(Z_1', Z_2', \dots, Z_k')) \quad (2.2)$$

and $Z_2', \dots, Z_k' \stackrel{iid}{\sim} F_{Z,\theta}$. The 1 in the subscript $\zeta_{1,k}$ denotes the 1 shared observation between the two subsamples. In general, $\zeta_{c,k}$ denotes the covariance of the form in (2.2) of two subsamples with c observations in common. From these results, we can produce a subbagging procedure with asymptotically normal predictions.

2.1.2 Subbagging

In subbagging, we emulate Breiman (1996)'s bagging algorithm which produces predictions by averaging many trees' predictions, with each tree using a bootstrapped replicate of the original training set. In subbagging, instead, we obtain our predictions by averaging many trees' predictions, with each tree built using a *subsample* of the training set, taken without replacement.

This paragraph introduces notation used in subbagged trees.

- Z_i ; observation i of our training set

- $\mathbf{X} = (X_1, \dots, X_d)$; predictor vector
- Y_i ; observed response for the i 'th observation
- k_n ; size of subsample
- m_n ; number of subsamples taken

As our training set, let $Z_1 = (\mathbf{X}_1, Y_1), \dots, Z_n = (\mathbf{X}_n, Y_n) \stackrel{iid}{\sim} F_{\mathbf{X}, Y}$ where $\mathbf{X} = (X_1, \dots, X_d)$ is a vector of our predictors, and $Y \in \mathbb{R}$ is the response. Fix $k \leq n$ and take k observations from our training set to be our subsample $(\mathbf{X}_{i_1}, Y_{i_1}), \dots, (\mathbf{X}_{i_k}, Y_{i_k})$. If we are interested in making a prediction at feature vector in the predictor space $\mathbf{x}^* \in \mathcal{X}$, we can write the prediction at \mathbf{x}^* generated by a tree built using only the subsample $(\mathbf{X}_{i_1}, Y_{i_1}), \dots, (\mathbf{X}_{i_k}, Y_{i_k})$ as a function $T_{\mathbf{x}^*}$ from $(\mathcal{X} \times \mathbb{R}) \times \dots \times (\mathcal{X} \times \mathbb{R}) \rightarrow \mathbb{R}$. Taking every possible $\binom{n}{k}$ subsample, building a tree, and predicting at \mathbf{x}^* , our final subbagged prediction at \mathbf{x}^* is

$$b_n(\mathbf{x}^*) = \frac{1}{\binom{n}{k}} \sum_{(i)} T_{\mathbf{x}^*}((\mathbf{X}_{i_1}, Y_{i_1}), \dots, (\mathbf{X}_{i_k}, Y_{i_k})) \quad (2.3)$$

The estimator in (2.3) takes the form of a U-statistic, since tree-based estimators are symmetric and do not take into account the order of the training observations in their predictions. However, the number of computations to produce $\binom{n}{k}$ trees and predictions grows very quickly as n increases, so one work-around for computational efficiency is to build and average over $m_n < \binom{n}{k}$ trees. This estimator is called an *incomplete* U-statistic, and when the m_n subsamples are selected uniformly at random with replacement from the $\binom{n}{k}$ possible subsamples, the incomplete U statistic remains asymptotically normal (Janson, 1984; Lee, 1990).

Another refinement of the estimator is to grow k alongside with n so that the trees can be grown to a greater depth, producing more accurate predictions. Then, our estimator becomes:

$$b_{n,k_n,m_n}(\mathbf{x}^*) = \frac{1}{m_n} \sum_{(i)} T_{\mathbf{x}^*,k_n}((\mathbf{X}_{i_1}, Y_{i_1}), \dots, (\mathbf{X}_{i_{k_n}}, Y_{i_{k_n}})) \quad (2.4)$$

Frees (1989) discusses statistics of this form and calls them *Infinite order* U-statistics (IOUS) in the complete case, when $m_n = \binom{n}{k_n}$, and *resampled* statistics when $m_n < \binom{n}{k_n}$. Mentch & Hooker (2016) introduce a theorem, reprinted in 2.1, that provides a central limit theorem for estimators of the same form as (2.4) with respect to their individual means, $\mathbb{E}b_{n,k_n,m_n}(\mathbf{x}^*)$, with regularity conditions for asymptotic normality.

The following condition controls the tail behavior of the predictions, calling that

they do not grow too large, allowing us to satisfy the Lindeberg condition for part *i* of Theorem 2.1:

Condition 1. Let $Z_1, Z_2, \dots \stackrel{iid}{\sim} F_Z$ with $\theta_{k_n} = \mathbb{E}h_{k_n}(Z_1, \dots, Z_{k_n})$ and define $h_{1,k_n}(z) = \mathbb{E}h_{k_n}(z, \dots, Z_{k_n}) - \theta_{k_n}$. Then for all $\delta > 0$,

$$\lim_{n \rightarrow \infty} \frac{1}{\zeta_{1,k_n}} \int_{|h_{1,k_n}(Z_1)| \geq \delta \sqrt{n\zeta_{1,k_n}}} h_{1,k_n}^2(Z_1) dP = 0$$

The theorem is as follows:

Theorem 2.1. Let $Z_1, Z_2, \dots \stackrel{iid}{\sim} F_Z$ and let U_{n,k_n,m_n} be an incomplete, infinite order U -statistic with kernel h_{k_n} that satisfies Condition 1. Let $\theta_{k_n} = \mathbb{E}h_{k_n}(Z_1, \dots, Z_{k_n})$ such that $\mathbb{E}h_{k_n}^2(Z_1, \dots, Z_{k_n}) \leq C < \infty$ for all n and some constant C and let $\lim \frac{n}{m_n} = \alpha$. Then as long as $\lim \frac{k_n}{\sqrt{n}} = 0$ and $\lim \zeta_{1,k_n} \neq 0$,

$$(i) \text{ if } \alpha = 0, \text{ then } \frac{\sqrt{n}(U_{n,k_n,m_n} - \theta_{k_n})}{\sqrt{k_n^2 \zeta_{1,k_n}}} \xrightarrow{d} \mathcal{N}(0, 1).$$

$$(ii) \text{ if } 0 < \alpha < \infty, \text{ then } \frac{\sqrt{m_n}(U_{n,k_n,m_n} - \theta_{k_n})}{\sqrt{\frac{k_n^2}{\alpha} \zeta_{1,k_n} + \zeta_{k_n,k_n}}} \xrightarrow{d} \mathcal{N}(0, 1).$$

$$(iii) \text{ if } \alpha = \infty, \text{ then } \frac{\sqrt{m_n}(U_{n,k_n,m_n} - \theta_{k_n})}{\sqrt{\zeta_{k_n,k_n}}} \xrightarrow{d} \mathcal{N}(0, 1)$$

In our regression scenario, the following proposition provides a more intuitive condition that satisfies Condition 1:

Proposition 2.1. For a regression function F , if there exists a constant c such that for all $k_n \geq 1$,

$$|h((\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_{k_n}, Y_{k_n}), (\mathbf{X}_{k_n+1}, Y_{k_n+1})) - h((\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_{k_n}, Y_{k_n}), (\mathbf{X}_{k_n+1}, Y_{k_n+1}^*))| \leq c|Y_{k_n+1} - Y_{k_n+1}^*|$$

where $Y_{k_n+1} = F(\mathbf{X}_{k_n+1}) + \epsilon_{k_n+1}$, $Y_{k_n+1}^* = F(\mathbf{X}_{k_n+1}) + \epsilon_{k_n+1}^*$, and where ϵ_{k_n+1} and $\epsilon_{k_n+1}^*$ are i.i.d. with exponential tails, then Condition 1 is satisfied.

From 2.1, building trees with subsamples the size of the square root of the training set makes certain that the variance of the U -statistic in part (i) converges to 0. In addition, Mentch and Hooker note that the number of predictors may grow with n as long as Condition 1 is still upheld. Proofs of the proposition, theorem, and condition can be found in the appendix of Mentch & Hooker (2016). The subbagging procedure is similar to Breiman (1996)'s bagging procedure; however, rather than use a full

bootstrap sample for each tree, we use a smaller subsample with observations drawn without replacement.

In Chapter 3, we provide pseudo-code, as well as functions in R to run this procedure, and for the upcoming algorithms: producing subsampled random forest predictions, estimating the variance for predictions of our ensembles, producing confidence intervals for ensemble predictions, and performing significance tests for predictors.

2.1.3 Random Forests

In our subbagging procedure, the randomness between each built tree results from each tree having its own individual subbagged training set. However, the subbagging procedure calls for the same building method for each tree in our ensemble. In contrast, as denoted by its name, random forests do not follow this ordered, identical process for each tree in the ensemble. The random forest procedure in Breiman (2001a) instead selects a random subset of predictors at each node on which the tree is allowed to create a split upon.

Thus, Mentch & Hooker (2016) define ω , an additional randomization parameter that determines the subset of available predictors at each node. They define a *random kernel U-statistic* as

$$U_{\omega;n,k_n,m_n} = \frac{1}{m_n} \sum_i h_{k_n}^{(\omega_i)}(Z_{i_1}, \dots, Z_{i_{k_n}}) \quad (2.5)$$

with similar arguments as in the subbagging scenario. As a reminder: m_n corresponds to the number of trees we create, Z_i an observation from the training set, and k_n the size of our subsample. With \mathbf{x}^* as our test predictors vector, we can write the random forest estimator as

$$r_{n,k_n,m_n}(\mathbf{x}^*) = \frac{1}{m_n} \sum_i T_{\mathbf{x}^*,k_n}^{(\omega_i)}((\mathbf{X}_{i_1}, Y_{i_1}), \dots, (\mathbf{X}_{i_{k_n}}, Y_{i_{k_n}}))$$

Mentch & Hooker (2016) develop additional theory that take into account the additional randomness incorporated by the random forest algorithm. Suppose that the randomization parameters that determine which variables are used at each node $\omega_1, \dots, \omega_{m_n} \stackrel{iid}{\sim} F_\omega$, and that these randomization parameters are independent of the original training set Z_1, \dots, Z_n . Essentially, the variables to be used at each node are to be independent of our observations, which is the case in the original random forest

algorithm. Then, consider the statistic

$$\begin{aligned} U_{\omega;n,k_n,m_n}^* &= \mathbb{E}_\omega \left(\frac{1}{m_n} \sum_{(i)} h_{k_n}^{(\omega_i)}(Z_{i_1}, \dots, Z_{i_{k_n}}) \right) \\ &= \mathbb{E}_\omega U_{\omega;n,k_n,m_n} \end{aligned}$$

Taking the expectation with respect to ω , the kernel becomes fixed and hence $U_{\omega;n,k_n,m_n}^*$ follows the U-statistic theory noted above in the Subbagging chapter. Thus, $U_{\omega;n,k_n,m_n}^*$ is asymptotically normal as both a complete and incomplete U-statistic by Theorem 2.1. Then, to show asymptotic normality of the random kernel U-statistic, we need to show:

$$\sqrt{n}(U_{\omega;n,k_n,m_n}^* - U_{\omega;n,k_n,m_n}) \xrightarrow{P} 0$$

Equivalently, for any $\delta > 0, \epsilon > 0, \exists N \in \mathbb{N}$ such that for all $n \geq N, P(\sqrt{n}(U_n^* - U_n) > \delta) < \epsilon$.

The following theorem from Mentch & Hooker (2016) shows that under certain condition, that the random kernel U-statistic also follows asymptotic normality.

Theorem 2.2. *Let $U_{\omega;n,k_n,m_n}$ be a random kernel U-statistic of the form in (2.5), such that $U_{\omega;n,k_n,m_n}^*$ satisfies Condition 1, and suppose that $\mathbb{E}h_{k_n}^2(Z_1, \dots, Z_{k_n}) < \infty$ for all n , $\lim_{n \rightarrow \infty} \frac{k_n}{\sqrt{n}} = 0$, and $\lim_{n \rightarrow \infty} \frac{n}{m_n} = \alpha$. Then, letting β index the subsamples, so long as $\lim_{n \rightarrow \infty} \zeta_{1,k_n} \neq 0$ and*

$$\lim_{n \rightarrow \infty} \mathbb{E} \left(h_{k_n}^{(\omega)}(Z_{\beta_1}, \dots, Z_{\beta_{k_n}}) - \mathbb{E}_\omega h_{k_n}^{(\omega)}(Z_{\beta_1}, \dots, Z_{\beta_{k_n}}) \right)^2 \neq \infty$$

$U_{\omega;n,k_n,m_n}$ is asymptotically normal and the limiting distributions are the same as those provided in Theorem 2.1

Recall that Condition 1 calls for the tail behavior of the predictions to not grow too large. In the random kernel U-statistics scenario, we estimate the variance parameters ζ_{1,k_n} and ζ_{k_n,k_n} still using the covariance between the estimates generated by the trees. The last condition of Theorem 2.2

$$\lim_{n \rightarrow \infty} \mathbb{E} \left(h_{k_n}^{(\omega)}(Z_{\beta_1}, \dots, Z_{\beta_{k_n}}) - \mathbb{E}_\omega h_{k_n}^{(\omega)}(Z_{\beta_1}, \dots, Z_{\beta_{k_n}}) \right)^2 \neq \infty$$

guarantees that the randomization parameter ω does not produce predictions from the same subsample that contrast too differently as $n \rightarrow \infty$. This can occur through

the random selection of the variable choices at each node of the tree. This condition is satisfied easily, if the response Y is bounded, and for common implementations of random forests. Below is an algorithm to produce asymptotically normal predictions from a subbagged random forest. Similarly to Subbagging, the subbagged algorithm is analogous to Breiman (2001a)’s original procedure, except using subbagged samples for an individual tree rather than a full bootstrap sample. An observation is taken m_n times *without* replacement to build each subsample.

2.2 Estimating Variance

The limiting distributions in Theorem 2.1 depend on three parameters, θ_{k_n} , as well as ζ_{1,k_n} and ζ_{k_n,k_n} , which we need to establish consistent estimators for. As a reminder, a consistent estimator has the property that as the number of observations n used to estimate a parameter θ increases, the sequence of estimators converges in probability to θ as $n \rightarrow \infty$. For example, the sample mean \bar{X}_n is a consistent estimator for the population mean, μ , by the Weak Law of Large Numbers. The first parameter we must estimate is the unknown mean parameter, $\theta_{k_n} = \mathbb{E}U_{n,k_n,m_n}$. We can use the prediction from our ensemble as a consistent estimator for θ_{k_n} .

In equation (2.2), ζ_{1,k_n} and ζ_{k_n,k_n} were defined as the covariance between two kernels with 1 shared observation and two kernels with k_n shared observations respectively. Hence, the sample covariance could be consistent estimators for ζ_{1,k_n} and ζ_{k_n,k_n} ; however, Mentch & Hooker (2016) find that in practice, this estimator produces estimates close to 0, which then possibly leads to an overall negative variance estimate. Using Lee (1990), they then provide an equivalent expression for ζ_{c,k_n} ,

$$\zeta_{c,k_n} = \text{var}(\mathbb{E}(h_{k_n}(Z_1, \dots, Z_{k_n}) | Z_1 = z_1, \dots, Z_c = z_c))$$

For our estimator for ζ_{c,k_n} , we reconstruct the formula above. From our training set $\mathbf{Z}_1, \dots, \mathbf{Z}_n$, select c observations, which we denote as $\tilde{\mathbf{z}}_1, \dots, \tilde{\mathbf{z}}_c$. We will refer to these c observations as our “fixed points”, as we will use this set repeatedly. Now, take n_{MC} , MC for ‘Monte Carlo’, subsamples from our training set, making sure that each contain the set of fixed points. For each Monte Carlo subsample, construct a tree and predict at test point \mathbf{x}^* . Next, take the average of the n_{MC} predictions, resulting in our calculation of $\mathbb{E}(h_{k_n}(Z_1, \dots, Z_{k_n}) | Z_1 = z_1, \dots, Z_c = z_c)$. We repeat the steps for $n_{\tilde{\mathbf{z}}}$ sets of fixed points and take the variance of the $n_{\tilde{\mathbf{z}}}$ averages resulting in our final estimate of ζ_{c,k_n} .

$$\hat{\zeta}_{c,k_n} = \text{var} \left(\frac{1}{n_{MC}} \sum_{i=1}^{n_{MC}} T_{\mathbf{x}^*, k_n}(\mathcal{S}_{\tilde{\mathbf{z}}^{(1),i}}), \dots, \sum_{i=1}^{n_{MC}} T_{\mathbf{x}^*, k_n}(\mathcal{S}_{\tilde{\mathbf{z}}^{(\tilde{n}_z),i}}) \right)$$

with $\tilde{\mathbf{z}}^{(j)}$ denoting the j^{th} set of initial fixed points and $\mathcal{S}_{\tilde{\mathbf{z}}^{(j),i}}$ denoting the j^{th} subsample that includes $\tilde{\mathbf{z}}^{(j)}$. The algorithm to estimate ζ_{1,k_n} and ζ_{k_n,k_n} are provided in chapter 3

2.3 Inference Procedures

2.3.1 Setting the scene: Simple Linear Regression

Let us draw a parallel to inference procedures for a more traditional statistical model, Simple Linear Regression. In this subsection, we draw from DeGroot & Schervish (2012) to describe such procedures. Assumptions for Simple Linear Regression include:

- (1) that the predictor is known, values of x_1, \dots, x_n are known ahead of time,
- (2) normality, that the conditional distribution of Y_i given x_i is normal,
- (3) linear mean, that there are parameters β_0, β_1 such that the conditional mean of Y_i given x_1, \dots, x_n has the form $\beta_0 + \beta_1 x_i$ for $i = 1, \dots, n$, i.e., $\mathbb{E}(Y_i | x_1, \dots, x_n) = \beta_0 + \beta_1 x_i \forall i = 1, \dots, n$
- (4) common variance, that the conditional variance of Y_i given x_1, \dots, x_i is σ^2 for all $i = 1, \dots, n$,
- (5) independence, that the random variables Y_1, \dots, Y_n are independent given observed x_1, \dots, x_n .

Assume for each value $X = x$, we have a random variable Y , which can be represented as $Y = \beta_0 + \beta_1 x + \epsilon$, where ϵ is a random variable with mean 0 and variance σ^2 . Then, the conditional distribution of Y given $X = x$ is the normal distribution, with mean $\beta_0 + \beta_1 x$ and variance σ^2 . Given the assumptions above, the conditional joint p.d.f. of Y_1, \dots, Y_n is

$$f_n(\mathbf{y} | \mathbf{x}, \beta_0, \beta_1, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}} \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2$$

Then, we can solve for the Maximum Likelihood Estimators of β_0, β_1 , and σ^2 to get our Least Squares Estimators.

Specifically, these estimators are

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (Y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{x}$$

Under the assumptions stated above, $\hat{\beta}_1$ is distributed normally with mean β_1 and variance σ^2/s_x^2 , and $\hat{\beta}_0$ is distributed normally with mean β_0 and variance $\sigma^2(\frac{1}{n} + \frac{\bar{x}^2}{s_x^2})$, with $s_x = (\sum_{i=1}^n (x_i - \bar{x})^2)^{1/2}$. The covariance of $\hat{\beta}_1$ and $\hat{\beta}_0$ is $-\frac{\bar{x}\sigma^2}{s_x^2}$.

Now, we can study the distribution of a linear combination of $\hat{\beta}_0$ and $\hat{\beta}_1$, say $T = c_0\hat{\beta}_0 + c_1\hat{\beta}_1$. Since $\hat{\beta}_0$ and $\hat{\beta}_1$ are both distributed normally, their sum T will also be distributed normally with mean $c_0\beta_0 + c_1\beta_1$. To find the variance of T , we can substitute our values of $\text{Var}(\hat{\beta}_0)$, $\text{Var}(\hat{\beta}_1)$, and $\text{Cov}(\hat{\beta}_0, \hat{\beta}_1)$ to get

$$\begin{aligned} \text{Var}(T) &= c_0^2 \text{Var}(\hat{\beta}_0) + c_1^2 \text{Var}(\hat{\beta}_1) + 2c_0c_1 \text{Cov}(\hat{\beta}_0, \hat{\beta}_1) \\ &= \sigma^2 \left(\frac{c_0^2}{n} + \frac{(c_0\bar{x} - c_1)^2}{s_x^2} \right) \end{aligned}$$

Now that we have these important parameters, we can move onto hypothesis tests and confidence intervals for linear combinations of β_0 and β_1 . Suppose we wish to test the following hypotheses:

$$H_0 = c_0\beta_0 + c_1\beta_1 = c_*$$

$$H_1 = c_0\beta_0 + c_1\beta_1 \neq c_*$$

A level α_0 test of the hypotheses is to reject H_0 if $|U_{01}| \geq T_{n-2}^{-1}(1 - \alpha_0/2)$ where T_{n-2}^{-1} is the quantile function of the t distribution with $n - 2$ degrees of freedom, and

$$U_{01} = \left[\frac{c_0^2}{n} + \frac{(c_0\bar{x} - c_1)^2}{s_x^2} \right]^{-1/2} \left(\frac{c_0\hat{\beta}_0 + c_1\hat{\beta}_1 - c_*}{\sigma'} \right)$$

with $\sigma' = (\frac{S^2}{n-1})^{1/2}$. The variable U_{01} has the t distribution with $n - 2$ degrees of freedom, and is a function of the observed data alone.

To create a $1 - \alpha_0$ confidence interval for the $c_0\beta_0 + c_1\beta_1$, use the open interval between the two random variables:

$$c_0\hat{\beta}_0 + c_1\hat{\beta}_1 \pm \sigma' \left[\frac{c_0^2}{n} + \frac{(c_0\bar{x} - c_1)^2}{s_x^2} \right] T_{n-1}^{-1} \left(1 - \frac{\alpha_0}{2} \right)$$

In this case, c_* is between the two random variables if and only if $|U_{01}| < T_{n-2}^{-1}(1 - \alpha_0/2)$.

Another inference procedure we often perform with simple linear regression is a test of significance for a given predictor. For example, we want to test the hypotheses

$$H_0 : \beta_1 = 0$$

$$H_1 : \beta_1 \neq 0$$

This test is the same described above, simply with $c_0 = 0$, $c_1 = 1$, and $c_* = 0$. Then, using the formula for U_{01} , we obtain the following random variable $U_1 = s_x \frac{\hat{\beta}_1}{\sigma'}$, which has the t distribution with $n - 2$ degrees of freedom if H_0 is true, similar to the linear combination scenario. After calculating the value of U_1 from a data set (\mathbf{x}, \mathbf{y}) , call this value u_1 , we can obtain the corresponding p-value with $P(U_1 \geq |u_1|) + P(U_1 \leq -|u_1|)$. In other words, given that the null hypothesis is true, and X has no relationship with Y , the probability we observe our test statistic u_1 , or a test statistic of greater magnitude, is less than $P(U_1 \geq |u_1|) + P(U_1 \leq -|u_1|)$.

2.3.2 Confidence Intervals and Hypothesis Tests for Ensemble Methods

In the previous sections, we've established that our predictions, both through subbagging and subbagged random forests, are asymptotically normal, and that we can consistently estimate the parameters of the normal distributions. Thus, we can produce confidence intervals, given a training set and a prediction vector \mathbf{x}^* , similarly to how we would produce a confidence interval for the mean of a sample. Ordinarily, we would find the sample mean, \bar{x} , and sample variance, $\hat{\sigma}^2$, and then take the $\alpha/2$ and $1 - \alpha/2$ quantiles of the normal distribution with parameters \bar{x} as the mean, and $\hat{\sigma}^2$, for variance, to produce our lower bound, LB , and upper bound, UB , respectively, with α level confidence. Or in the simple linear regression case above, find the mean and variance of the linear combination $T = c_0\hat{\beta}_0 + c_1\hat{\beta}_1$, and take the $\alpha/2$ and $1 - \alpha/2$ quantiles of the T distribution to produce our upper and lower bound.

To produce confidence intervals for our subbagged (and subbagged random forest) predictions, first estimate the mean, $\hat{\theta}_{k_n}$, of our normal distribution by taking the mean of many subbagged trees (or random forests) predictions at the vector \mathbf{x}^* and $\frac{1}{\hat{\alpha}} \frac{k_n^2}{m} \hat{\zeta}_{1,k_n} + \frac{1}{m} \hat{\zeta}_{k_n,k_n}$ for our variance, taken from part (ii) of Theorem 2.1, with $\hat{\alpha} = n/m_n$. Our confidence intervals are the $\alpha/2$ and $1 - \alpha/2$ quantiles of this normal

distribution.

These confidence intervals can also be used to test hypotheses regarding the true mean prediction of our subbagged tree (or random forests). The null and alternate hypotheses are as follows:

$$H_0 : \theta_{k_n} = c$$

$$H_1 : \theta_{k_n} \neq c$$

The test statistic for this hypothesis test is

$$t = \frac{\hat{\theta}_{k_n} - c}{sd(\hat{\theta}_{k_n})}$$

and we reject H_0 if t is less than the $\frac{\alpha}{2}$ or greater than the $1 - \frac{\alpha}{2}$ quantile of the standard normal. The type 1 error rate of this test is $P(\text{reject } H_0 | H_0 \text{ true}) = P(t < \frac{\alpha}{2} \text{ or } t > 1 - \frac{\alpha}{2} | \theta_{k_n} = c) = \alpha$. This test is essentially a test of whether or not c is within our α level confidence interval. If $c \in [LB, UB]$, then we fail to reject our null hypothesis. On the other hand, if $c \notin [LB, UB]$ reject the null hypothesis that $\theta_{k_n} = c$.

As a reminder, recall that these confidence intervals are centered around the expected prediction θ_{k_n} , and not the true underlying regression function. We can compare this inference procedure with the commonly used ‘test set’ approach: withhold a small ‘test’ set from our training set, build a model onto the training set, and compare the model predictions on the test set with the actual responses. The ‘test set’ approach provides the user with information on the accuracy of the model at different locations within the feature space, but provides little information on the stability of the predictions. Instead, the confidence interval approach described above allows users to explore the prediction variability at specific points within the feature space, and answer the question of how much of the accuracy of predictions are due to chance.

2.3.3 Tests of Significance

The limiting distributions in Theorem 2.1 and Theorem 2.2 allow us to test for the significance of features. Often, we have a large amount of predictors, and we suspect that only a few of these predictors have a significant relationship with our response variable.

Consider the following scenario: Suppose that the training set has d predictors,

X_1, \dots, X_d , and consider a reduced set $\mathbf{X}^{(R)} \subset \{X_1, \dots, X_d\}$. Then, let $\mathbf{x}_{\text{TEST}} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ be a set of predictor vectors where we are interested in making predictions. Let g denote the function that maps predictor vectors to their prediction, and let $g^{(R)}$ denote the function that maps reduced predictor vectors from $\mathbf{X}^{(R)}$. For a single prediction point of interest, i.e. for a single predictor vector, \mathbf{x}_{TEST} , the true mean prediction θ_{k_n} generated by trees using the entire, full predictor space is $g(\mathbf{x}^*)$. Similarly, the true mean prediction generated by trees using the reduced predictor space is $g^{(R)}(\mathbf{x}^*) = \theta_{k_n}^{(R)}$. If none of the predictors removed to form the reduced predictor space play a significant role in the predictions of \mathbf{x}_{TEST} , then the predictions generated by the trees to form $g(\mathbf{x}_{\text{TEST}})$ and $g^{(R)}(\mathbf{x}_{\text{TEST}})$ should be very similar. In order to see the influence of the predictors *not* included in the reduced predictor space, we test the hypothesis

$$\begin{aligned} H_0 : g(\mathbf{x}_i) &= g^{(R)}(\mathbf{x}_i) \forall \mathbf{x}_i \in \mathbf{x}_{\text{TEST}} \\ H_1 : g(\mathbf{x}_i) &\neq g^{(R)}(\mathbf{x}_i) \text{ for some } \mathbf{x}_i \in \mathbf{x}_{\text{TEST}} \end{aligned}$$

Rejecting this null hypothesis means that some feature not included in the reduced predictor space has significant influence in the prediction of at least one test point.

To perform this hypothesis test given a training set of size n , first take m_n subsamples, denoted S_1, \dots, S_{m_n} , each of size k_n , and build a tree with each subsample. Then, given a prediction vector \mathbf{x}_i , take the average of the m_n trees to obtain $\hat{g}(\mathbf{x}_i)$. Using the same subsamples S_1, \dots, S_{m_n} , take the average of the prediction of m_n trees, built only with features from the reduced feature space $\mathbf{X}^{(R)}$, at \mathbf{x}_i to obtain $\hat{g}^{(r)}(\mathbf{x}_i)$. Define the difference between the two ensemble predictions as $\hat{D}(\mathbf{x}_i)$, which can be written as an incomplete U-statistic.

$$\begin{aligned} \hat{D}(\mathbf{x}_i) &= \hat{g}(\mathbf{x}_i) - \hat{g}^{(R)}(\mathbf{x}_i) \\ &= \frac{1}{m_n} \sum_{(j)} T_{\mathbf{x}_i, k_n}(S_j) - \frac{1}{m_n} \sum_{(j)} T_{\mathbf{x}_i, k_n}^{(R)}(S_j) \\ &= \frac{1}{m_n} \sum_{(j)} \left(T_{\mathbf{x}_i, k_n}(S_j) - T_{\mathbf{x}_i, k_n}^{(R)}(S_j) \right) \end{aligned}$$

Thus, given a single point of interest, \hat{D} is asymptotically normal, and so \hat{D}^2 is asymptotically χ_1^2 after the appropriate scaling. However, given that we are often interested in the significance of predictors at more than one point, then we can define

$\hat{\mathbb{D}}$ to be the vector of differences at multiple points

$$\hat{\mathbb{D}} = (\hat{D}(\mathbf{x}_1), \dots, \hat{D}(\mathbf{x}_N))$$

In the one variate case, each variable is normally distributed. As a result, the variable is either multivariate normal, or it does not have a joint density function with respect to Lebesgue measure. If it does have a joint density function with respect to Lebesgue measure, $\hat{\mathbb{D}}$ has a multivariate normal distribution with mean vector

$$\boldsymbol{\mu} = (g(\mathbf{x}_1) - g^{(R)}(\mathbf{x}_1), \dots, g(\mathbf{x}_N) - g^{(R)}(\mathbf{x}_N))^T$$

We estimate $\boldsymbol{\mu}$ with $\hat{\boldsymbol{\mu}} = \hat{\mathbb{D}}^T$. Additionally, we have a covariance matrix Σ , with parameters Σ_{1,k_n} and Σ_{k_n,k_n} . These are the multivariate analogues of ζ_{1,k_n} , and ζ_{k_n,k_n} , obtained by replacing the variance calculations in the algorithms for ζ_{1,k_n} and ζ_{k_n,k_n} with covariance. Like in the single variable case, our final result for Σ is $\frac{1}{\alpha} \frac{k_n^2}{m} \hat{\Sigma}_{1,k_n} + \frac{1}{m} \hat{\Sigma}_{k_n,k_n}$. In the next chapter are the algorithms, and code, to obtain estimates for Σ_{1,k_n} and Σ_{k_n,k_n} .

Finally, we can combine these estimates to form our test statistic, which will asymptotically follow the χ^2 distribution as the square of a standardized multivariate normal distribution, assuming the null hypothesis is true.

$$\hat{\boldsymbol{\mu}}^T \hat{\Sigma}^{-1} \hat{\boldsymbol{\mu}} \sim \chi_N^2$$

To test our original hypotheses, we compare $\hat{\boldsymbol{\mu}}^T \hat{\Sigma}^{-1} \hat{\boldsymbol{\mu}}$ to the $1 - \alpha$ quantile of the χ_N^2 distribution to produce a test with type 1 error rate α . If our test statistic is larger than this critical value, we reject the null hypothesis.

This section demonstrates the procedures to produce confidence intervals and hypothesis tests for predictions, as well as tests of significance for predictors. In the case for inference procedures regarding predictions, for both the tree-based ensemble models and simple linear regression model, we produce predictions that follow named distributions. In the linear regression case, we perform some modifications to transform our linear combination of $\hat{\beta}_0$ and $\hat{\beta}_1$ into a test statistic with the t distribution with $n - 2$ degrees of freedom. On the other hand, the subbagged ensembles produce predictions that follow the Normal distribution. Then, for both the tree-based ensembles and simple linear regression, we take quantiles from the appropriate distribution to form confidence intervals and hypothesis tests regarding the predictions from each model. In contrast, in testing the significance of a predictor, the tree-ensemble proce-

dure deviates quite a bit from the simple linear regression case. In order to perform the tree-ensemble significance tests, we must perform simulations to estimate the difference between trees that include the particular predictor and trees that exclude the predictor, as well as the variance of this difference.

Chapter 3

Algorithms and Code

In this section, we transcribe the pseudo-code provided in Mentch & Hooker (2016) for the subbagging, subsampled random forest, and variance estimation procedures. We also provide our own code, supplemented with comments, to implement said algorithms in R.

3.1 Subbagging Algorithm

This pseudo-code describes the algorithm for Subbagging described in Section 2.1.2. In this procedure, the user provides a training data set, the subsample size k_n , number of subsamples m_n , and a test vector, \mathbf{x}^* , within the predictor space to produce a subbagged estimate $b_{n,k_n,m_n}(\mathbf{x}^*)$.

Algorithm 1: Subbagging Algorithm

Input : training set, size of subsamples k_n , number of subsamples m_n , test point \mathbf{x}^*

Output: subbagged estimate $b_{n,k_n,m_n}(\mathbf{x}^*)$

- 1 Load training set
 - 2 Select size of subsamples k_n , and number of subsamples m_n
 - 3 **for** i in 1 to m_n **do**
 - 4 Take subsample of size k_n from training set
 - 5 Build tree using subsample
 - 6 Use tree to predict at test point, \mathbf{x}^*
 - 7 **end**
 - 8 Take the mean of the m_n predictions to get final estimate $b_{n,k_n,m_n}(\mathbf{x}^*)$
-

Below, we demonstrate an implementation of the Subbagging algorithm in R.

```

1  #load necessary libraries
2  library(tidyverse)
3  library(rpart)
4
5  #Subbagging Function
6  subbag_func <- function(df, m, k, x_star, cp = .01){
7
8    predictions <- data.frame(y_hat = rep(NA, m))
9    for (i in 1:m) {
10      subsample_index <- sample(1:nrow(df), k, replace = FALSE)
11      subsample <- df[subsample_index,]
12      subsample.tree <- rpart(y ~ . , data = subsample,
13                             control = rpart.control(minsplit = 3,
14                                                         cp = cp))
15      prediction <- predict(subsample.tree, newdata = x_star)
16      predictions[i,1] <- prediction
17    }
18    y_hat_av <- mean(predictions$y_hat)
19
20    return(y_hat_av)
21  }

```

Lines 2-3 load libraries. Tidyverse contains several general use functions, and rpart builds our regression trees (Therneau, Atkinson, Ripley, & Ripley, 2015; Wickham et al., 2019). Lines 6-21 contain the body of our function. We first allocate an empty dataframe of size m , in which we'll store the predictions from our individual trees. Then repeat the following steps m times:

- select a subsample of size k from the initial training set without replacement (lines 10)
- build a tree with the subsample (line 12)
- predict at \mathbf{x}^* using newly built tree (line 15)
- place prediction in dataframe and repeat

Finally, we take the mean of our predictions (line 18) and return this value to get our final estimate $b_{n,k_n,m_n}(\mathbf{x}^*)$.

Note one extra input for our R function, cp . The complexity parameter is an argument for rpart which determines how deep each tree grows. The user provides a

value so that if a split does not improve the R^2 of the model by cp , the program will not split the branch any further. Generally, a larger cp value will lead to shallower trees. Because Mentch & Hooker (2016) do not specify a particular cp value, we provide a default value for this parameter, but allow a user to specify their preferred cp value.

3.2 Subsampled Random Forest Algorithm

The subsampled random forest algorithm is very similar to the previously mentioned subbagging procedure. Note the inclusion of the random variable selection in line 5 of the pseudo-code.

Algorithm 2: Subsampled Random Forest Algorithm

Input : training set, size of subsamples k_n , number of subsamples m_n , test point \mathbf{x}^*

Output: final estimate $r_{n,k_n,m_n}(\mathbf{x}^*)$

- 1 Load training set
 - 2 Select size of subsamples k_n , and number of subsamples m_n
 - 3 **for** i in 1 to m_n **do**
 - 4 Take subsample of size k_n from training set
 - 5 Build tree using subsample, based on randomization parameter ω
 - 6 Use tree to predict at test point, \mathbf{x}^*
 - 7 **end**
 - 8 Take the mean of the m_n predictions to get final estimate $r_{n,k_n,m_n}(\mathbf{x}^*)$
-

```

1  #loading library
2  library(randomForest)
3
4  #subsampled random Forest Function
5  rf_subsamp_func <- function(df, m, k, x_star, mtry = 3){
6
7    predictions_rf <- data.frame(y_hat = rep(NA, m))
8    for (i in 1:m) {
9      subsample_index <- sample(1:nrow(train), k, replace = FALSE)
10     subsample <- train[subsample_index,]
11     subsample.rf <- randomForest(y ~ . , data = subsample,
```

```

12         ntree = 1,
13         mtry = mtry)
14 prediction <- predict(subsample.rf, newdata = x_star)
15 predictions_rf[i,1] <- prediction
16 }
17 y_hat_av <- mean(predictions_rf$y_hat)
18
19 return(y_hat_av)
20 }

```

The procedure to produce random forest subbagged estimates is very similar to the previous subbagging procedure. The key difference lies on lines 11-13. In this case, we use the `randomForest` package to build a single tree using the `subsample`. The key difference is that at each node of the tree, a number of variables, with Mentch & Hooker (2016)'s use of 3 as a default value, is randomly chosen for the tree to make its split. Similarly to the `cp` parameter in the subbagging algorithm, the user is able to specify another value for `mtry` according to their preferences.

3.3 Variance Estimation Algorithms

Next, we describe algorithms to estimate the variance parameters, ζ_{1,k_n} and ζ_{k_n,k_n} , for θ_{k_n} . Recall, from Section 2.2, that ζ_{c,k_n} represents the covariance between two kernels with c shared arguments, or equivalently, the variance between the expected value of our kernel, given c known observations. We use the following formula, from Section 2.2 as an estimator for ζ_{c,k_n} :

$$\hat{\zeta}_{c,k_n} = \text{var} \left(\frac{1}{n_{MC}} \sum_{i=1}^{n_{MC}} T_{\mathbf{x}^*,k_n}(\mathcal{S}_{\ddagger(\infty),\rangle}^{\ddagger}), \dots, \sum_{i=1}^{n_{MC}} T_{\mathbf{x}^*,k_n}(\mathcal{S}_{\ddagger(\ddagger),\rangle}^{\ddagger}) \right)$$

To get estimates for ζ_{1,k_n} and ζ_{k_n,k_n} , use the following algorithm:

Algorithm 3: ζ_{1,k_n} Estimation Procedure

Input : training set, number of initial fixed point sets $n_{\tilde{z}}$, number of Monte Carlo samples n_{MC} , test point \mathbf{x}^*

Output: ζ_{1,k_n} estimate

```

1 for  $i$  in 1 to  $n_{\tilde{z}}$  do
2   | Select initial fixed point  $\tilde{z}^{(i)}$ 
3   | for  $j$  in 1 to  $n_{MC}$  do
4   |   | Select subsample  $S_{\tilde{x}^{(i)},j}$ 
5   |   | Build tree using subsample  $S_{\tilde{x}^{(i)},j}$ 
6   |   | Use tree to predict at test point,  $\mathbf{x}^*$ 
7   | end
8   | Record average of  $n_{MC}$  predictions
9 end
10 Compute variance of the  $n_{\tilde{z}}$  averages

```

The equivalent code in R is as follows:

```

1 var_1_finder <- function(df, n_z, n_mc, x_star, cp = .01){
2
3   predictions <- rep(NA, n_mc)
4   pred_means <- rep(NA, n_z)
5   for (i in 1:n_z) {
6     index <- sample(1:nrow(df),1)
7     z_tilde <- df[index,]
8     for (j in 1:n_mc) {
9       vec = 1:nrow(df)
10      vec = vec[-index]
11      subsample_index <- sample(vec, k-1, replace = FALSE)
12      subsample <- df[subsample_index,]
13      subsample[k,] <- z_tilde
14      subsample.tree <- rpart(y ~ . , data = subsample,
15                             control = rpart.control(minsplit = 3,
16                                                         cp = cp))
17      prediction <- predict(subsample.tree, newdata = x_star)
18      predictions[j] <- prediction
19    }

```

```

20   pred_means[i] <- mean(predictions)
21 }
22 zeta_1_kn <- var(pred_means)
23 return(zeta_1_kn)
24
25 }
```

In the above code, lines 3-4 preallocate vectors, one in which we store the predictions from our subbagged trees. The other is a vector to store the mean of those predictions. We repeat the following $n_{\tilde{z}}$ times:

- Select our initial, single fixed point, which will be shared among the n_{MC} Monte Carlo subsamples. Now, repeat the following steps n_{MC} times:
 - For each fixed point, manually exclude it from the training set, and then select $k - 1$ observations from the training set without replacement (lines 9-12).
 - Then, manually include the fixed point, \tilde{z} , so that it is included in each subsample exactly once (line 13).
 - We then build a tree using the subsample, and place its prediction at \mathbf{x}^* within the predictions vector, which should eventually be n_{MC} long (lines 14-18).
- We place the mean of each prediction vector in the mean vector, $n_{\tilde{z}}$ long (line 20).

Finally, return the variance of the $n_{\tilde{z}}$ averages (lines 20-23).

The procedure to estimate ζ_{k_n, k_n} is very similar, and also easier to implement. Because the kernels share k_n observations with subsamples on size k_n , there is no need to take the average of multiple Monte Carlo subsamples, as each subsample will be the same. Then, the algorithm amounts to computing the variance of $n_{\tilde{z}}$ predictions from trees built with subsamples of size k_n .

Algorithm 4: ζ_{k_n, k_n} Estimation Procedure**Input** : training set, number of initial fixed point sets n_z , test point \mathbf{x}^* **Output:** ζ_{1, k_n} estimate

```

1 for  $i$  in 1 to  $n_z$  do
2   | Select subsample  $S_{\tilde{\mathbf{x}}^{(i)}, j}$ 
3   | Build tree using subsample  $S_{\tilde{\mathbf{x}}^{(i)}, j}$ 
4   | Use tree to predict at test point,  $\mathbf{x}^*$ 
5 end
6 Compute variance of the  $n_z$  predictions

```

```

1 var_k_finder <- function(df, n_z, x_star, cp = .01){
2
3   predictions <- rep(NA, n_z)
4   for (i in 1:n_z) {
5     subsample_index <- sample(1:nrow(df), k, replace = FALSE)
6     subsample <- df[subsample_index,]
7     subsample.tree <- rpart(y ~ ., data = subsample,
8                             control = rpart.control(minsplit = 3,
9                                                         cp = cp))
10    prediction <- predict(subsample.tree, newdata = x_star)
11    predictions[i] <- prediction
12  }
13  zeta_k_n_kn <- var(predictions)
14
15  return(zeta_k_n_kn)
16 }

```

3.4 Building Confidence Intervals

In Chapter 2, we established estimators for the parameters for the limiting normal distributions from our subbagged ensemble and subsampled random forest predictions. In this chapter, we have developed code to implement algorithms to estimate the parameters. With our new estimates, we are now ready to build confidence intervals for our subbagged and subsampled random forest predictions. Following Mentch

& Hooker (2016), we consider two underlying regression functions:

$$\text{SLR: } g(x_1) = 2x_1; \mathcal{X} = [0, 20]$$

$$\text{MARS: } g(\mathbf{x}) = 10 \sin(\pi x_1 x_2) + 20(x_3 - .05)^2 + 10(x_4) + 5x_5; \mathcal{X} = [0, 1]^5$$

In both scenarios, the features were chosen at random uniformly, and responses were sampled from $g(\mathbf{x}) + \epsilon$, with $\epsilon \stackrel{iid}{\sim} \mathcal{N}(0, 10)$ to form our training sets.

Below, we provide code to generate one instance of these training sets.

```

1  set.seed(2)
2  n = 1000
3  #Simple training set
4  x_1 <- runif(n, min = 0, max = 20)
5  e <- rnorm(n, mean = 0, sd = sqrt(10))
6  y = 2*x_1 + e
7  df <- data.frame('y' = y,
8                   'x_1' = x_1)
9
10 #MARS training set
11 x_1 <- runif(n, min = 0, max = 1)
12 x_2 <- runif(n, min = 0, max = 1)
13 x_3 <- runif(n, min = 0, max = 1)
14 x_4 <- runif(n, min = 0, max = 1)
15 x_5 <- runif(n, min = 0, max = 1)
16 e <- rnorm(n, mean = 0, sd = sqrt(10))
17 y = 10*sin(pi*x_1*x_2) + 20*(x_3-.05)^2 + (10*x_4) + (5*x_5) + e
18
19 df <- data.frame('y' = y,
20                  'x_1' = x_1,
21                  'x_2' = x_2,
22                  'x_3' = x_3,
23                  'x_4' = x_4,
24                  'x_5' = x_5)

```

Recall that these confidence intervals are to encompass expected prediction θ_{k_n} , not necessarily the true value of the underlying function. To estimate the true mean prediction θ_{k_n} at $x_1 = 10$, we take the mean of the predictions of many subbagged

ensembles, in this case, the mean of the predictions of 1000 subbagged ensembles.

```

1  #change eval = TRUE when knitting final document
2  get_true_mean_simple <- function(n, m, k, x_star, cp = .01){
3    true_means <- rep(NA, 1000)
4    for (i in 1:1000) {
5      x_1 <- runif(n, min = 0, max = 20)
6      e <- rnorm(n, mean = 0, sd = sqrt(10))
7      y = 2*x_1 + e
8      df <- data.frame('y' = y,
9                       'x_1' = x_1)
10
11     true_means[i] <- subbag_func(df, m, k,
12                                x_star, cp = cp)
13     if (i %% 10 == 0){
14       print(c(i, true_means[i]))
15     }
16   }
17
18   true_mean <- mean(true_means)
19   return(true_mean)
20 }
21 set.seed(2)
22 true_mean_simple <- get_true_mean(n, m, k, x_star, cp = .01)
23 true_mean_simple

```

In our runthrough, the mean prediction of the subbagged ensemble is 20.00688, which is very close to the true value in the regression function of 20, as well as very similar to the 20.02 found by Mentch & Hooker (2016).

The code below produces a confidence interval in the SLR case. Following Mentch & Hooker (2016), we first begin with $n = 200, m = 200, k = 30$, and predict at $x_1 = 10$. After generating our data, we use the *subbag_func* function to estimate the mean of the normal limiting distribution, and the two variance estimation functions to get the variance parameters. We estimate ζ_{1,k_n} using $n_{\bar{z}} = 50$ and $n_{MC} = 250$. We estimate ζ_{k_n,k_n} by calculating the variance between 500 predictions from 500 new subsamples.

```

1  #setting scenario
2  set.seed(2)
3  n = 200
4  m = 200
5  k = 30
6  reps = 250
7  n_z = 50
8  n_mc = 250
9  cp = .01
10 alpha_hat = n/m
11 x_star <- data.frame(x_1 = 10)
12
13 #generate data
14 x_1 <- runif(n, min = 0, max = 20)
15 e <- rnorm(n, mean = 0, sd = sqrt(10))
16 y = 2*x_1 + e
17 df <- data.frame('y' = y,
18                  'x_1' = x_1)
19
20 #estimating parameters
21 var_1 <- var_1_finder(df, n_z, n_mc, x_star, cp)
22 var_k <- var_k_finder(df, 500, x_star, cp)
23 variance <- (((k^2)/alpha_hat)/m * var_1) + var_k/m
24 mean <- subbag_func(df, m, k, x_star, cp)
25
26 LB <- qnorm(.025, mean , sd = sqrt(variance))
27 UB <- qnorm(.975, mean , sd = sqrt(variance))
28 CI_df <- data.frame(LB = LB, UB = UB)
29
30 CI_df

```

LB UB

1 17.62607 21.50348

Table 3.1: Coverage Probabilities for Simple/MARS

Underlying Function	n	k	cp	θ_{k_n}	Coverage Probability
SLR	200	30	0.01	20.00688	0.920
SLR	200	30	0.00	19.99000	0.920
SLR	1000	60	0.01	19.98775	0.948
SLR	1000	60	0.00	19.99411	0.908
MARS	500	50	0.01	17.40834	0.992
MARS	500	50	0.00	17.55247	1.000
MARS	1000	75	0.01	17.43780	1.000
MARS	1000	75	0.00	17.55574	0.992

Repeating the process above 250 times produces 250 confidence intervals. In our experiment, 230/250 of the confidence intervals contained the “true mean” prediction, resulting in a coverage probability of .92, slightly lower than the .95 we would expect. We repeat the process for scenarios, such as SLR with deep trees ($cp = 0$), SLR with larger training sets ($n = 1000$, $m = 1000$, $k = 60$), the MARS case with $n = m = 500$ and $k = 50$, MARS with $n = m = 1000$ and $k = 75$, and the same MARS training sets with deep trees. noting their results in the table 3.1. We achieve similar results compared to Mentch & Hooker (2016). In addition to the experiments with various test sizes, we also provide the results with ensembles composed with deep trees.

The impact of changing the cp value in the tree building mechanism is unclear from our results. In some cases, changing the cp value from 0.01 to 0.00 resulted in lower coverage rates, for example, the SLR ($n = 1000$) and MARS ($n = 1000$) cases. In other scenarios, changing the cp had no effect on coverage rate, SLR ($n = 200$), or even increased coverage rate MARS ($n = 500$). Figures 3.1 and 3.2 display all 250 confidence intervals in each n, m, cp scenario.

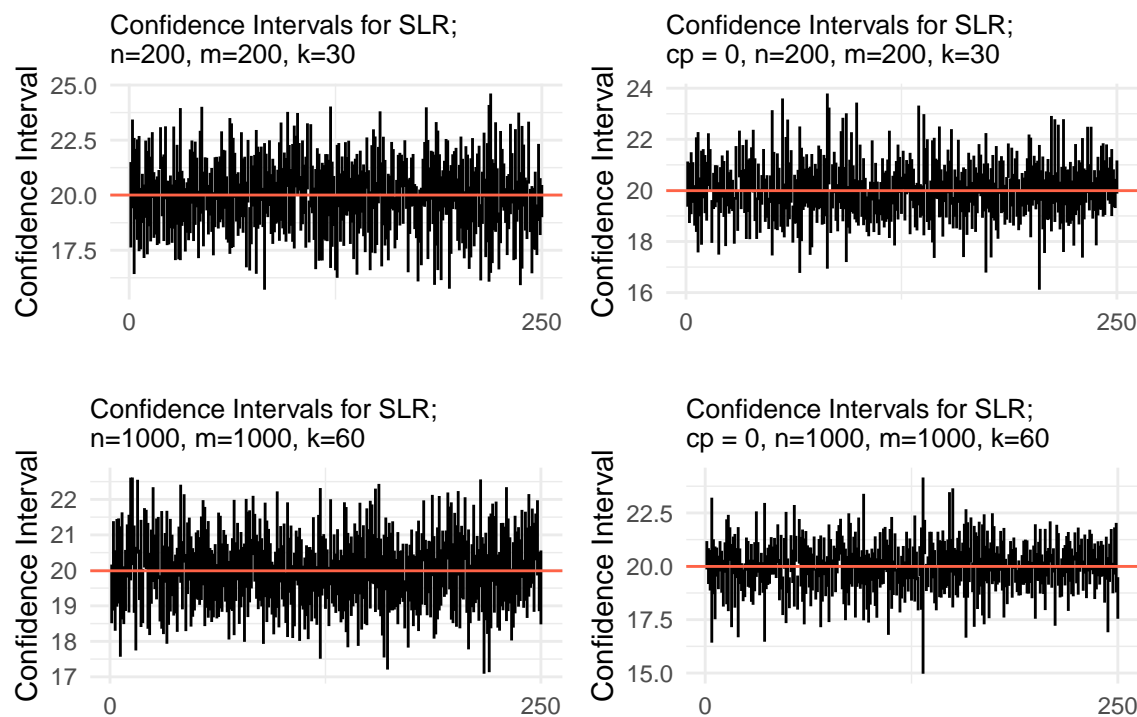


Figure 3.1: Confidence Intervals for Simple Scenario

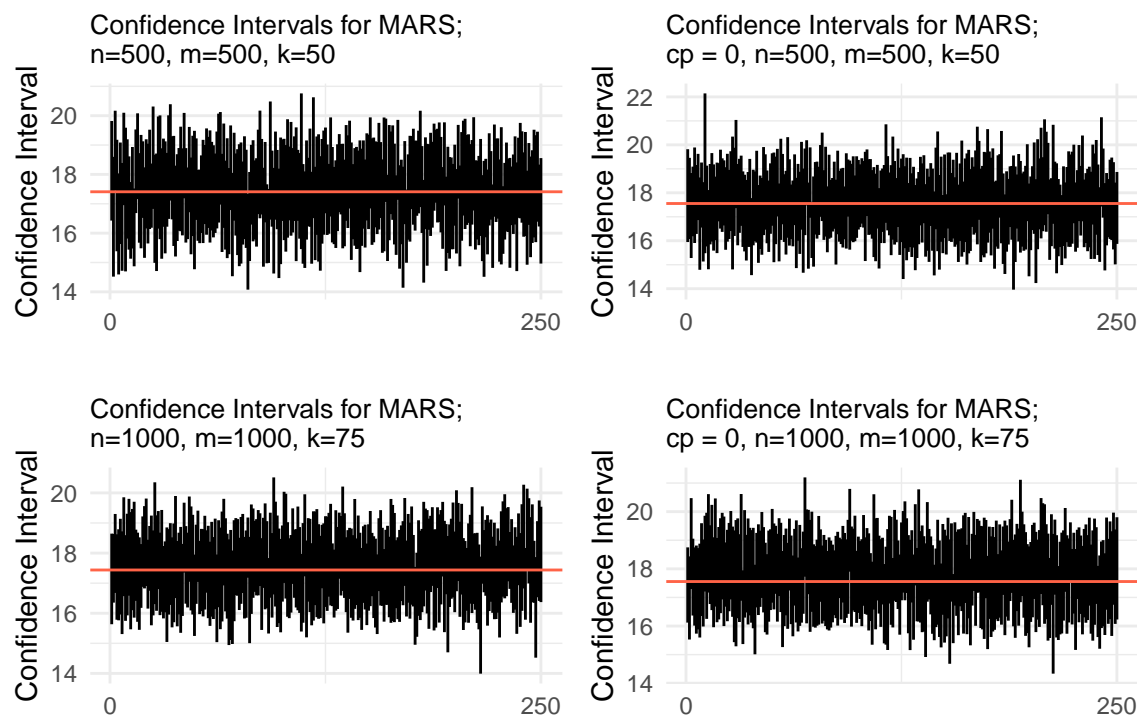


Figure 3.2: Confidence Intervals for MARS Scenario

3.5 Performing Significance Tests

As noted in Chapter 2, in order to test the significance of certain predictors, we must create new functions that allow us to test for the difference in the true mean predictions of trees built using the full predictor space and the true mean predictions of trees built using a reduced predictor space. Consider the following functions:

First the function to produce the vector of observed differences in the predictions, \hat{D} .

```

1 ensemble_tree_hp <- function(df, excluded_var,
2                             test_set, m, k, cp = .01){
3   #setting up the function: preallocating vectors, defining variable
4   excluded_var <- enquo(excluded_var) #define variable to exclude
5   prediction_full <- rep(NA, dim(test_set)[1])
6   prediction_r <- rep(NA, dim(test_set)[1])
7   predictions <- list(y_hat_full = rep(NA, m),
8                      y_hat_r = rep(NA, m))
9   #for loop to store vector of predictions into predictions list
10  for (i in 1:m) { #take m subsamples
11    subsample_index <- sample(1:nrow(df), k, replace = FALSE)
12    subsample <- df[subsample_index,]
13    subsample_r <- subsample %>%
14      select(-!!excluded_var)
15    subsample.tree <- rpart(y ~ . , data = subsample,
16                          control = rpart.control(minsplit = 3,
17                                                  cp = cp))
18    subsample_r.tree <- rpart(y ~ . , data = subsample_r,
19                            control = rpart.control(minsplit = 3,
20                                                  cp = cp))
21    for (j in 1:dim(test_set)[1]) {
22      prediction_full[j] <- predict(subsample.tree,
23                                  newdata = test_set[j,])
24      prediction_r[j] <- predict(subsample_r.tree,
25                                newdata = test_set[j,])
26    }
27    predictions[[1]][i] <- list(prediction_full)
28    predictions[[2]][i] <- list(prediction_r)

```

```

29 }
30 y_hat_av_full <- vector(length = length(predictions$y_hat_full[[1]]))
31 y_hat_av_r <- vector(length = length(predictions$y_hat_full[[1]]))
32 empty_vec_full <- vector(length = m)
33 empty_vec_r <- vector(length = m)
34 for (j in 1:length(predictions$y_hat_full[[1]])){
35   for (i in 1:m){
36     empty_vec_full[i] <- predictions$y_hat_full[[i]][j]
37     empty_vec_r[i] <- predictions$y_hat_r[[i]][j]
38   }
39   y_hat_av_full[j] <- mean(empty_vec_full)
40   y_hat_av_r[j] <- mean(empty_vec_r)
41 }
42 dif_vec <- y_hat_av_full - y_hat_av_r
43
44 return(list(y_hat_av_full = y_hat_av_full,
45            y_hat_av_r = y_hat_av_r,
46            dif_vec = dif_vec))
47 }

```

In the above function, *ensemble_tree_hp*, lines 3-7 perform general set up. We prepare the to-be-excluded variable for use, and set aside two vectors, one for the full space predictions and one for the reduced space predictions. Each of these vectors is exactly the length of our test set, and each element of the vector is to be a prediction using a test observation. We also create a list of length m , the number of subsamples, that will hold each vector of predictions, both full and reduced. Now, we repeat the following m times:

- produce a subsample by sampling k observations from the training set without replacement (lines 10-12)
- remove to-be-excluded variable from the subsample to create our reduced predictor space subsample (line 13-14)
- create two trees, built with the full subsample and reduced subsample. Then with each tree, predict at each test point in our test set, and place into the two ‘prediction’ vectors that were allocated in lines 3-7. (lines 15-25)

- place entire predictions vectors into respective full and reduced predictions list (lines 27-28)

Now, create two vectors with length equal to the test set, and two vectors with equal length of m . The m length vectors are filled with the predictions made at a single test point, each made from the m different trees. Place predictions from the tree built from both the full predictor space into one vector, and place other predictions from the reduced predictor space into the other vector (lines 35-38). Find the average of the m predictions at each test point, and place into the test set length vectors (lines 39-40). Finally, we obtain our difference vector, with length equal to the test set, by subtracting the mean predictions of the reduced-space predictions vector from the full-space predictions vector (line 42). The function returns a list of the full and reduced space predictions vector, as well as the difference vector, and each vector can be easily extracted from the output. To produce our estimate $\hat{\mu}$, we simply take the transpose of our difference vector.

In order to produce our χ^2 test statistic, we must also estimate the our covariance matrix Σ . To do so, we must estimate the covariance matrix parameters Σ_{1,k_n} and Σ_{k_n,k_n} , the multivariate analogues of ζ_{1,k_n} and ζ_{k_n,k_n} , to get our estimate for $\Sigma = \frac{1}{\hat{\alpha}} \frac{k_n^2}{m} \hat{\Sigma}_{1,k_n} + \frac{1}{m} \hat{\Sigma}_{k_n,k_n}$. Next, we go over the R code to produce these covariance matrix parameters. As a reminder, m represents the number of subsamples we take, k_n the size of each subsample, and $\hat{\alpha} = n/m$.

3.5.1 Variance Estimation Algorithms for Multiple Test Points

Algorithm 5: Σ_{1,k_n} Estimation Procedure

Input : training set, number of initial fixed point sets $n_{\tilde{z}}$, number of Monte Carlo samples n_{MC} , test points \mathbf{x}_N^*

Output: ζ_{1,k_n} estimate

```

1 for  $i$  in 1 to  $n_{\tilde{z}}$  do
2   Select initial fixed point  $\tilde{z}^{(i)}$ 
3   for  $j$  in 1 to  $n_{MC}$  do
4     Select subsample  $S_{\tilde{x}^{(i)},j}$  that includes  $\tilde{x}^{(i)}$ 
5     Build *full* tree using subsample  $S_{\tilde{x}^{(i)},j}$ 
6     Build *reduced* tree using subsample  $S_{\tilde{x}^{(i)},j}$  using only reduced
      prediction space
7     Use full tree to predict at test points,  $\mathbf{x}_N^*$ 
8     Use reduced tree to predict at test points,  $\mathbf{x}_N^*$ 
9     Record difference in predictions
10  end
11  Record average of  $n_{MC}$  predictions
12 end
13 Compute covariance of the  $n_{\tilde{z}}$  averages

```

```

1 var_1_finder_hp <- function(df, excluded_var, test_set,
2                             n_z, n_mc, k, cp = .01){
3   excluded_var <- enquo(excluded_var)
4   mean_difs <- vector(mode = "list", length = n_z)
5   diff_in_predictions <- vector(mode = "list", length = n_mc)
6   prediction_full <- rep(NA, dim(test_set)[1])
7   prediction_r <- rep(NA, dim(test_set)[1])
8   for (i in 1:n_z) {
9     index <- sample(1:nrow(df), 1)
10    z_tilde <- df[index,]
11    for (j in 1:n_mc) {
12      vec = 1:nrow(df)
13      vec = vec[-index]
14      subsample_index <- sample(vec, k-1, replace = FALSE)

```

```

15     subsample <- df[subsample_index,]
16     subsample[k,] <- z_tilde #
17     subsample.tree <- rpart(y ~ . , data = subsample,
18                             control = rpart.control(minsplit = 3,
19                                                         cp = cp))
20     subsample_r <- subsample %>%
21       select(-!!excluded_var)
22     subsample_r.tree <- rpart(y ~ . , data = subsample_r,
23                               control = rpart.control(minsplit = 3,
24                                                         cp = cp))
25     for (p in 1:dim(test_set)[1]) {
26       prediction_full[p] <- predict(subsample.tree,
27                                     newdata = test_set[p,])
28       prediction_r[p] <- predict(subsample_r.tree,
29                                  newdata = test_set[p,])
30     }
31     diff_in_predictions[[j]] <- (prediction_full - prediction_r)
32   }
33   sum_dif <- diff_in_predictions[[1]]
34   for (p in 2:length(diff_in_predictions)) {
35     sum_dif <- sum_dif + diff_in_predictions[[p]]
36   }
37   mean_dif <- sum_dif / length(diff_in_predictions)
38   mean_difs[[i]] <- mean_dif
39 }
40 mean_dif_matrix <- as.matrix(t(data.frame(mean_difs)))
41 rownames(mean_dif_matrix) <- NULL
42
43 return(cov(mean_dif_matrix))
44 }

```

Algorithm 6: Σ_{k_n, k_n} Estimation Procedure**Input** : training set, number of initial fixed point sets $n_{\tilde{z}}$, test points \mathbf{x}_N^* **Output:** ζ_{1, k_n} estimate

- 1 **for** i *in* 1 *to* $n_{\tilde{z}}$ **do**
- 2 Select subsample $S_{\tilde{x}^{(i)}, j}$
- 3 Build tree using subsample $S_{\tilde{x}^{(i)}, j}$
- 4 Use tree to predict at test point, \mathbf{x}^*
- 5 **end**
- 6 Compute variance of the $n_{\tilde{z}}$ averages

```

1 var_k_finder_hp <- function(df, excluded_var, test_set,
2                             n_z, k, cp = .01){
3   excluded_var <- enquo(excluded_var)
4   pred_difs <- vector(mode = "list", length = n_z)
5   prediction_full <- rep(NA, dim(test_set)[1])
6   prediction_r <- rep(NA, dim(test_set)[1])
7   for (i in 1:n_z) {
8     subsample_index <- sample(1:nrow(df), k, replace = FALSE)
9     subsample <- df[subsample_index,]
10    subsample.tree <- rpart(y ~ ., data = subsample,
11                           control = rpart.control(minsplit = 3,
12                                                    cp = cp))
13    subsample_r <- subsample %>%
14      select(-!!excluded_var)
15    subsample_r.tree <- rpart(y ~ ., data = subsample_r,
16                             control = rpart.control(minsplit = 3,
17                                                    cp = cp))
18    for (p in 1:dim(test_set)[1]) {
19      prediction_full[p] <- predict(subsample.tree,
20                                  newdata = test_set[p,])
21      prediction_r[p] <- predict(subsample_r.tree,
22                                newdata = test_set[p,])
23    }
24    pred_difs[[i]] <- (prediction_full - prediction_r)
25  }
26  pred_dif_matrix <- as.matrix(t(data.frame(pred_difs)))

```

```
27 rownames(pred_dif_matrix) <- NULL
28
29 return(cov(pred_dif_matrix))
30 }
```

The functions for the estimation of the variance parameters of Σ follow a similar focus on the differences between trees built using the full subsample and trees built using a reduced sample. The algorithms for Σ_{1,k_n} and Σ_{k_n,k_n} are very similar to the single-variable estimation algorithms, only returning the covariance between mean predicted differences rather than the variance, since we are now working with matrices. In the next chapter, we present the results from simulations for the significance tests.

Chapter 4

Significance Test Simulation

In this section, we present the results of simulations using the significance test mentioned above on the following cases, SIMPLE and POLY. We investigate the effectiveness of the test, namely its Type 1 and Type 2 error, and we record complications and difficulties that arose during this process.

4.1 Significance Test: SIMPLE

First, we proceed with a simple case, denoted SIMPLE, with a simple linear relationship between Y and one predictor X_1 . The predictor X_2 has an independent relationship with Y . Each predictor is generated randomly with the Uniform distribution from $[0,1]$. For these significance tests, we choose a single test point $\mathbf{x} = (x_1 = .5, x_2 = .5)$.

```
1 x_1 <- runif(n, min = 0, max = 1)
2 x_2 <- runif(n, min = 0, max = 1)
3 e <- rnorm(n, mean = 0, sd = sqrt(10))
4 y = 5*x_1 + e
5 df <- data.frame('y' = y,
6                  'x_1' = x_1,
7                  'x_2' = x_2)
8 test_set <- data.frame('x_1' = .5,
9                        'x_2' = .5)
```

With this simple scenario, we perform many simulations to gauge the effectiveness of the proposed significance test. In our simulations, we test the following null hypotheses, first that X_1 has no relationship with Y , and second, that X_2 has no

relationship with Y . Formally, we test the hypotheses below:

$$H_0 : g(x_i) = g^{(R)}(x_i); x_i = (x_1 = .5, x_2 = .5)$$

$$H_1 : g(x_i) \neq g^{(R)}(x_i); x_i = (x_1 = .5, x_2 = .5)$$

First, we proceed with the case where $X^{(R)} = \{X_2\}$, when we test and exclude X_1 . Next, we test X_2 's significance and $X^{(R)} = \{X_1\}$. Because Y has a relationship with only X_1 , a test with a low type 1 error, the probability of rejecting the null hypothesis given it is true, should rarely reject the null hypothesis when excluding X_2 , since X_2 is unrelated to Y . On the other hand, a test with low type 2 error, an error resulting from not rejecting a false null hypothesis, should reject the null hypothesis often when excluding X_1 , since X_1 has a direct relationship with Y .

After performing 100 simulations for significance tests, excluding X_1 and then X_2 , we obtain the following results:

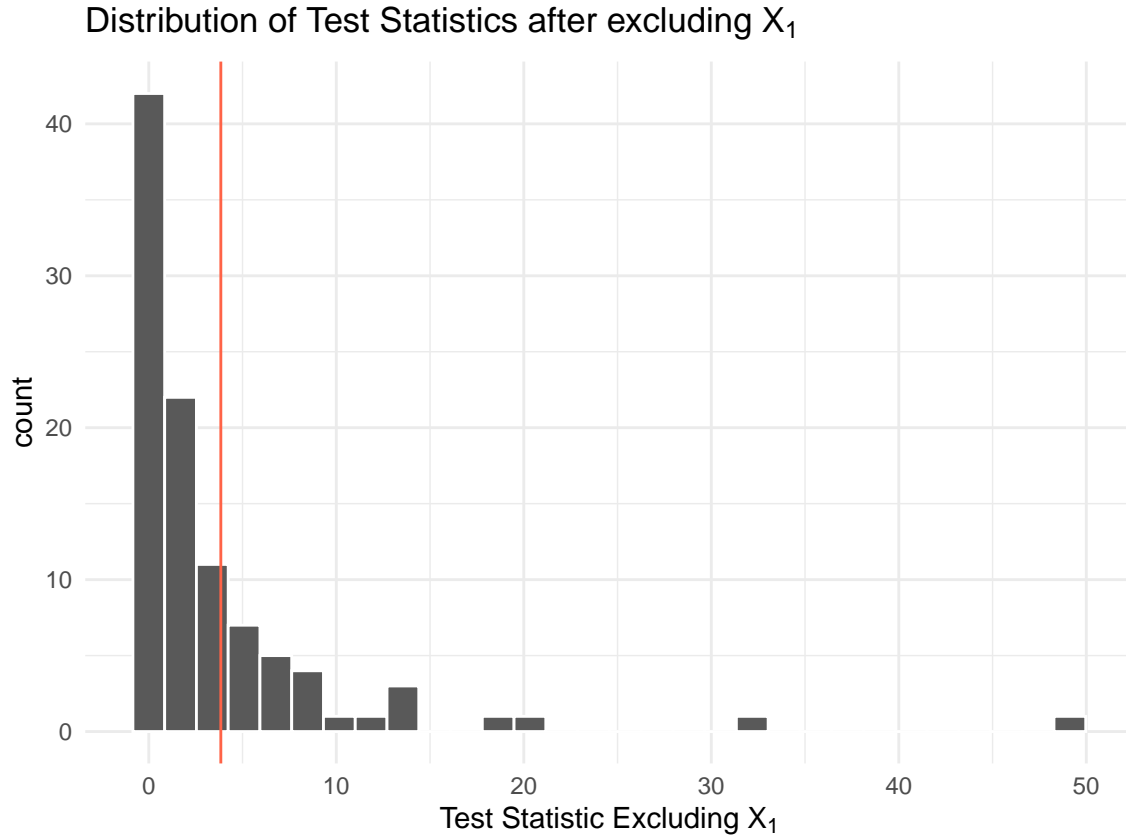


Figure 4.1: Histogram of Test Statistics Testing the Significance of X_1

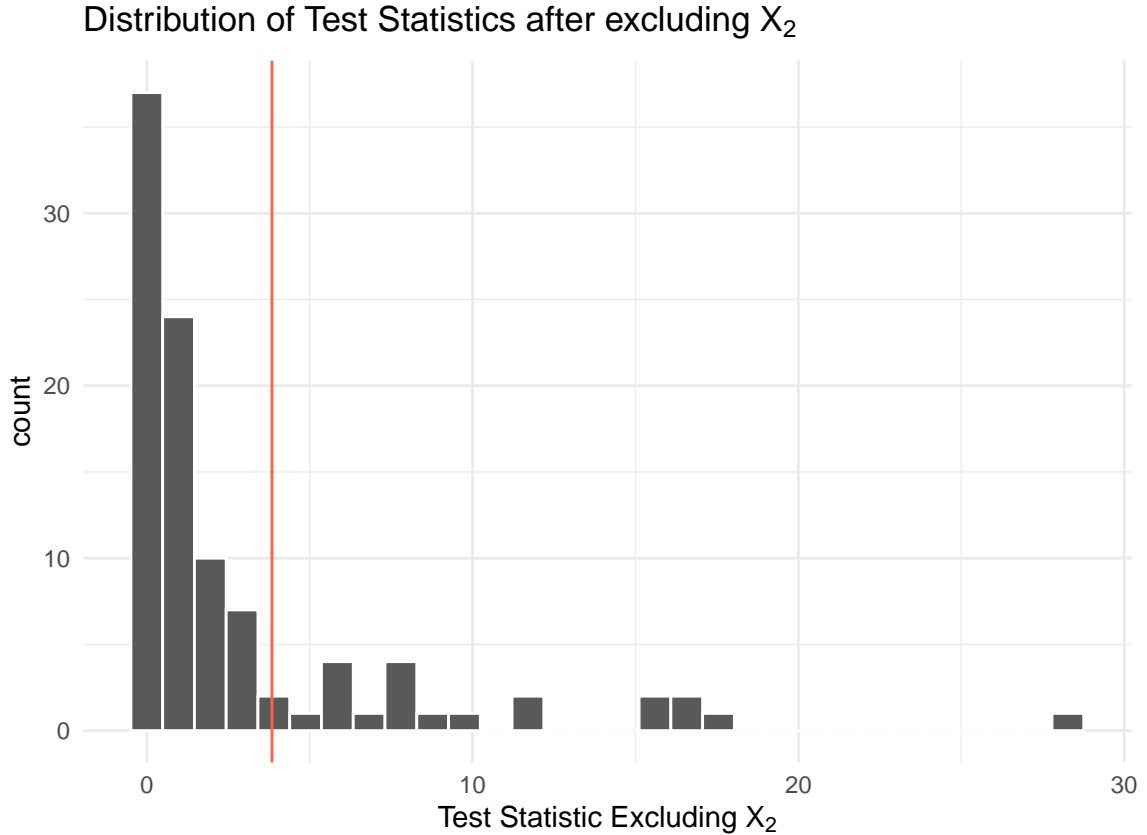


Figure 4.2: Histogram of Test Statistics Testing the Significance of X_2

In both Figures 4.1 and 4.2, the red vertical line represents the edge of the critical region for the hypothesis test. This line depicts the 95'th quantile of the χ_1^2 distribution. If the null hypothesis were true, and there is no difference between the functions mapping the full and reduced feature space to the true mean prediction, only 5% of test statistics would appear equal to or greater than this value. Thus, for all hypothesis tests whose test statistics lie greater than this value, we reject the null hypothesis.

Our simulation resulted in test statistics that rejected the null hypothesis correctly 27 times out of 100 simulations. In other words, 27 of the test statistics in Figure 4.1 were greater than the 95'th quantile of the χ_1^2 distribution. On the other hand, Figure 4.2 depicts the results when we compare a full predictor space versus a reduced space with only X_1 . Ordinarily, we would expect only 5 of the 100 simulations to incorrectly reject the null hypothesis since Y has a relatively simple relationship with only X_1 . However, Figure 4.2 illustrates many test statistics greater than our cut-off value. In fact, we observe 21 test statistics greater than the 95'th quantile of the χ_1^2 distribution, which was unexpected.

Table 4.1: Summary Statistics for SIMPLE simulation

Excluded Var.	Median $ \hat{\mu} $	Median $\hat{\Sigma}$	Median Test Stat.
X1	0.4766149	0.1337234	1.2759371
X2	0.2676363	0.0775247	0.9342868

Ordinarily, we expect the test statistics in the X_1 exclusion case to be larger than the test statistics in the X_2 exclusion case, if the test is to reject the null hypothesis when the null hypothesis is false and not reject the null when the null is true. We find slight evidence for this rationalization in our simulation. Both the mean and the median for the test statistics in the X_1 exclusion case are larger than the corresponding mean and median for the test statistics in the X_2 exclusion case. As a whole, the distribution of the X_1 excluded test statistics has a slightly heavier right tail than the distribution of the X_2 excluded test statistics. While Figures 4.1, 4.2, and Table 4.1 show a general difference between the test statistics in the two tests, ideally, the significance test would have higher power and reject the null hypothesis more often in the X_1 exclusion case, i.e. decreasing our type 2 error. Additionally, we would want to decrease our type 1 error, i.e. decreasing the rejections in the X_2 exclusion case.

In a future extension of this work, we encourage the researcher to experiment empirically, through further simulation, on different choices for the variance of the difference in predictions, for example, the sum of the variances of the full space predictions and variance of reduced space predictions. Another course of study could be into the variance of the estimators that produce our sample mean and variance, leading to our test statistic. Consider a scenario where we examine the variance of a different test statistic, the sample variance of observations drawn from a normal sample. Note that $\frac{(N-1)S^2}{\sigma^2} \sim \chi_{n-1}^2$. Then, after some algebra, we find that $\text{Var}(S^2) = \frac{2\sigma^4}{(n-1)}$. Note that the sample variance for these observations from a normal distribution increase with the variance of our observations, and decrease when n is large. Also note that the predictions from a single regression tree are highly variable, although we do seek to reduce this variance by aggregating over many trees, and through the random forest variable subset mechanism. In our simulation, we find that the estimation for Σ did result in a wide range of values for $\hat{\Sigma}$. This in turn possibly brought about a high number of false positives when investigating then unrelated predictor, X_2 , where we incorrectly reject the null hypothesis due to a large test statistic. McAlexander & Mentch (2020) also note that this procedure may sometimes produce a high number of false positives because of the variability of the tree algorithm. Because the trees

in the random forest are grown to maximum depth, they are very receptive to false predictors and can alter the predictions the random forest makes to a significant degree. Future work can include investigations into the pruning of trees and whether asymptotic normality still exists for trees grown under this method.

4.2 Significance Test: POLY

Another issue that appeared during the simulations arose when solving for $\hat{\Sigma}^{-1}$ in the test statistic. After excluding certain variables, estimating the variance parameters Σ_{1,k_n} and Σ_{k_n,k_n} to produce $\hat{\Sigma}$, attempting to invert $\hat{\Sigma}$ would lead to an error: “system is exactly singular”.

Consider the following data set, denoted POLY, where Y has polynomial relationships with predictors X_1, X_2, X_3, X_4, X_5 . The predictor X_6 has no relationship with Y . Each predictor is generated randomly from the Uniform distribution from $[0,1]$.

```

1  set.seed(11)
2  n = 1000
3  x_1 <- runif(n, min = 0, max = 1)
4  x_2 <- runif(n, min = 0, max = 1)
5  x_3 <- runif(n, min = 0, max = 1)
6  x_4 <- runif(n, min = 0, max = 1)
7  x_5 <- runif(n, min = 0, max = 1)
8  x_6 <- runif(n, min = 0, max = 1)
9  e <- rnorm(n, mean = 0, sd = sqrt(10))
10 y <- 2*x_1 + 4*x_1^2 + 3*x_2^2 + 4*x_3^3 + .1*x_5 + e
11 df <- data.frame('y' = y,
12                   'x_1' = x_1,
13                   'x_2' = x_2,
14                   'x_3' = x_3,
15                   'x_4' = x_4,
16                   'x_5' = x_5,
17                   'x_6' = x_6)

```

We then test for the significance of X_6 with 5 different set of test points. The first test set included 101 evenly dispersed points, with the first test point being $x_1 = \dots = x_6 = 0$, the second $x_1 = \dots = x_6 = 0.01$, and so on, with the last point being $x_1 = \dots = x_6 = 1$. We performed the significance test described above;

however, the test failed, as we could not invert the variance matrix $\hat{\Sigma}$. We perform the test again, with 41 evenly dispersed points, which also failed to complete. However, the significance tests successfully complete when the test sets reach sizes 21, 11, and 6, and they all find that X_6 does not play a significant role in the ensemble tree predictions. But why did the testing procedure not function in the first two cases with larger test sets?

Recall how a regression tree assigns a prediction given a new observation: break the predictor space into rectangular areas, in which the tree assigns the same prediction to any observation that falls within the same area (see Figures 4.3 and 4.4). We hypothesize that this prediction assignment leads to computation issues when trying to solve the estimated covariance matrix $\hat{\Sigma}$ when the test set is particularly dense. The columns in our covariance matrix were linearly dependent on each other. Note that each entry in $\hat{\Sigma}_{1,k_n}$ and $\hat{\Sigma}_{k_n,k_n}$ is the covariance of the mean difference in predictions from a full to reduced space with another mean difference in predictions. Then, one potential issue that leads to a singular, non-invertible variance matrix is the fact that when test points are very close together, it is likely that the ensemble methods make the same predictions between multiple test points, resulting in the same mean difference between multiple test points. Then, this can lead to the columns of our variance parameters matrices to become identical, giving rise to an un-invertible $\hat{\Sigma}$. Under these conditions, we cannot produce a test statistic following Mentch & Hooker (2016)'s procedure.

Tree to Find Flipper Length

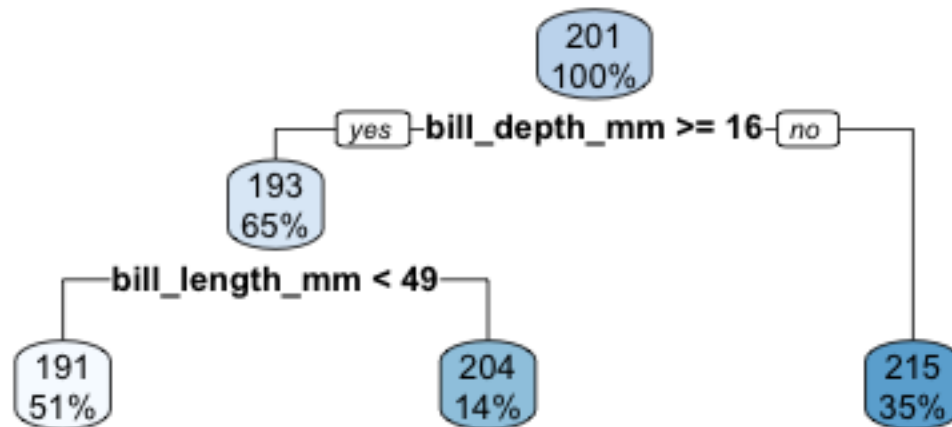


Figure 4.3: Simple Regression Tree for Flipper Length

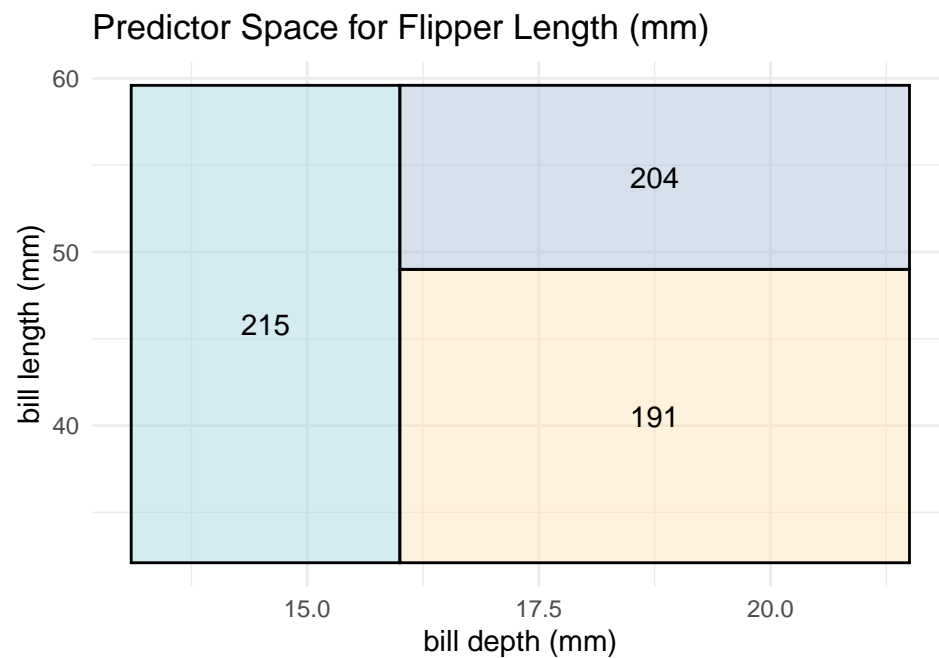


Figure 4.4: Partition of Predictor Space According to Flipper Length Regression Tree

Chapter 5

Conclusion

5.1 Findings and Results

In this thesis, we have described the random forest algorithm and two main branches of study on inference technique on random forests, Mentch & Hooker (2016) and Wager & Athey (2018). Following Mentch & Hooker (2016), we present ensemble predictions as U-statistics, leading way to asymptotic normality, which we can build inference procedures based on. In particular, we study and recreate Mentch & Hooker (2016)'s inference procedures, confidence intervals for random forest predictions and significance tests for predictors. In Chapter 3, we provide pseudo-code and functions to generate subbagged and random forest predictions, estimates of variance parameters, and to generate confidence intervals and perform significance tests in R. Finally, Chapter 4 presents the findings of a simple simulation, and describes particular issues that arose during this procedure.

5.2 Limitations

This work focuses on subbagged ensembles as a whole, and develops theory for subbagged random forests as a subclass of subbagged ensembles. However, the simulations for the confidence intervals and significance tests are built off of the general class of subbagged ensembles. A future extention of this work can perform an in-depth investigation into the subbagged random forest method and perform simulations specific to the random forest.

Additionally, the scope of the project and simulations are limited to generated data from relatively simple distributions. As noted in the next section, a natural follow-up

to the project is to test its utility on real data. Finally, the unexpected results in the significance test simulations merit further investigation. Will additional simulations, perhaps on new data, exhibit the same results?

5.3 Future Extensions

Future extensions of this thesis include performing tests and simulations on more unique data generation methods, for example, predictors generated from the Cauchy distribution. Another data generation related experiment can utilize different types of variables, such as using categorical variables for predictors and as a response. Another natural extension to the thesis are inference procedures using real, messy data, and comparing the random forest inference to results from the traditional regression procedure.

A more theoretical extension of the work could perform a deeper study into the properties of the parameter estimates and the significance test statistic. For example, a future researcher could extend from Mentch & Hooker (2017) to study how the choice of the test set impacts the hypothesis test. In addition to continuing the analysis on the Mentch & Hooker (2016) branch of random forest inference, another avenue for research is performing a parallel analysis on the Wager & Athey (2018) work.

Appendix A

Code

A github repository that stores the code used in my thesis can be found at `https://github.com/paulhnguyen/thesis_code`

References

- 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 (with comments and a rejoinder by the author). *Statistical Science*, 16(3), 199–231.
- DeGroot, M. H., & Schervish, M. J. (2012). *Probability and statistics*. Pearson Education.
- Dietterich, T. G. (1998). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, 32, 1–22.
- Frees, E. W. (1989). Infinite order u-statistics. *Scandinavian Journal of Statistics*, 29–45.
- Friedman, J. H., & Hall, P. (2007). On bagging and nonlinear estimation. *Journal of Statistical Planning and Inference*, 137(3), 669–683.
- Halmos, P. R. (1946). The theory of unbiased estimation. *The Annals of Mathematical Statistics*, 17(1), 34–43.
- Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8), 832–844.
- Hoeffding, W. (1992). A class of statistics with asymptotically normal distribution. In *Breakthroughs in statistics* (pp. 308–334). Springer.
- Horst, A. M., Hill, A. P., & Gorman, K. B. (2020). *Palmerpenguins: Palmer archipelago (antarctica) penguin data*. <http://doi.org/10.5281/zenodo.3960218>
- Janson, S. (1984). The asymptotic distributions of incomplete u-statistics. *Zeitschrift Für Wahrscheinlichkeitstheorie Und Verwandte Gebiete*, 66(4), 495–505.
- Lee, A. (1990). U-statistics: Theory and applications. Marcel Dekker Inc., New York.
- McAlexander, R. J., & Mentch, L. (2020). Predictive inference with random forests: A new perspective on classical analyses. *Research & Politics*, 7(1), 2053168020905487.
- Mentch, L., & Hooker, G. (2016). Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *The Journal of Machine Learning Research*,

- 17(1), 841–881.
- Mentch, L., & Hooker, G. (2017). Formal hypothesis tests for additive structure in random forests. *Journal of Computational and Graphical Statistics*, 26(3), 589–597.
- Therneau, T., Atkinson, B., Ripley, B., & Ripley, M. B. (2015). Package “rpart”. *Available Online: Cran. Ma. Ic. Ac. Uk/Web/Packages/Rpart/Rpart. Pdf (Accessed on 20 April 2016)*.
- Wager, S., & Athey, S. (2018). Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523), 1228–1242.
- Wager, S., Hastie, T., & Efron, B. (2014). Confidence intervals for random forests: The jackknife and the infinitesimal jackknife. *The Journal of Machine Learning Research*, 15(1), 1625–1651.
- Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., . . . Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4(43), 1686. <http://doi.org/10.21105/joss.01686>