Senior Thesis in Mathematics

Non-parametric Alternative Techniques for Propensity Score Estimation

Author:
Emma Godfrey

Advisor:
Dr. Jo Hardin

Submitted to Pomona College in Partial Fulfillment
of the Degree of Bachelor of Arts

April 8, 2021

Abstract

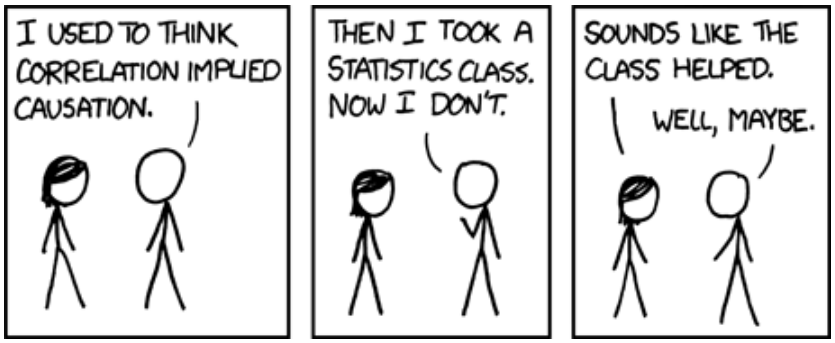

Figure 1: Correlation is not causation! (source: https://xkcd.com/552/)

Statisticians and students worldwide feverishly chant the phrase "correlation is not causation". While accurate, the blanket statement disproportionately focuses statisticians' attention on correlation while dismissing possible causal mechanisms. In my thesis, I discuss recent research detailing the use of propensity scores as a covariate balancing technique for causal inference. More specifically, I outline logistic regression and various supervised machine learning techniques including CART, ensemble CART methods, and neural networks as methods for propensity score estimation. Then, I simulate a causal relationship within a data set and apply each technique to assess which estimation technique is most effective at uncovering the causal relationship. Does the effectiveness of the estimation technique depend on the size of the causal effect? What about the sample size? What about a misspecified model? I will explore these questions.

Contents

1 Introduction to Propensity Scores **1**
- 1.1 Correlation vs. Causation . . . 1
- 1.2 Notation and Average Treatment Effect . . . 2
- 1.3 Balancing Score . . . 4
  - 1.3.1 Propensity Score Definition . . . 4
- 1.4 Propensity Score Stratification . . . 5
- 1.5 Evaluating the Propensity Score Usefulness . . . 6
  - 1.5.1 Bias Reduction . . . 6
  - 1.5.2 Pre-treatment Covariate Hypothesis Testing . . . 7
  - 1.5.3 Checking for Covariate Balance After Stratification Example . . . 8
- 1.6 Estimating the Average Treatment Effect . . . 13

2 Logistic Regression for Propensity Score Estimation **15**
- 2.1 Generalized Linear Models . . . 15
- 2.2 Logistic Regression Theory . . . 16
  - 2.2.1 First, Some Notation . . . 16
  - 2.2.2 Logistic Regression as a GLM . . . 16
- 2.3 Pros and Cons of Logistic Regression . . . 20

3 Tree-based methods for Propensity Score Estimation **21**
- 3.1 Motivation for CART Methods . . . 21
- 3.2 Classification and Regression Trees (CART) . . . 22
- 3.3 Classification Tree Ensemble Learners . . . 25
  - 3.3.1 Bagged CART . . . 25
  - 3.3.2 Boosted CART . . . 29

4 Neural Networks for Propensity Score Estimation **35**
4.1 Motivation for Neural Networks 35
4.2 The Perceptron 36
4.3 Multi-Layer Neural Networks 38
4.3.1 Training a Neural Network with Backpropagation . . . 40
4.4 Pros and Cons of Neural Networks for Propensity Score Estimation 43

5 Simulation and Results **45**
5.1 Treatment Assignment Mechanism 46
5.2 Simulation 1: Vary Treatment Effect Size and Sample Size . . 46
5.3 Simulation 2: Vary Pre-treatment Covariate Strength 49
5.4 Simulation 3: Model Misspecification 51
5.4.1 Simulation 3a: Unspecified Interaction Term 52
5.4.2 Simulation 3b: Covariate Exclusion 54
5.5 Conclusions and Takeaways 55

Chapter 1

Introduction to Propensity Scores

1.1 Correlation vs. Causation

Scientific research suggests a *correlation* between quitting smoking and an increase in body weight [Audrain-McGovern and Benowitz, 2011]. Suppose I am interested in researching whether quitting smoking, on average, *causes* a change in body weight. In this study, quitting smoking is called the **explanatory variable** while change in body weight is called the **response variable**. Although prior research suggests a strong correlation between quitting smoking and change in body weight, the correlation does not prove causation.

As the name suggests, causation implies event A, quitting smoking, caused event B, an increase in body weight. In contrast, correlation implies that there exists a relationship between event A and event B. While it is tempting to assume causation given strong correlation, there exist many possible explanations for a relationship with strong correlation. For instance,

1. The opposite could be true: B causes A
2. There is an unobserved causal variable: A and B could be correlated, but both events are caused by C
3. There exists a dependency in the relationship: A causes B as long as D happens

4. There is a chain reaction: A causes E, and E then causes B

In all of the above instances, it is scientifically incorrect to conclude that A causes B. In items 2-4, there exists what is called a **confounding variable**. A confounding variable is a variable which affects **both** the explanatory and response variable. For instance, access to healthcare could affect both a subject's change in body weight and whether the subject quits smoking.

Randomized Controlled Trials (RCTs) are often considered the gold standard for causal inference. In RCTs, proper subject randomization establishes highly similar distributions of covariates in both treatment and control groups. With equal distribution of covariates (i.e., equal distribution of all possible confounding variables), one can estimate the treatment effect by comparing the outcomes of those in the treated group to those in the control.

While RCTs are ideal, conducting an experiment with sufficient randomization and control is expensive and often unethical. In the smoking and bodyweight example, it would be completely unethical to tell half of subject to quit smoking and the other half to continue smoking! Due to ethics and accessibility, observational studies are vastly more common. Unfortunately, observational studies make it nearly impossible to conclude causation due to the abundance of confounding variables and unequal distribution of covariates across the untreated and treated groups. Prior literature suggests that the usage of propensity scores can indirectly allow scientists to conclude causation and estimate an unbiased treatment effect in observational settings [Rosenbaum and Rubin, 1984].

1.2 Notation and Average Treatment Effect

Consider a study in which we are trying to estimate the effect of a treatment on a response variable. Here, let 1 and 0 denote two different treatments. Given N participants, consider the ith where $i = 1, \ldots, N$. The ith participant will either receive the treatment 1 or treatment 0. If we let R denote the random response variable, then r_{i1} is the response associated with the ith participant receiving treatment 1 and r_{i0} the response associated with the ith participant receiving treatment 0.

Causal effects are comparisons of r_{i1} and r_{i0}, typically $r_{i1} - r_{i0}$ [Rosenbaum and Rubin, 1984]. Mathematically, the average treatment effect is defined as

follows:

$$ATE = E(r_1) - E(r_0) \text{ where E(*) is the population average} \tag{1.1}$$

Let $z_i = 1$ if the ith individual is assigned to the treatment group and $z_i = 0$ if assigned to control group. Moreover, let $\mathbf{x}_i$ denote a vector of pretreatment observed covariate measures for individual i. For instance, if we are interested in drawing a causal relationship between quitting smoking and change in weight, logical pretreatment covariates would likely include: sex, income, smoking intensity, smoking duration, access to heath care, alcohol use, etc. For each subject i, their covariate measures comprise the vector $\mathbf{x}_i$.

Since each participant receives only a single treatment, either r_{i1} or r_{i0} must be unknown. The unrealized response for subject i is called the **counterfactual**. In RCTs, this unknown response is not an issue as proper subject randomization ensures that, on average, the treated and untreated populations do not systematically differ [Rosenbaum and Rubin, 1984]. In terms of expected value, for every pretreatement covariate x, $E(x \mid z = 1) = E(x \mid z = 0)$. Thus, in RCTs, simply comparing the average treated response to the average untreated response provides an unbiased estimate for the average treatment effect.

However, observational studies introduce systematic differences between the treated population and untreated population. In terms of expected value, for any pretreatment covariate x, $E(x \mid z = 1) \neq E(x \mid z = 0)$. On average, the values of an observed pretreatment covariate are different for the treated and the untreated. As such, it no longer suffices to compare the outcomes of the treated and the untreated to estimate the average treatment effect as in RCTs. In studies that lack covariate balance between the treated and untreated, we must know all subjects' responses under treatment **and** under no treatment to estimate an unbiased average treatment effect. Since receiving treatment and not receiving treatment are mutually exclusive, there will always exist an unknown counterfactual response. To remedy the missing data problem, we must compare each treated subject to a *similar* untreated subject. The similarity of two subjects will be determined by their respective conditional probabilities of receiving treatment given a vector of covariates.

With sufficient pretreatment similarity between a treated and untreated subject, we can estimate an treatment effect. From there, we can take sample averages to obtain the average treatment effect, the average change in response resulting from treatment.

1.3 Balancing Score

As mentioned in Section 1.1, RCTs allow us to estimate the average treatment effect by comparing the outcomes of the treatment and control group. More specifically, $ATE = E(r \mid z = 1) - E(r \mid z = 0)$ where r denotes the subjects' response and z denotes a binary treatment indicator. In observational studies, however, individuals in the treated group often systematically differ from individuals in the untreated group. For instance, individuals who quit smoking likely have better access to healthcare than those who do not quit smoking. The systematic differences between those who quit smoking, the treated group, and those who do not quit smoking, the untreated group, motivate the use of propensity scores as a balancing mechanism.

As defined by Rosenbaum and Rubin [1983], a balancing score, $b(\mathbf{x})$, is a function of the observed covariates $\mathbf{x}$ such that the conditional distribution of $\mathbf{x}$ given $b(\mathbf{x})$ is the same for the treated ($z = 1$) and the untreated ($z = 0$). For example, individuals with the same balancing score, $b(\mathbf{x})$, will therefore have similar access to healthcare regardless of treatment group.

The propensity score, defined in the following section, is an example of a balancing score. Rosenbaum and Rubin [1983] show that, conditional on some assumptions detailed later, the difference in response means between the treated and untreated at each value of the propensity score provides an unbiased estimation of the average treatment effect at that propensity score value.

1.3.1 Propensity Score Definition

Propensity scores are balancing scores and provide a mechanism to estimate the average treatment effect. A propensity score is a subject's probability of receiving treatment conditional on a vector of observed pretreatment covariates, $\mathbf{x}_i$. Let $e_i(\mathbf{x})$ denote subject i's propensity score.

$$e(\mathbf{x}_i) = P(z_i = 1 \mid \mathbf{x}_i) \tag{1.2}$$

In order to achieve an unbiased estimate for the average treatment effect, a few assumptions must hold. First, the observed covariates, $\mathbf{x}$, must explain all of the pre-existing differences between the treated and untreated group; this is known as the **strong ignorability assumption**. In essence, the strong ignorability assumption says that all of potential confounding variables have been measured prior to treatment.

Additionally, an individual's outcome must not be affected by another individual's outcome; this is known as the **stable unit treatment value assumption**. Lastly, all individuals must have a propensity score strictly less than 1; no individual receives treatment with complete certainty.

Under the prior assumptions, Rosenbaum and Rubin [1983] show that, on average, the expected value of $\mathbf{x}$, conditional on $e(\mathbf{x})$, is the same regardless of treatment group. Mathematically, for a given covariate vector $\mathbf{x}$,

$$E(\mathbf{x} \mid e(\mathbf{x}), z = 1) = E(\mathbf{x} \mid e(\mathbf{x}), z = 0) \tag{1.3}$$

In other words, if a matched treated-untreated subject pair or subclass of subjects are homogeneous in $e(\mathbf{x})$, then the matched pair or subclass will have the same distribution in $\mathbf{x}$ [Rosenbaum and Rubin, 1984]. Rosenbaum and Rubin [1984] go on to show that, under strongly ignorable treatment assignment, observations with the same propensity score can act as controls for one another, and thus the expected difference in their responses equals the average treatment effect. Note, this result allows us to estimate the average treatment effect without ever knowing the counterfactual response!

Given the proposed theory, the question now is: how do we calculate and condition on the propensity scores? It turns out that propensity scores can be applied via matching, stratification, and weighting. In this paper, I will apply propensity scores using stratification, also known as sub-classification. Sub-classification is detailed in the following section.

1.4 Propensity Score Stratification

Stratification is a common method for controlling systematic differences between treated and untreated groups. The main idea is to divide the sample into "similar" groups based on the observed covariates; each group is called a 'strata'. Then, the researcher can compare the treated and untreated observations within a given strata to estimate a strata average treatment effect.

The propensity score is a scalar function of the p covariates and contains adequate information to successfully balance the covariates [Rosenbaum and Rubin, 1983]. Rosenbaum and Rubin [1984] claim that as few as 5 strata based on propensity scores are sufficient to remove 90% of the bias within the p covariates. So, while in an ideal world we would form a strata for each distinct propensity score, $e(\mathbf{x}_i)$, to achieve a perfect covariate balance within the treated and untreated subjects, in practice, this is impractical and often

unnecessary. The propensity scores are simply estimations, and thus it is impossible to create strata perfectly homogeneous in $e(\mathbf{x})$ that contain both treated and untreated subjects.

Instead of perfect homogeneity, strata are defined by intervals of propensity scores. Neuhäuser et al. [2018] study the optimal number of strata to estimate the average treatment effect. While Rosenbaum and Rubin [1984] suggest using 5 strata, Neuhäuser et al. [2018] suggest the researcher can obtain minimal bias and maximal power under the null with 10 strata. However, Neuhäuser et al. [2018] note that under the circumstance where there isn't a need for propensity score balancing, more highly refined strata yield substantially less power. However, both Rosenbaum and Rubin [1984] and Neuhäuser et al. [2018] estimate propensity scores using parametric techniques, and the optimal strata may differ depending of the technique. For the purposes of my simulations, I use 5 strata.

1.5 Evaluating the Propensity Score Usefulness

The goal in propensity score stratification is to eliminate the covariate imbalance between the treated and untreated groups. There exists covariate balance if the distribution of covariates is the same between the treated and untreated group [Otok et al., 2020].

As mentioned in Section 1.3.1, Rosenbaum and Rubin [1983] show that if we condition on the propensity score, then the distribution of observed covariates will be independent of treatment assignment. The end goal of achieving covariate balance motivates a method to evaluate the usefulness of our propensity scores. How well did the propensity scores balance the distribution of covariates between the treated and untreated groups? Only once we have reached covariate balance can we determine the average causal treatment effect.

1.5.1 Bias Reduction

We can determine how well a propensity score estimates the probability of treatment by looking at bias reduction. Prior to stratification, the initial

bias in $e(\mathbf{x})$ is B_I.

$$B_I = E(e(\mathbf{x}) \mid z = 1) - E(e(\mathbf{x}) \mid z = 0) \tag{1.4}$$

Let I_k be a range of propensity scores that defines subclass k where $k = 1, \ldots, K$. The resulting bias in $e(\mathbf{x})$ is B_S.

$$B_S = \sum_{k=1}^{K} [E(e(\mathbf{x}) \mid z = 1, e(\mathbf{x}) \in I_k) - E(e(\mathbf{x}) \mid z = 0, e(\mathbf{x}) \in I_k)] P(e(\mathbf{x}) \in I_k) \tag{1.5}$$

We can then measure the percent reduction in bias in $e(\mathbf{x})$ resulting from propensity score stratification.

$$\text{Percent Bias Reduction(PBM)} = (1 - \frac{B_S}{B_I}) \cdot 100\% \tag{1.6}$$

While a large percent bias reduction is often a sign of improvement, it does not indicate perfect covariate balance. In addition to bias reduction analysis, it is vital to check for covariate balance between the treated and untreated subjects for each covariate within each strata.

1.5.2 Pre-treatment Covariate Hypothesis Testing

Propensity score stratification is an attempt to create subclasses of observations that are "similar enough" in pre-treatment covariate distribution such that the only difference between observations within a strata is treatment group. If we can achieve this similarity condition, then we can estimate the average treatment effect. "Similar" is too vague for our purposes. As such, we quantify significant dissimilarity via a hypothesis test to check for significant differences between the treated and untreated individuals within a given strata for a given pre-treatment covariate. Within a strata, we hope that there are no statistically significant differences in average pre-treatment covariate values between treatment groups. To check, we follow the hypothesis tests outlined below.

Note: The following tests are performed within each strata (defined by a range of propensity scores) for each pre-treatment covariate. If we adequately estimate the propensity scores, there should be no significant differences between treatment groups within a strata for the pre-treatment covariates.

1. **Two Population t-test**

   If the covariate is continuous, one can check for covariate balance via a t-test. A t-test is a statistical technique to decide whether there is a significant difference in the mean of a variable between two groups. After propensity score subclassification, we do not want any significant differences between the means of the untreated and treated for a given covariate within each subclass.

   To test for covariate balance within each subclass, construct a hypothesis test for each covariate of the following form. First, let group 1 denote all treated individuals. Similarly let group 2 denote all untreated individuals. Let μ_1, μ_2 denote the respective population mean for group 1 and group 2 for a given pre-treatment covariate. Additionally, let $\bar{x}_1, \bar{x}_2$ denote the respective sample means and N_1, N_2 represent the size of the samples for group 1 and group 2. Finally, let s_1, s_2 denote the sample standard deviations.

   - **Hypotheses**:
     $H_0 : \mu_1 = \mu_2$ (Null Hypothesis)
     $H_1 : \mu_1 \neq \mu_2$ (Alternative Hypothesis)
   - **Statistic:**
     $T = \frac{\bar{x}_1 - \bar{x}_2}{s_p\sqrt{\frac{1}{N_1}+\frac{1}{N_2}}}$ where $s_p^2 = \frac{(N_1-1)s_1^2+(N_2-1)s_2^2}{N_1+N_2-2}$
     Note: The t-statistic is distributed according to a t-distribution with $(N_1 + N_2 - 2)$ degrees of freedom. Mathematically,
     $$T \sim t_{N_1+N_2-2}$$
   - **Test:**
     $\delta = \{\text{Reject Null Hypothesis if } |T| > t_{1-\alpha/2}\}$
     Note: Rejecting the null hypothesis suggests that there is a statistically significant difference in the population means.

1.5.3 Checking for Covariate Balance After Stratification Example

In this example, I apply the t-test outlined above on a real-world problem. More specifically, the data I will use comes from the r package 'cidata' and contains 1,566 observations.

Our casual question of interest is: **Does quitting smoking cause a change in body weight?**

It follows that change in body weight, denoted wt82_71, is the response variable. Quitting smoking, denoted 'qsmk', will serve as the treatment indicator variable; 1 denotes the subject quit smoking. Lastly, subjects' age, income, education, and smoking intensity serve as pre-treatment covariates. To better grasp the data structure, consider a sample of 10 observations from the data set.

qsmk <dbl>	age <dbl>	income <dbl>	education <fctr>	smokeintensity <dbl>	wt82_71 <dbl>
1	43	19	3	30	15.87225712
0	34	18	3	40	-1.81900392
0	54	16	1	20	0.56215469
0	51	*NA*	3	10	0.11013065
1	71	17	1	40	-18.94095375
0	47	20	3	10	5.32963261
0	56	19	3	20	0.45070290
1	48	18	3	2	8.27681735
0	47	19	3	20	6.80691978
0	47	18	2	12	3.85963261

11–20 of 1,566 rows Previous 1 2 3 4 5 6 … 100 Next

Figure 1.1: 'wt82_71'(response variable), 'qsmk'(treatment variable)

While the end goal is to estimate the average causal effect of quitting smoking on body weight, we do not have a randomized controlled trial. As such, there likely exist systematic differences between those who quit smoking (i.e., received treatment) and those who did not (i.e., did not receive treatment). For instance, it is possible that those who quit smoking had lower initial smoking intensities compared to those who did not quit smoking. To visually investigate this claim, consider the plot below:

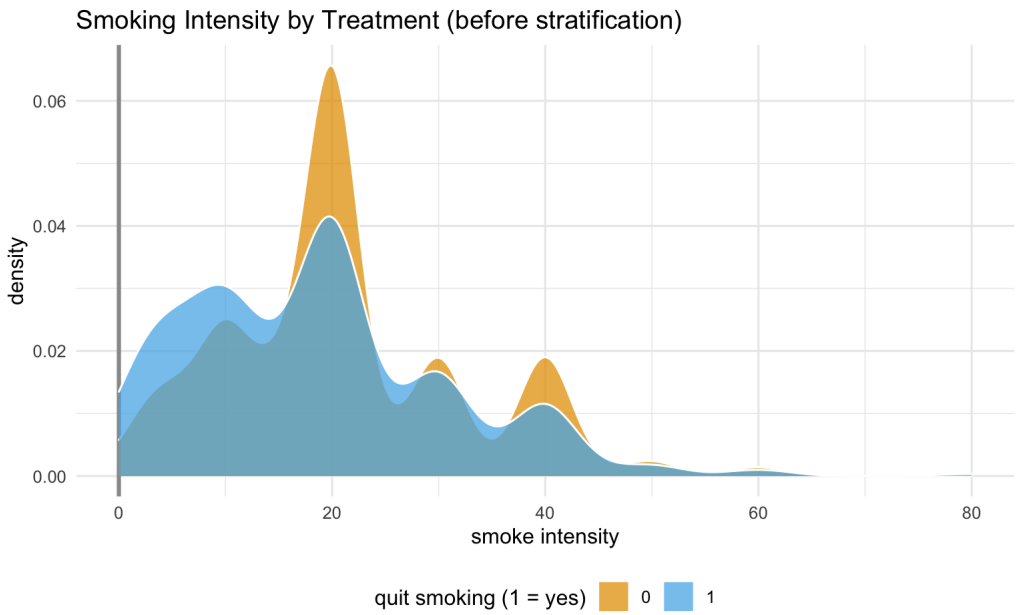

Figure 1.2: Pre-treatment smoking intensity levels segmented by those who quit smoking and those who did not quit smoking. Note, the the mean initial smoking intensity for those who quit smoking is lower than for those who did not quit smoking.

Looking at the graph above, it seems as though the subjects who quit smoking had initial smoking intensities lower than those who did not quit smoking. To put numbers to our intuition, I run a t-test to see whether there exists a statistically significant difference in population means for smoking intensity between those who quit and those who did not quit. The output of the t-test (computed in r) confirms our intuition. The sample smoking intensity mean for those who did not quit smoking is 21.19 compared to 18.60 for those who did quit smoking. Using the sample means and standard errors, we obtain a t-statistic of 3.68. The p-value associated with the t-statistic is 0.00025 which is statistically significant at the $\alpha = 0.05$ level. Thus, we conclude that there is a statistically significant difference in the population level smoking intensity levels for those who quit smoking and those who did not quit.

If we were to compute the average treatment effect by taking the

difference of the weighted mean responses for the treated and untreated group, we essentially would be comparing oranges to apples. More specifically, there are significant pre-treatment differences between those who quit smoking and those who did not, and thus we cannot isolate the treatment effect. In order to achieve an 'apples to apples' comparison, we use propensity score stratification to create groups of individuals with similar initial probabilities of quitting smoking according to their pre-treatment characteristics.

For simplicity, I estimated all propensity scores (the probability of receiving treatment conditional on pre-treatment covariates) using logistic regression. I then segmented the data into 5 strata defined by ranges of propensity scores. Consider the 3rd strata. The graph below compares the smoking intensity levels for those who quit smoking and those who did not quit smoking **within the 3rd strata**.

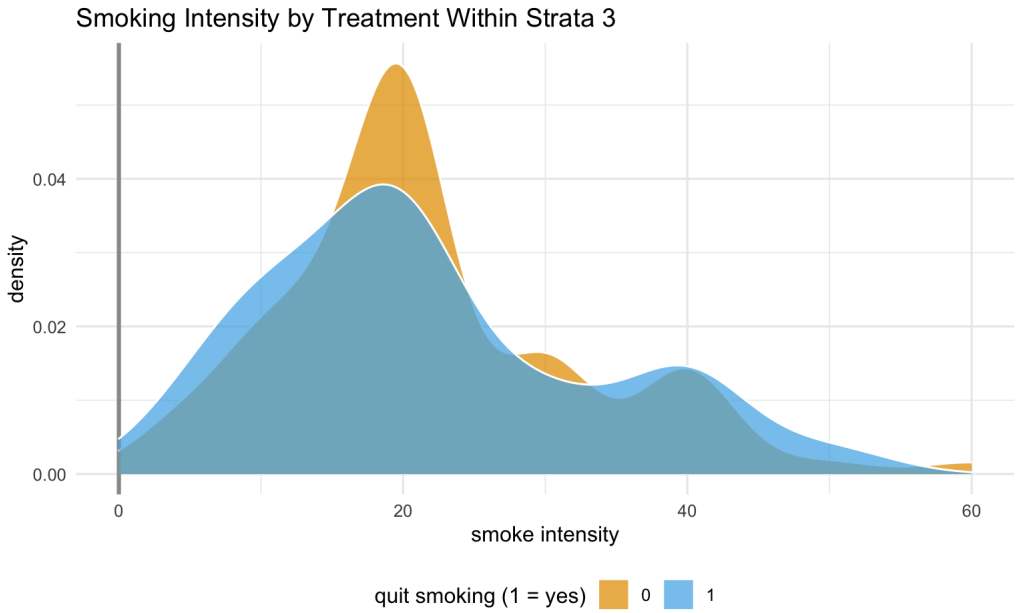

Figure 1.3: Pre-treatment smoking intensity levels within the 3rd strata segmented by those who quit and those who did not quit smoking. Note, the mean initial smoking intensities for those who quit smoking is similar to those who did not quit smoking.

Upon initial glance, there seems to be greater balance between the pre-treatment smoking intensity levels for those who quit smoking vs. those who did not quit. The goal of propensity score stratification is to create groups of subjects that are similar in all pre-treatment covariates. As such, we hope that there is not a significant difference within the 3rd strata in pre-treatment smoking intensities between those who quit smoking and those who did not quit smoking. To test, I perform a two-sided t-test.

The results of the t-test (performed in r) are as follows. Within strata 3, the sample smoking intensity mean for those who quit smoking is 21.81 and 22.05 for those who did not quit smoking. This yield a t-statistic of 0.16 and a corresponding p-value of 0.87; the p-value is insignificant at the $\alpha = 0.5$ significance level. Thus, we conclude that there is not a statistically significant difference between the smoking intensities for those who quit smoking and those who did not quit smoking within strata 3. Yay!

To fully evaluate the propensity score stratification effectiveness, we would repeat this process for each pre-treatment covariate within each strata. Successful propensity score stratification will yield insignificant pre-treatment covariate differences between those who quit smoking and those who did not quit smoking within each strata.

2. **Test of Proportions**

   If the pre-treatment covariate is categorical and the response is categorical, we use the difference test of proportions, also know as the z-test. Here, let p_1, p_2 denote the population proportions of successes in group 1 and group 2 respectively. Given a pre-treatment covariate with 2 levels, $\hat{p}_1$ denotes the sample proportion of successes in group 1 over the group 1 sample size. Similarly, $\hat{p}_2$ denotes the sample proportion of successes in group 2 over the group 2 sample size.

   - **Hypotheses:**
     $H_0 : p_1 = p_2$ (Null Hypothesis)
     $H_1 : p_1 \neq p_2$ (Alternative Hypothesis)
   - **Statistic:**
     $$Z = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\frac{\hat{p}_1(1-\hat{p}_1)}{N_1} + \frac{\hat{p}_2(1-\hat{p}_2)}{N_2}}}$$

- **Test:**

  $\delta = \{\text{Reject Null Hypothesis if } |Z| > z_{1-\alpha/2}\}$

  Note: Rejecting the null hypothesis suggests that there is a statistically significant difference in the population proportions between group 1 and group 2 for a given covariate.

1.6 Estimating the Average Treatment Effect

Recall, in randomized controlled trials we estimate an unbiased average treatment effect simply by comparing the average response values between treatment groups.

$$ATE = E(r \mid z = 1) - E(r \mid z = 0)$$

However, in observational studies, the treated an untreated groups will often systematically differ in observed pre-treatment covariates. As we saw in Section 1.4, to achieve a similar covariate balance as in RCTs, we stratify the subjects according to their estimated propensity scores. Once divided into strata and checked for covariate balance, we compare response values within each strata to achieve an unbiased estimate for the ATE.

Let $\widehat{ATE}$ denote the estimated ATE. The steps to estimate the unbiased $\widehat{ATE}$ are as follows.

1. Divide observations into strata defined by a range of propensity scores (see Chapter 2, 3, 4). Let each strata be denoted by I_k where $k = 1 \dots K$.

2. Determine the average outcome of the treatment and control groups within each subclass.

   $N_t = \sum_{k=1}^{K} N_{tk}$ Total number of treated individuals across all strata.

   $N_c = \sum_{k=1}^{K} N_{ck}$ Total number of untreated individuals across all strata.

   $\bar{r}_{tk} = \sum_{i=1}^{N_{tk}} \frac{r_{itk}}{N_{tk}}$ Average response of the treated individuals in strata k.

   $\bar{r}_{ck} = \sum_{i=1}^{N_{ck}} \frac{r_{ick}}{N_{ck}}$ Average response of the untreated individuals in strata k.

3. Calculate the estimated ATE, $\widehat{ATE}$,

$$\widehat{ATE} = \sum_{k=1}^{K} \frac{N_{tk} + N_{ck}}{N_t + N_c}(\bar{r}_{tk} - \bar{r}_{ck}) \tag{1.7}$$

Summation of the weighted average treatment effects over all strata.

4. Calculate the standard error, $SE(\widehat{ATE})$,

$$\text{SE}(\widehat{ATE}) = \sqrt{\sum_{k=1}^{K}(\frac{N_{tk} + N_{ck}}{N_t + N_c})^2(\frac{s_{tk}^2}{N_{tk}} + \frac{s_{ck}^2}{N_{ck}})} \tag{1.8}$$

Once we calculate $\widehat{ATE}$ and the standard error of $\widehat{ATE}$, we can perform significance testing on the the true average treatment effect parameter, denoted ATE.

- Hypotheses & Significance:

  H_0 : ATE $= 0$ There is no treatment effect.

  H_1 : ATE $\neq 0$ There is a treatment effect.

  $\alpha = 0.05$ This will control our type I error.

- Statistic:

  $Z = \frac{\widehat{ATE} - 0}{\text{SE}(\widehat{ATE})}$

- Test:

  $\delta = \{\text{Reject null if } |Z| > Z_{0.975}\}$

In this case, rejecting the null hypothesis and accepting the alternative hypothesis suggests that the parameter average treatment effect is significantly different from zero.

Chapter 2

Logistic Regression for Propensity Score Estimation

Propensity score stratification (detailed in Section 1.4) provides an excellent method to estimate the unconfounded average treatment effect. To successfully construct the strata, we must determine each subject's propensity score, a.k.a. each subject's conditional probability of receiving treatment given a vector of pre-treatment covariates. Logistic regression provides one method to estimate propensity scores. It is important to recognize that our goal in Chapters 2, 3, and 4 is to successfully predict whether a subject will receive treatment. Notably, we are **not** predicting the response variable of interest. If we can successfully predict each subject's probability of receiving treatment, then we essentially, within each strata, have access to 'apples to apples' comparisons between treatment groups.

In this chapter, I detail the theory of logistic regression, the application to propensity score estimation, and the benefits and drawbacks of using logistic regression.

2.1 Generalized Linear Models

The logistic regression model exists within the family of generalized linear models (shortened as GLM). Aside from logistic regression, some examples of GLMs include Poisson regression, log-linear regression, and linear regression. GLMs are characterized by three similarities. In all GLMs, there exists the following components:

1. Random component: a probability distribution describing the response variable, denoted Y. The random component is often described by distributions from the exponential family (Normal, Poisson, Binomial, etc).

2. Linear component: The explanatory variables and the linear combination of such variables. i.e. $\beta_0 + \beta_1 X_1 + \ldots \beta_n X_p$

3. Link function (η): The functional relationship between the expected value of the response variable and the linear explanatory predictors.

GLMs require the observations in the data to be independent. While a GLM does not necessarily assume a linear relationship between response and explanatory variables, it does assume a linear relationship between the link function and the explanatory variables. Mathematically, the link function must have the following form: $\eta(E(Y \mid \mathbf{x})) = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_p$. Lastly, the errors are assumed homoskedastic (i.e. errors are independent of the explanatory variables). To estimate the coefficients $(\beta_1, \ldots, \beta_p)$, GLMs use maximum likelihood estimation (MLE).

2.2 Logistic Regression Theory

2.2.1 First, Some Notation

Within an observational study of N subjects, consider subject i, where $i = 1 \ldots N$. Let Z_i denote a random variable indicating subject i's treatment status. Z_i can take on two values: $z_i = 1$ (denotes treatment) with probability $e(\mathbf{x}_i)$ or $z_i = 0$ (denotes no treatment) with probability $1 - e(\mathbf{x}_i)$. Recall, $e(\mathbf{x}_i)$ represents subject i's propensity score. More specifically, $e(\mathbf{x}_i) = \mathrm{P}(z_i = 1 \mid \mathbf{x}_i)$ where $\mathbf{x}_i$ is a vector of observed pre-treatment covariates for subject i. To model the binary outcome (treatment vs. no treatment) and the uncertainty of each event, we can use a Bernoulli distribution:

$$Z_i \sim \text{Bernoulli}(e(\mathbf{x}_i)) \text{ where } i = 1, \ldots, N \tag{2.1}$$

2.2.2 Logistic Regression as a GLM

Logistic regression falls within the broader Generalized Linear Model class. Binary logistic regression models the probability of the occurrence of a binary

event given categorical or numerical explanatory variables. Additionally, logistic regression allows us to estimate the effect of a series of explanatory variables on the response. Logistic regression maps to the general GLM criteria as follows:

1. Random component: $Z_i \sim \text{Bernoulli}(e(\mathbf{x}_i))$ where Z_i is the treatment indicator variable for subject i.

2. Linear component: Let $\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$ denote the linear combination of explanatory variables. Note that explanatory variables can undergo transformations.

3. Link function: $\eta(\mathbf{x}_i) = \text{logit}(e(\mathbf{x}_i)) = \ln\left(\frac{e(\mathbf{x}_i)}{1-e(\mathbf{x}_i)}\right)$

From section 2.1, we know the link function is defined such that it has a linear relationship with the series of explanatory variables. Thus,

$$\text{logit}(e(\mathbf{x}_i)) = \ln\left(\frac{e(\mathbf{x}_i)}{1-e(\mathbf{x}_i)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p \tag{2.2}$$

From this relationship, we can solve for $e(\mathbf{x}_i)$ thereby deriving the estimated probability of treatment. Let $\alpha = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$ for simplicity.

$$\begin{aligned}
\ln\left(\frac{e(\mathbf{x}_i)}{1-e(\mathbf{x}_i)}\right) &= \alpha && \text{(Given by equation 2.2)} \\
\frac{e(\mathbf{x}_i)}{1-e(\mathbf{x}_i)} &= e^{\alpha} && \text{(Exponentiate both sides)} \\
e(\mathbf{x}_i) &= e^{\alpha} \cdot (1 - e(\mathbf{x}_i)) && \text{(Cross Multiply)} \\
e(\mathbf{x}_i) &= e^{\alpha} - e(\mathbf{x}_i) \cdot e^{\alpha} && \text{(Distribute } e^{\alpha}\text{)} \\
e(\mathbf{x}_i) + e(\mathbf{x}_i) \cdot e^{\alpha} &= e^{\alpha} && \text{(Isolate } e(\mathbf{x}_i)\text{)} \\
\hat{e(\mathbf{x}_i)} &= \frac{e^{\alpha}}{1+e^{\alpha}} = \text{logit}^{-1}(\alpha) && \text{(Divide by } (1+e^{\alpha})\text{)}
\end{aligned}$$

First, note that $\hat{e(\mathbf{x}_i)} \in [0, 1]$ and thus nicely satisfies the bounds of a probability. From the estimated regression equation, we estimate all β values through maximum likelihood estimation (MLE). In maximum likelihood estimation, we maximize the conditional likelihood of observing the data given a

distribution and parameter. For the purposes of this thesis, all optimization will be done in R.

In linear regression, our response variable is estimated by a linear combination of the explanatory variables. In contrast, in logistic regression, the linear combination of the explanatory variables estimates the **ln(odds)** of treatment given a set of explanatory variables.

The **odds** of an event is defined as ratio of the probability of success to the probability of failure. Probability and odds are defined as follows:

$$\text{Probability} = \frac{\#\text{ successes}}{\#\text{ possible outcomes}} \tag{2.3}$$

$$\text{Odds} = \frac{\#\text{ successes}}{\#\text{ failures}} \tag{2.4}$$

To better understand the concept of odds, consider the following example.

Example 2.1. Probability of Drawing a Diamond
You are asked to draw one card from a standard 52 playing card deck. What is the probability that your chosen card is a diamond?

$$P(\text{diamond}) = \frac{\#\text{ Diamonds in Deck}}{\#\text{ Cards in Deck}} = \frac{13}{52} = 0.25$$

The **probability** of drawing a diamond from a 52 card deck is 25%.

Odds provide an alternative measure of likelihood for a given outcome.

Example 2.2. Odds of Drawing a Diamond From a 52 Card Deck

$$\text{Odds(diamond)} = \frac{\#\text{ diamonds}}{\#\text{ non-diamonds}} = \frac{13}{39} = 0.33\overline{3}$$

The **odds** of drawing a diamond from a 52 card deck is $0.33\overline{3}$.

Note, in logistic regression, we are frequently more concerned with odds rather than probability due to simpler interpretation of the coefficients. More specifically, researchers are often more interested in odds ratios rather differences in odds due to the simplicity of interpretation.

In linear regression, $E(Y \mid X_1) = \beta_0 + \beta_1 X_1$, the interpretation of β_1 is simple; for a unit increase in X_1, $E(Y \mid X_1)$ increases by β_1. We can see this clearly by subtracting the two equations: $\beta_0 + \beta_1(x) - \beta_0 - \beta_1(x+1) = \beta_1$. However, in logistic regression, the comparison of models happens in ln(odds) units. Consider the following.

Let $Z_i \sim \text{Bernoulli}(e(\mathbf{x}_i))$ where $z_i = 1$ means subject i received treatment and $z_i = 0$ means the subject did not receive treatment. Let X denote an explanatory variable. Note, $e(\mathbf{x}_i)$ denotes subject i's probability of receiving treatment (i.e., the propensity score).

Using the logit link function, we obtain the following relationship between explanatory variables and response:

$$\ln\left(\frac{e(\mathbf{x}_i)}{1 - e(\mathbf{x}_i)}\right) = \beta_0 + \beta_1 X$$

Often researchers are interested in the effect of a unit change in the explanatory variable on the response. Let $X = x$ for individual 1, and let $X = x + 1$ for individual 2. More formally,

$$\ln(\text{odds}[X = x]) = \beta_0 + \beta_1(x)$$

$$\ln(\text{odds}[X = x + 1]) = \beta_0 + \beta_1(x + 1)$$

We can exponentiate both sides to isolate the odds.

$$\text{odds}[X = x] = e^{\beta_0 + \beta_1(x)}$$

$$\text{odds}[X = x + 1] = e^{\beta_0 + \beta_1(x+1)}$$

The **odds ratio**(OR) for a unit change in X is:

$$\text{OR} = \frac{\text{odds}[X = x + 1]}{\text{odds}[X = x]} = \frac{e^{\beta_0 + \beta_1(x+1)}}{e^{\beta_0 + \beta_1(x)}} = e^{\beta_1} \tag{2.5}$$

The odds ratio does not depend on X! This means that *regardless* of the initial X-value, the odds ratio for a unit change in the explanatory

variable will be constant, e^{β_1}. The odds ratio describes the change in odds due to a unit increase in the explanatory variable.

Since the OR is constant regardless of the initial value of the explanatory variable and the OR has a nice interpretation, it is widely used and discussed in logistic regression. The constant characteristic is not true for other statistics we may want to calculate. For instance, the difference in odds is **not** constant, but rather highly dependent on the initial value of X.

$$\text{odds}[X = x+1] - \text{odds}[X = x] = e^{\beta_0+\beta_1(x+1)} - e^{\beta_0+\beta_1(x)} = e^{\beta_0}(e^{\beta_1(x+1)} - e^{\beta_1(x)}) \tag{2.6}$$

2.3 Pros and Cons of Logistic Regression

Logistic regression is overwhelmingly the most common technique used for propensity score estimation. Westreich et al. [2010] detail the advantages and disadvantages in using logistic regression for propensity score estimation. First, the logistic model produces likelihoods constrained to [0,1] which is necessary in probability estimation. Note, propensity score estimation is probability estimation, so the constraint is good! Moreover, logistic regression is fairly intellectually accessible without exposure to advanced mathematics. Due to the extensive research and studies surrounding logistic regression, it is conveniently implemented within most statistical software including R, STATA, and Python. These advantages are convincing in using logistic regression for propensity score estimation. However, logistic regression has several pitfalls which are sparsely acknowledged in existing literature.

First, logistic regression assumes a linear relationship between the logit link function and predictor variables. Westreich et al. [2010] found that over 90% of studies using logistic regression to estimate propensity scores consistently do not assess this assumption of linearity. Additionally, Westreich et al. [2010] claim researchers rarely used interaction terms within their models; this oversight is highly problematic as logistic regression is a parametric statistical technique which rests heavily on accurate predefined relationships between the explanatory variables and treatment level. Since non-parametric machine learning models often require fewer assumptions and look to the data to retroactively define interactions, we will consider these models in subsequent chapters.

Chapter 3

Tree-based methods for Propensity Score Estimation

In estimating propensity scores, we estimate each subject's probability of receiving treatment conditional on a set of pre-treatment covariates. As we saw in Chapter 2, logistic regression, the most commonly used method for propensity score estimation, and its ability to predict treatment with high accuracy rests entirely on a researcher's ability to correctly specify the functional form of covariates and the presence or absence of interactions between covariates. Without sufficient domain and mathematical knowledge, mispecifications of covariate relationships are common.

In this chapter, to avoid the need to pre-specify the functional form of the model, I motivate the use of the non-parametric tree-based techniques for propensity score estimation. More specifically, I begin with an individual classification tree and then further explore ensemble methods such as bagging and boosting.

3.1 Motivation for CART Methods

As described in the end of Chapter 2, parametric models rely heavily on correctly specified functional forms of the covariates and the presence or absence of interactions. Should the researcher incorrectly specify the relationship between the covariates, propensity score stratification will not yield an unbiased estimate of the average treatment effect [de Vries et al., 2018]. Thus, many have suggested the use of machine learning techniques, including tree-based

methods, to eliminate the need to satisfy the previously stated assumptions of covariate functional form and interactions.

As I traverse through alternative methods for propensity score estimation, it is important to remember that the goal is to model the probability of treatment for each subject. More importantly, we are **not** trying to predict the response of interest in the study. Consider again the example of whether quitting smoking causes an increase in body weight. In propensity score estimation, we are trying to model the probability that a subject quits smoking, not the change in body weight. Tree-based methods provide one way to model the probability that a subject quits smoking.

3.2 Classification and Regression Trees (CART)

CART is a supervised learning technique that finds an optimal set of rules to partition the data based on input values, a.k.a. covariates. A single tree consists of a root node, branches, and leaf/terminal nodes. At each node, there is an *if/else* statement which defines the binary splits of the data; **leaf nodes** results from this binary split. Once the model reaches an 'optimal' point, the most recent leaf nodes are deemed **terminal nodes**.

As mentioned above, each non-terminal leaf node further splits based on some criteria thereby creating additional leaf nodes. The goal in data partitioning is to minimize the heterogeneity amongst the response values. In the case of propensity score estimation, the response of interest is treatment exposure. A common measure of heterogeneity is called the Gini impurity, denoted G. Given C total classes, let $p(i)$ denote the probability of choosing a point belonging to class i. The Gini impurity calculation occurs **within each node** and is calculated as follows:

$$G = \sum_{i=1}^{C} p(i)(1 - p(i)) \tag{3.1}$$

Gini impurity is used to determine which variable to split on to minimize heterogeneity. The variable and split-point that results in the maximal decrease in Gini impurity is chosen. After deciding the split criteria, one must specify the criteria for terminating the recursive partitioning process. A common strategy to decide whether a node will further split or become a terminal node is based on the existing number of observations within the

node. Should the node have fewer than the minimum number of observations to split, then that node will become a terminal node and tree development will cease. Additionally, many researchers set a maximum tree depth to avoid overfitting; if the maximum depth is reached, the tree will stop developing. Lastly, once there is a developed classification tree with many terminal nodes, the researcher must decide the criteria upon which to classify the terminal nodes. Most often, this is decided via 'majority rules'; the terminal node is labeled as the class to which the majority of the observations belong.

Example 3.1. Classification Tree Example

Consider again the casual questions posed in Subsection 1.5.3. As a refresher, we are wondering whether quitting smoking causes a change in body weight. In the data set sourced from the 'cidata' package in r, 'qsmk' denotes the treatment variable (1 = subject quit smoking) and 'wt73_82' denotes the response variable. We consider 4 other explanatory variables: 'age', 'income', 'education', and 'smokeintensity'.

Recall, this chapter is all about propensity score estimation. As such, we are trying to estimate each subject's probability of quitting smoking given their pre-treatment age, income, education, and smoking intensity. To do so, we will use a single classification tree.

In the classification tree, the **root node** includes all observations in the data set (N = 1,566). Then, the algorithm searches for the variable and split point combination which results in the greatest decrease in Gini impurity. In this case, the first binary split is defined by "smokinghistory ≥ 15". The data is partitioned accordingly, thereby creating 2 **leaf nodes**. Now, the algorithm looks within each leaf node and determines the optimal split based on largest decrease in Gini impurity. In this example, the leaf node where "smokinghistory ≥ 15" becomes a terminal node since there does not exist a further partition that results in a significant decrease in Gini impurity. However, the leaf node where "smokinghistory < 15" is further partitioned. More specifically, the greatest decrease in Gini impurity results from the split "age < 52"; this split creates two additional leaf nodes. The partitioning process continues until the leaf nodes reach the stopping criterion, thereby becoming **terminal nodes**.

Once the classification tree model is built out entirely, we construct our propensity score predictions. Consider a single terminal node. The propensity score for the observations within that terminal node is calculated via

the proportion of treated observations within the node. For instance, if a terminal node has 10 subjects total and 8 of the 10 subjects quit smoking, then the propensity score for all 10 subjects within the terminal node would be $\frac{8}{10} = 0.80$.

The following classification tree diagram reflects the data partitions mentioned above. Note, there are two numbers within each leaf; the top number represents the probability of quitting smoking, and the bottom number represents the percentage of total subjects who fall into that class.

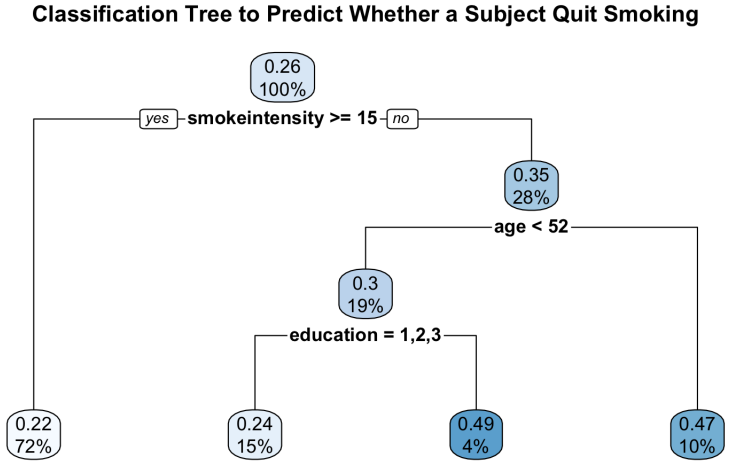

Figure 3.1: Example of a single classification tree used to estimate propensity scores. In this case, the classification tree seeks to predict whether a subject quits smoking given a vector of pre-treatment covariates.

There are a few things to note about Gini impurity and classification trees. In order for a classification tree to split, there must be an *immediate* decrease in Gini impurity. While many choices of splits will decrease Gini impurity, we always choose the split that *maximally* decreases Gini impurity. This property leads to what is commonly called a **greedy algorithm**; at every split, the tree chooses the split which is best at the current moment without considering future paths. The greediness of the algorithm can lead to sub-optimal models.

In addition to being a greedy algorithm, classification trees are highly sensitive to a few new training observations and thus are called high-variance estimators. Put more simply, if we randomly split the training data into

two parts and fit a tree to each part, the two resulting trees would be very different. The high variability of the estimator can cause a drastic decrease in the predictive ability [de Vries et al., 2018]. For this reason, I turn to ensemble tree-based methods.

3.3 Classification Tree Ensemble Learners

Single classification trees can be used to form ensemble learners. An **ensemble learner** is a learner which uses many simple learners to improve predictive performance. Here, I will consider two ensemble learners: bootstrapped aggregated (bagged) CART and boosted CART.

3.3.1 Bagged CART

Bootstrapped aggregated (bagged) CART was first introduced by Leo Breiman in 1996. Bagged CART is fairly straightforward: fit a classification tree to many independent bootstrapped training sets and aggregate the trees to form a final prediction. This aggregation technique aims to decrease the variance of a predictor and in turn maximize predictive capabilities [Breiman, 1995]. To build this learner, there are three main things to understand:

1. Classification and Regression Trees (detailed in section 3.2)
2. Bootstrapping
3. Aggregating estimates

In this section, I focus on bootstrapping and aggregation. The bootstrap is a statistical re-sampling technique used to construct random samples of the existing data. In real life, we only ever have one data set to train the model on. Bootstrapping is a method to construct additional training data sets without gathering any additional data. In bootstrapping, a researcher repeatedly samples from the original training data set with replacement; each sample is considered an observation in the new bootstrapped, training data set. I will return to the **why** part of bootstrapping in a bit.

Constructing a single bootstrapped classification tree consists of two steps. First, construct a random sample with replacement of size N from existing training data set (also of size N). Second, build a classification tree using the bootstrapped sample as the training data set. Then, repeat steps 1 and

2 until you have reached the chosen number of individual classification trees. After this process, we end up with many separate bootstrapped classification trees.

Now, we turn our focus toward the bootstrap aggregation step, also known as bagging. As mentioned at the end of section 3.2, single classification trees are high-variance estimators. A method to counter this high-variance property is bagging. In bagging, the idea is to combine many weak learners, such as classification trees, fitted on bootstrapped training sets and predict a new observation's class via a majority rule.

The method of bagging is known to reduce variance in the estimator as shown via a typical result in statistical theory. As eloquently noted by Hastie et al. [2001] and translated into our notation, we let $\hat{e}_1(\mathbf{x}_i), \ldots, \hat{e}_B(\mathbf{x}_i)$ denote B different propensity score estimations each with variance σ^2. It follows that the average of the predictions, $\hat{e}_{avg}(\mathbf{x}_i)$, is distributed with variance σ^2/B, which is notably smaller than σ^2.

As such, in bagged CART, we take B independent training data sets and build a regression tree for each data set. After we have built B different regression trees, we use each regression tree to calculate B different propensity scores for each subject i ($i = 1, \ldots, N$). Finally, for each subject i, we average the B propensity score estimations. In mathematical notation, subject i's propensity score estimation is obtained as follows:

$$\hat{e}_{avg}(\mathbf{x}_i) = \frac{1}{B} \sum_{b=1}^{B} \hat{e}_b(\mathbf{x}_i) \tag{3.2}$$

However, in real life, **we only have a single data set**; this is the glory of the bootstrap! The bootstrap will allow us to construct the B (non-independent) training sets via repeated sampling with replacement. Using the bth bootstrapped sample, we can estimate $\hat{e}^*_{avg}(\mathbf{x}_i)$ via a bagged learner of the following form. Note, the only difference between equation 3.2 and equation 3.3 is a slight change in notation to specify the use of bootstrapped data sets via a *.

$$\hat{e}^*_{avg}(\mathbf{x}_i) = \frac{1}{B} \sum_{b=1}^{B} \hat{e}^*_b(\mathbf{x}_i) \tag{3.3}$$

Consider the following diagram. First, the researcher constructs B bootstrapped samples. Note that the samples are constructed via sampling with

replacement and thus the bootstrapped samples are each different from the original sample. Additionally, each bootstrapped sample is the same size as the original sample, 13 observations, however with varying color proportions. Next, the researcher fits a decision tree to each bootstrapped sample. Final predictions for a test data set are made via averaging the individual model predictions. In this case, an individual model's prediction in the proportion of red observations to total observations at a given terminal node; thus, you are averaging proportions to obtain a final proportion. If the final averaged proportion of red observations to total observation is greater than 50%, then we classify the observation as "red".

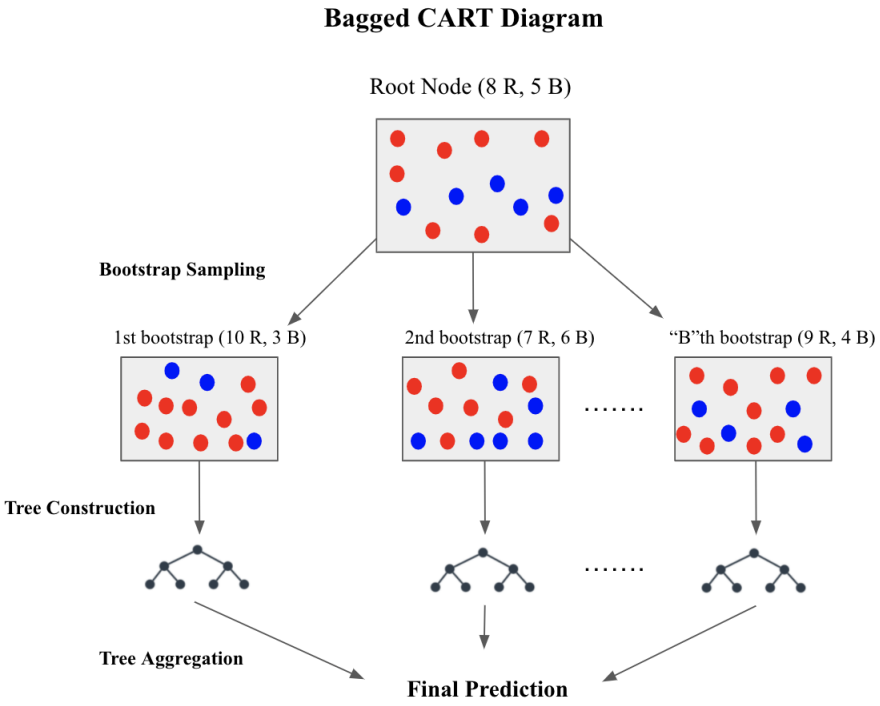

Figure 3.2: Bagged CART diagram. From the original sample, we construct B bootstrapped samples. For each bootstrapped sample, we build a decision tree. To obtain a final model prediction, we average over the individual tree predictions. For classification, we choose a cutoff point (typically 0.5) and predict the observation as "red" if the averaged prediction is greater than 0.5, blue otherwise.

In the case of propensity score estimation, we are not concerned with final class predictions, but rather the **probability** that an observation belongs to the treated class. Thus, averaging the proportions of treated individuals out of total individuals within the predicted class of each tree will suffice for propensity score estimation.

Random Forests

As Hastie et al. [2001] describe, random forests offer a slight improvement over bagged CART by way of tree de-correlation. More specifically, each split within each decision tree considers a random subset of explanatory variables. Other than this slight tweak, random forest construction parallels that of bagged CART. Generally, a random forest determines each split within each tree considering only $\sqrt{p}$ explanatory variables, where p is the total number of explanatory variables available. This means that each split is built using fewer than half of the available predictors. Using fewer predictor variables might seem counter-intuitive for the goal of maximizing predictive capabilities, however it can drastically reduce the variance of the estimator.

Consider a situation in which there is a single strong predictor and a handful of moderate predictors. Using a bagged CART model, we build 100 regression trees. Likely, most if not all of the 100 trees will use the strong predictor to determine the initial split; this results in 100 decision trees with very similar construction and thus highly correlated predictions. In statistics, the reduction in variance resulting from averaging correlated quantities is substantially less than the variance reduction from averaging uncorrelated quantities. Thus, to achieve minimal variance across predictions, we want to average uncorrelated or less correlated quantities; a random forest allows us to do this because the strong predictor will not be the first split in all the trees since each tree considers fewer than half the variables [Breiman, 2001].

In the diagram below, we can see that the process of tree construction is identical to that of bagged CART, **except** the selection of predictors on which each split is made. Prior to model construction, a researcher must specify a number of predictor variables to be used to build each tree, typically this is $\sqrt{p}$ where p is the total number of predictor variables. Then, in determining a best split within each tree, we randomly select a different subset of $\sqrt{p}$ variables. This process can drastically reduce the variance in predictions, especially when there are many correlated predictor variables [Breiman, 2001].

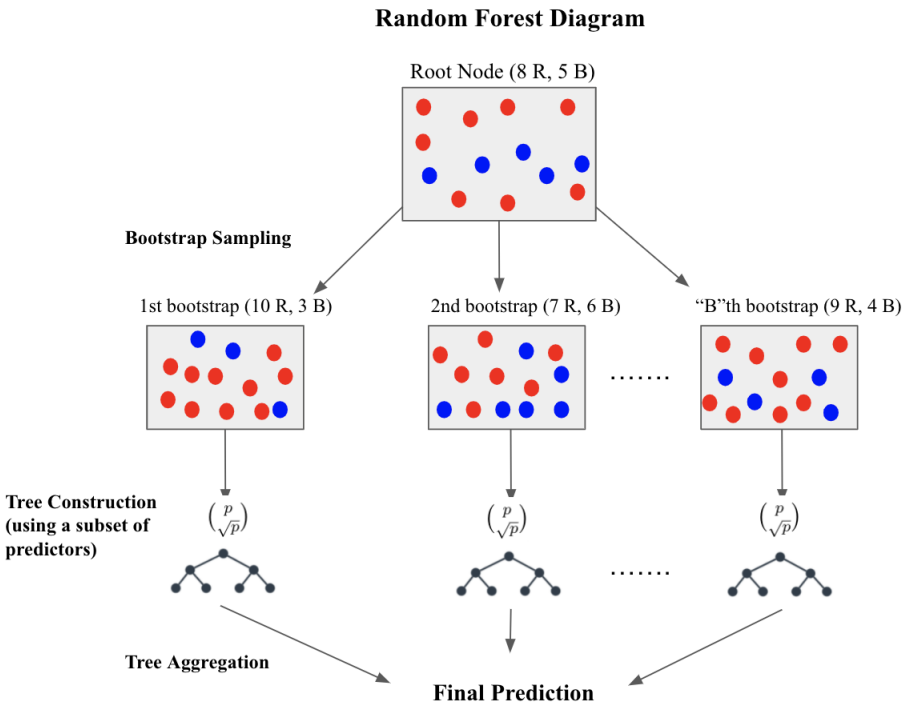

Figure 3.3: Random forest diagram. First, construct B bootstrapped samples from the original sample. Next, for each bootstrapped sample, randomly choose a different $\sqrt{p}$ set of predictor variables (this differentiates a random forest from bagged CART). Build the decision tree for each bootstrapped sample and obtain a final prediction by averaging the B individual predictions.

3.3.2 Boosted CART

In Section 3.3.1, I detail bagged CART and random forests as ensemble learners that build many trees in parallel to obtain a final prediction. Boosted CART is an alternative method which builds many shallow tree-based weak learners in a *sequential* fashion where the following tree is built using information from the previous. The boosted CART model I will consider is called the Generalized Boosting Model (GBM).

The Generalized Boosting Model is an algorithm that sequentially com-

bines simple regression trees in pursuit of a model with high predictive capabilities. Note, in our case, we want a model that can accurately predict a subject conditional probability of receiving treatment, the propensity score.

The first thing to note is that the GBM algorithm models the ln(odds of treatment), rather than the probability of treatment directly; in Chapter 2, we saw logistic regression also models ln(odds of treatment)! Let $g(\mathbf{x})$ denote the GBM model. Mathematically, the output of the model is the following:

$$\hat{g}(\mathbf{x}_i) = \ln\left(\frac{\hat{e}(\mathbf{x}_i)}{1-\hat{e}(\mathbf{x}_i)}\right) \text{ where } \hat{e}(\mathbf{x}_i) \text{ denotes subject i's estimated propensity score} \tag{3.4}$$

Note, we can easily isolate subject i's propensity score in equation 3.4 to obtain:

$$\hat{e}(\mathbf{x}_i) = \frac{e^{\hat{g}(\mathbf{x}_i)}}{1+e^{\hat{g}(\mathbf{x}_i)}} \tag{3.5}$$

As I previously mentioned, GBM is an algorithm that iteratively builds a set of simple regression trees which are then combined to obtain a final model. Initially, the algorithm sets $\hat{g}(\mathbf{x})$ equal to the baseline log-odds of treatment: $\log(\bar{z}/(1-\bar{z})$ where $\bar{z}$ is the sample proportion of treated subjects. Next, the algorithm searches for an adjustment, $h(\mathbf{x})$, to improve model fit.

How should we measure model fit? Well, the model should maximize the likelihood of observing the given data! Let's explore this further.

Consider a study with N subjects. Let Z_i denote the random variable corresponding to treatment for subject i where $i = 1, \ldots, N$. We defined the propensity score for subject i, $e(\mathbf{x}_i)$, as the conditional probability of receiving treatment, $Z_i = 1$. As such, we can describe the distribution of a single treatment variable as: $Z_i \sim \text{Bernoulli}(e(\mathbf{x}_i))$ for $i = 1, \ldots, N$. Z_i has a corresponding marginal probability density function:

$$P(Z_i = z_i) = (e(\mathbf{x}_i))^{z_i}(1-e(\mathbf{x}_i))^{1-z_i} \tag{3.6}$$

From equation 3.6, we can plug in $z_i = 1$ and easily see that $P(Z_i = 1) = e(\mathbf{x}_i)$.

In this study, we consider N independent subjects who each have an observed treatment value $z_1, \ldots, z_N$. Let $\underline{z}$ denote the vector consisting of $z_1, \ldots, z_N$. To represent the likelihood, L, of observing $\underline{z}$, we take the product of the marginal PDFs to obtain a joint PDF. More specifically,

[Note: $e(\mathbf{x}_i) = \frac{e^{g(\mathbf{x}_i)}}{1+e^{g(\mathbf{x}_i)}}$]

$$L = P(\underline{z} \mid e(\mathbf{x}_1), \ldots, e(\mathbf{x}_N)) = \prod_{i=1}^{N} \left(\frac{e^{g(\mathbf{x}_i)}}{1 + e^{g(\mathbf{x}_i)}} \right)^{z_i} \left(\frac{e^{g(\mathbf{x}_i)}}{1 + e^{g(\mathbf{x}_i)}} \right)^{1-z_i} \tag{3.7}$$

In improving the model's 'goodness of fit', we seek to maximize the likelihood, L, of observing $\underline{z}$. However, we will soon see that maximizing $\ln(L)$ is easier, and, due to the monotonic property of the natural log, equally valid. So, we will quantify an improvement in model fit via maximizing the Bernoulli log-likelihood, $\ln(L)$.

To derive the Bernoulli log-likelihood equation, first take the log of both sides of equation 3.7 to obtain the following equation.

$$l = \ln(L) = \sum_{i=1}^{N} z_i \ln \left(\frac{e^{g(\mathbf{x}_i)}}{1 + e^{g(\mathbf{x}_i)}} \right) + (1 - z_i) \ln \left(\frac{e^{g(\mathbf{x}_i)}}{1 + e^{g(\mathbf{x}_i)}} \right) \tag{3.8}$$

Next, use properties of logarithms to decompose the fractions on the right side of equation 3.8.

$$l = \sum_{i=1}^{N} z_i \Big[g(\mathbf{x}_i) - \ln(1 + e^{g(\mathbf{x}_i)}) \Big] + (1 - z_i) \Big[- \ln(1 + e^{g(\mathbf{x}_i)}) \Big] \tag{3.9}$$

Finally, simplify equation 3.9 and obtain the equation below known as the Bernoulli log-likelihood.

$$l = \sum_{i=1}^{n} z_i g(\mathbf{x}_i) - \log(1 + \exp(g(\mathbf{x}_i)) \tag{3.10}$$

We can see from equation 3.10 that l is maximized when there is agreement between $g(\mathbf{x}_i)$ and z_i such that $g(\mathbf{x}_i)$ is negative when $z_i = 0$ and $g(\mathbf{x}_i)$ is positive when $z_i = 1$ [McCaffrey et al., 2005]. This makes intuitive sense as we maximize the log-odds when the treatment assignment indicator variable agrees with the log-odds of receiving treatment.

In searching for an adjustment, $h(\mathbf{x})$, McCaffrey et al. [2005] note that the expected log-likelihood of the data should increase with the adjustment h where λ is a step-size constant. In terms of expected value,

$$E(l(\hat{g} + \lambda h)) > E(l(\hat{g})) \tag{3.11}$$

If the algorithm finds an adjustment such that the above equation holds, then the model becomes $g(\mathbf{x}) \leftarrow g(\mathbf{x}) + \lambda h(\mathbf{x})$ [McCaffrey et al., 2005]. The process of iteratively searching and adding adjustments continues for a specified number of iterations.

Now, what is this adjustment, $h(\mathbf{x})$? In generalized boosted models, $h(\mathbf{x})$ takes the form of a regression tree and fits the residuals of the current model. As McCaffrey et al. [2005] note, choosing the adjustment, $h(\mathbf{x})$ to be a regression tree which models the residuals of the current fit is equivalent to estimating the derivative of the Bernoulli log-likelihood function (see equation 3.10). The residual for subject i is the difference between z_i, the treatment indicator, and the estimated propensity score, $e(\mathbf{x}_i)$. More specifically, the model's residual for subject i is defined as:

$$r_i = z_i - \frac{1}{1 + \exp(-\hat{g}(\mathbf{x}_i))} \text{ where } \hat{g}(\mathbf{x}_i) = \ln\left(\frac{e(\mathbf{x}_i)}{1 - e(\mathbf{x}_i)}\right) \tag{3.12}$$

Following the construction of the regression tree to model the residuals, the tree will have partitioned the subjects into k terminal nodes, denoted $T_1 \dots T_k$. Freedman and Berk [2008] suggests finding the optimal adjustment to the current model, $\hat{g}(\mathbf{x})$, separately for each terminal node. More specifically, the optimal adjustment for all the individuals within the kth terminal node is defined by:

$$h(\mathbf{x}) = \arg\max_\theta \sum_{x_i \in T_k} z_i(\hat{g}(\mathbf{x}_i) + \theta) - \log(1 + \exp(\hat{g}(\mathbf{x}_i) + \theta)) \tag{3.13}$$

$$h(\mathbf{x}) \approx \frac{\sum_{x_i \in T_k} z_i - e(\mathbf{x}_i)}{\sum_{x_i \in T_k} e(\mathbf{x}_i)(1 - e(\mathbf{x}_i))} \tag{3.14}$$

The following steps, as described by McCaffrey et al. [2005], compactly summarize the generalized boosted model algorithm:

1. Initialize $\hat{g}_0(\mathbf{x}) = \ln(\frac{\bar{z}}{1-\bar{z}})$
2. For m in $1 \dots M$, do the following:
   - Compute $r_i = z_i - \frac{1}{1+\exp(-\hat{g}_{m-1}(\mathbf{x}_i))}$ for $i = 1, \dots, N$
   - Build a regression tree to model all the r_i. As a result, the data gets partitioned into k terminal notes denoted $T_1 \dots T_k$.

- Compute the optimal update for each terminal node (as suggested by McCaffrey et al. [2005]):

$$\theta_{k(\mathbf{x})} = \frac{\sum_{x_i \in T_k} z_i - e(\mathbf{x}_i)}{\sum_{x_i \in T_k} e(\mathbf{x}_i)(1 - e(\mathbf{x}_i))} \tag{3.15}$$

- Update the overall model to: $\hat{g}_m(\mathbf{x}) \leftarrow \hat{g}_{m-1}(\mathbf{x}) + \lambda\theta_{k(\mathbf{x})}$ where λ denotes a shrinkage parameter and $k(\mathbf{x})$ determines which terminal node an observation with feature vector $\mathbf{x}$ falls into.

It is also important to note that there is a stochastic component within each new regression tree adjustment. More specifically, the algorithm selects a different random sub-sample of the data and builds the adjustment, $h(\mathbf{x})$, using this subset [McCaffrey et al., 2005]. Freedman and Berk [2008] suggests that a sub-sample constructed using 50% of the original data can reduce the bias and variance of our final propensity score model.

For a generalized boosted model, there are various important parameters to specify prior to model construction. More specifically, shrinkage, number of weak learners, and maximum depth within a weak learner are parameters that control the learner's tendency to overfit the data. The shrinkage parameter controls the learning rate by scaling the contribution of each tree to the enlarging model. If the shrinkage constant is small, a learner will generally require a larger number of weak learners to achieve the same predictive capabilities. McCaffrey et al. [2005] suggests using a shrinkage parameter of 0.0005. The maximum depth of each tree determines the complexity of interactions allowed within each model. Generally, the smaller the maximum depth (i.e., the number of splits within a tree), the less likely the model is to overfit the data. McCaffrey et al. [2005] suggests limiting the number of splits within each regression tree to 4, allowing for 4-way interactions between covariates in a given tree. While generalized boosted models are powerful, the parameter tuning is vital and takes substantially more time than random forest parameter tuning.

In addition to the time-intensive parameter tuning, boosted CART is sensitive to data outliers as sequential trees are trained to fit the residuals of the prior model. Due to the constant focus on minimizing the residuals, boosted CART is also prone to overfitting if not adequately reigned through parameter tuning. Lastly, because the individual trees in a generalized boosted model are built one after another, building this model can be very time-intensive; this contrasts a random forest where all of the individual trees are

built in parallel and thus do not use information from the prior trees. Aside from these significant but manageable flaws, generalized boosted models also present many benefits.

As McCaffrey et al. [2005] note, a GBM inherits many of the benefits of regression trees for propensity score estimation. For example, each individual regression tree is computationally quick and can handle many different data types (continuous, ordinal, categorical, and missing values). Moreover, McCaffrey et al. [2005] note that due to the invariant nature of regression trees to one-to-one variable transformations, "whether we use age, log(age), or age^2, we get exactly the same propensity score adjustments". Additionally, including a large number of covariates or a covariates that are highly correlated does not negatively impact the GBM model. Aside from the benefits of regression trees, the boosting nature of a GBM allows the model to capture main effects and produce a smooth fit [McCaffrey et al., 2005]. These countless advantages of a GBM motivate its use for propensity score estimation.

Chapter 4

Neural Networks for Propensity Score Estimation

4.1 Motivation for Neural Networks

In the previous chapters, I detailed logistic regression and tree-based techniques as potential models for predicting treatment assignment given pretreatment covariates. While popular, logistic regression requires specification of the relationships and distributions of covariates prior to model building; the need for variable specification prompted the use of a non-parametric models. Tree-based methods such as random forest and boosted CART provided non-parametric alternatives for propensity score estimation methods. However, many of the tree-based methods are sensitive to outliers and have a tendency to overfit the data. To remedy the short comings of both logistic regression and tree-based methods, I propose neural networks as an alternative method for propensity score estimation.

First proposed in 1944 by Warren McCullough and Walter Pitts, artificial neural networks are an alternative class of machine learning methods capable of modeling highly complex functions through a dense interconnected stream of nodes. A neural network loosely mirrors the process of a human brain. The human brain contains billions of neurons each of which intakes, processes, and transfers electric signals. More specifically, a neuron intakes electric signals through dendrites, processes the information in the neuron cell body, and then converts the processed signals to a single output which is transferred through the axon to the next neuron. The connection region between the

axon of one neuron and the dendrites of another are called synapses. Human learning occurs in the strengthening or weakening of these synaptic connections. This process of human learning can be partially simulated through an artificial neural network (ANN) [Aggarwal, 2018].

A neural network consists of an interconnected sequence of nodes, represented by neurons, which intakes an observation's feature vector and outputs the desired prediction [Aggarwal, 2018]. To unwrap the mathematics underneath the madness, we must first start simple.

4.2 The Perceptron

The simplest neural network is called the perceptron. A perceptron is a single-layer network where a set of inputs is directly mapped to an output via a linear function of weighted edges and inputs.

Consider the situation where each observation in the training data is of the form $(\mathbf{x}, z)$ where $\mathbf{x} = [x_1, \ldots, x_p]$ representing a feature vector of length p and $z \in$ {-1,1} representing the observed binary classification label. In the case of propensity score estimation, $\mathbf{x}$ denotes the vector of pre-treatment covariates and z denotes the treatment indicator. Since $\mathbf{x}$ is of size p, the input layer must contain p input nodes which transmit the features with weighted edges, $\mathbf{w} = [w_1, \ldots, w_p]$, to an output node. At the output node, we calculate the linear function, $\mathbf{w} \cdot \mathbf{x} = \sum_{i=0}^{p} w_i x_i$. Since this is a classification problem, the final output, $\hat{z}$, is determined by the sign function defined below which maps a real value to positive or negative 1. The sign function serves as the activation function, here.

$$\hat{z} = \text{sign}\{\mathbf{w} \cdot \mathbf{x}\} = \text{sign}\{\sum_{i=0}^{p} w_i x_i\} \tag{4.1}$$

Consider the following diagram representing the structure of a perceptron. Note, the first input is 1, and $1 * w_0$ will always be constant, w_0. w_0 represents the bias and is useful when the the distribution of the binary class variable is highly imbalanced [Aggarwal, 2018]. In the diagram below, we see that each input, x_i, is weighted according to w_i. Then, we sum the weighted inputs where 0 is the activation threshold.

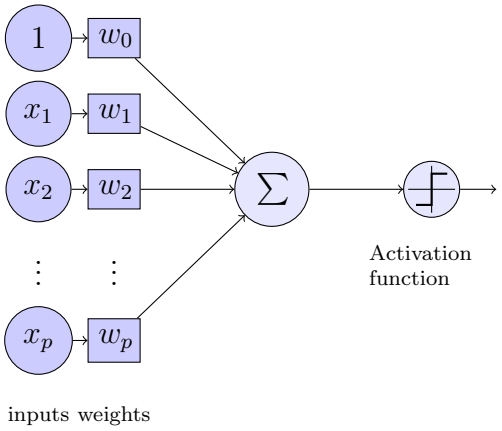

Figure 4.1: Diagram of a perceptron. A feature vector, $[x_1, \ldots, x_p]$, is weighted according to a weight vector, $[w_1, \ldots, w_p]$, with an additional bias term represented by w_0. We sum the weighted inputs and pass the result through an activation function with a threshold of 0.

To determine the weights associated with each input, we must solve yet another optimization problem. The goal of the perceptron algorithm is to minimize the number of mis-classifications. More formally, the goal is to:

$$\text{Minimize}_{\mathbf{w}} \left\{ \sum_{i=0}^{N} (z_i - \hat{z}_i)^2 = \sum_{i=0}^{N} (z_i - \text{sign}\{\mathbf{x}_i \cdot \mathbf{w}\})^2 \right\} \tag{4.2}$$

The utilization of gradient descent to minimize the loss function requires a differentiable loss function. The gradient associated with equation 4.2 is a non-differentiable function with a staircase-like structure. Thus, to use gradient descent, we must use a smooth approximation of the gradient [Aggarwal, 2018].

$$\bigtriangledown L_{\text{Smooth}} = \sum_{i=0}^{N} (z_i - \hat{z}_i)\mathbf{x}_i \tag{4.3}$$

The weights in the network are updated as each data point, $(\mathbf{x}_i, z_i)$, travels through. More specifically, when $\mathbf{x}_i$ enters the network, the weights are updated as follows.

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha(z_i - \hat{z}_i)\mathbf{x}_i \tag{4.4}$$

Note, that the weights are only updated when the error term $(z_i - \hat{z}_i)$ is non-zero. The perceptron algorithm randomly cycles through all the training examples and updates the weights accordingly until convergence. Once the algorithm reaches convergence, $\mathbf{w} \cdot \mathbf{x} = 0$ defines the linear hyperplane which best classifies the data [Aggarwal, 2018]. It follows that the perceptron algorithm will do well on data that is linearly separable and poorly on data that is non-linearly separable. The perceptron algorithm's inability to model more complex data patterns motivates the use of multi-layer neural networks.

4.3 Multi-Layer Neural Networks

A perceptron, as described above, contains only a single computational layer which occurs at the output layer. In contrast, multi-layer neural networks contain more than a single computational layer. In a multi-layer neural network, the computations performed at each layer are not visible outside the network; these intermediate computational layers are called "hidden" layers. Just as a perceptron, there are nodes within each visible and hidden layer. Importantly, each node of one layer connects to every node of the following layer. This creates a *dense* network where every node is interconnected. Additionally, these connections are forward directional. In other words, the output of the previous layer becomes the input of the following layer. This property of successive layers feeding into one another is called a *feed-forward* network [Aggarwal, 2018].

The output of each layer within a neural network can be represented as a column vector with dimensionality determined by the number of nodes within the layer. More specifically, if a neural network with k layers has respective number of nodes in each layer $l_1, \ldots, l_k$, then the output column vector of each layer denoted $\bar{h}_1, \ldots, \bar{h}_k$ has dimension $l_1, \ldots, l_k$. For instance, consider the first layer which contains l_1 nodes. Each of the l_1 nodes will produce an output which together can be represented as a column vector, denoted $\bar{h}_1$, of dimension $l_1 \times 1$.

We can define the dimensions of a weight matrix W in terms of the dimension of the previous and successive layers. Let W_1 denote the matrix of weights applied when connecting the input layer of dimension p to the first hidden layer of dimension l_1. Since each node must connect to every node of the successive layer, W_1 is a matrix of size $p \times l_1$. Similarly, the

matrix of weights applied to the outputs of the ith hidden layer denoted W_i has dimension $l_i \times l_{i+1}$. Lastly, assume the neural network has size o output. Then the dimension of the weight matrix connecting the final hidden layer to the output layer of sizes l_k and o respectively has dimension $l_k \times o$ and is denoted W_{k+1}. Let Φ denote an arbitrary activation function. We can formalize the weighting and transformation process from layer to layer with the following series of recursive equations.

$$\bar{h}_1 = \Phi(W_1^T \mathbf{x}) \qquad \text{input to hidden layer} \tag{4.5}$$

$$\bar{h}_{i+1} = \Phi(W_{i+1}^T \bar{h}_i) \; \forall i \in \{1, \ldots, k-1\} \qquad \text{hidden to hidden layer} \tag{4.6}$$

$$o = \Phi(W_{k+1}^T \bar{h}_k) \qquad \text{hidden to output layer} \tag{4.7}$$

As shown above, the output vector of each layer is determined by weighting the inputs and then applying an activation function, Φ. In section 4.2, we noted that the perceptron only gives us access to linear models when trying to classify the data; the linearity restriction motivated the use of multi-layer neural networks. Multi-layer neural networks allow access to non-linear models through the use of *non-linear activation functions*.

If we instead used a linear activation function (a.k.a. the identity function), $\Phi(*) = *$, the output of the neural network would reduce to a linear combination of the inputs. Consider a network with a single hidden layer. Let $\mathbf{x}$ denote a vector of inputs for layer i. The resulting output is $\bar{h}_i$.

$$\bar{h}_i = \Phi(W_i^T \cdot \mathbf{x}) = W_i^T \cdot \mathbf{x} \tag{4.8}$$

Similarly, the resulting output for the network is defined as follows.

$$\bar{h}_{i+1} = \Phi(W_{i+1}^T \cdot \bar{h}_i) = W_{i+1}^T \cdot \bar{h}_i \tag{4.9}$$

The two equations can be composed in the following way when Φ is the identity function.

$$\bar{h}_{i+1} = W_{i+1}^T \cdot \Phi(W_i^T \cdot \mathbf{x}) = W_{i+1}^T (W_i^T \cdot \mathbf{x}) \tag{4.10}$$

Note, the output of the network, $\bar{h}_{i+1}$, can be fully represented as a linear combination of the inputs.

In propensity score estimation, we need a method that allows us to capitalize on interactions and power relationships between various covariates.

This motivates the use of a non-linear activation function for multi-layer neural networks.

For the purposes of propensity score estimation, I will define Φ as the **Sigmoid function**. The Sigmoid function maps inputs on the real line to the interval (0,1) and thus is a logical choice for estimating probabilities as in propensity score estimation. The Sigmoid function is defined as follows where $*$ is an arbitrary input:

$$\Phi(*) = \frac{1}{1 + e^{-*}} \tag{4.11}$$

4.3.1 Training a Neural Network with Backpropagation

In the case of a perceptron, we were able to use a stochastic gradient descent algorithm to train the single-layer neural network. More specifically, we could calculate the gradient and adjust the weight vector accordingly. However, in the case of multi-layer neural networks, computing the gradient is difficult due to the compositional nature of the loss function. To calculate the gradient of a composition loss function, we use back propagation. In essence, back propagation allows us to see how much of the total loss each node is responsible for and update the weights accordingly. If a node is responsible for a heavy loss, it makes intuitive sense to decrease the weight associated with the given node.

So, we need a way to estimate how the total loss changes with respect to a given weight w_i. The change in loss with respect to weight is given by the partial derivative, $\frac{\partial L}{\partial w_i}$. If $\frac{\partial L}{\partial w_i} > 0$, the loss increases as w_i increases. If $\frac{\partial L}{\partial w_i} < 0$, the loss decreases as w_i increases. We can update w_i to w_i' in such a way that the loss will always decrease. Let λ denote a learning rate parameter. More specifically,

$$w_i' = w_i - \lambda \frac{\partial L}{\partial w_i} \tag{4.12}$$

Note, that if $\frac{\partial L}{\partial w_i} > 0$, then w_i' will be smaller than w_i. This is intuitively sound as we want to decrease the weight associated with nodes which causes the loss function to increase as w_i increases. Similarly, if $\frac{\partial L}{\partial w_i} < 0$, the updated weight, w_i', will be larger than the original weight, w_i. The same intuition

applies: we want to increase the weights associated with the nodes which cause the loss function to decrease.

In order to control the rate at which the neural network updates the weights and converges on a final model, we introduce a hyperparameter λ. λ is typically a small value between 0 and 1. Choosing a small λ will result in a much longer training phase as the weights are barely changing through each iteration. However, choosing a large λ could result in converging on a sub-optimal set of parameters.

Now that we have a rule to update the weights (see equation 4.12), we turn our focus to calculating $\frac{\partial L}{\partial w_i}$. The main tool to use is the chain rule from calculus. Let $F = f(g(x))$. The chain rule is as follows:

$$F' = f'(g(x)) \cdot g'(x) \tag{4.13}$$

Using the uni-variate and multi-variate chain rule, we can propagate the resulting loss backwards through the network to the initial layer. This technique allows us to see the loss attributable to a given node and update the corresponding weight accordingly.

For a given weight w_i, we find all possible paths from weight w_i to output $\hat{e}$. Using the multi-variate chain rule, we calculate the partial derivative of the loss with respect the weight w_i for each path. To obtain the overall derivative of loss with respect to w_i, we aggregate the partial derivatives [Aggarwal, 2018]. We now consider a more concrete example.

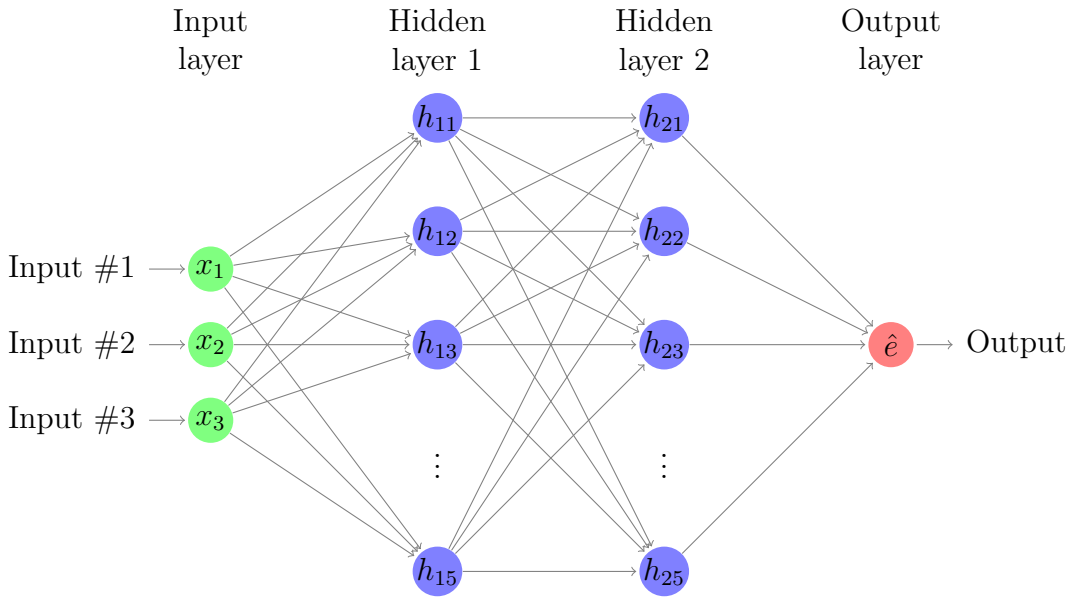

Figure 4.2: Multi-layer neural network to illustrate backpropagation

Consider all of the paths backwards from $\hat{y}$ to w_1 in Figure 4.2 above. In order to calculate $\frac{\partial L}{\partial w_1}$, we must trace each path and aggregate the partial derivatives. The calculus is as follows:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{e}} \cdot \frac{\partial \hat{e}}{\partial w_1} \tag{4.14}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{e}} \cdot \left(\frac{\partial \hat{e}}{\partial h_{21}} \cdot \frac{\partial h_{21}}{\partial w_1} + \frac{\partial \hat{e}}{\partial h_{22}} \cdot \frac{\partial h_{22}}{\partial w_1} + \cdots + \frac{\partial \hat{e}}{\partial h_{25}} \cdot \frac{\partial h_{25}}{\partial w_1} \right) \tag{4.15}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{e}} \cdot \left(\frac{\partial \hat{e}}{\partial h_{21}} \cdot \frac{\partial h_{21}}{\partial h_{11}} \cdot \frac{\partial h_{11}}{\partial w_1} + \frac{\partial \hat{e}}{\partial h_{22}} \cdot \frac{\partial h_{22}}{\partial h_{11}} \cdot \frac{\partial h_{11}}{\partial w_1} + \cdots + \frac{\partial \hat{e}}{\partial h_{25}} \cdot \frac{\partial h_{25}}{\partial h_{11}} \cdot \frac{\partial h_{11}}{\partial w_1} \right) \tag{4.16}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{e}} \cdot \left(\frac{\partial \hat{e}}{\partial h_{21}} \cdot \frac{\partial h_{21}}{\partial h_{11}} + \frac{\partial \hat{e}}{\partial h_{22}} \cdot \frac{\partial h_{22}}{\partial h_{11}} + \cdots + \frac{\partial \hat{e}}{\partial h_{25}} \cdot \frac{\partial h_{25}}{\partial h_{11}} \right) \frac{\partial h_{11}}{\partial w_1} \tag{4.17}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{e}} \cdot \left(\frac{\partial \hat{e}}{\partial h_{21}} \cdot \frac{\partial h_{21}}{\partial h_{11}} + \frac{\partial \hat{e}}{\partial h_{22}} \cdot \frac{\partial h_{22}}{\partial h_{11}} + \cdots + \frac{\partial \hat{e}}{\partial h_{25}} \cdot \frac{\partial h_{25}}{\partial h_{11}} \right) x_1 \tag{4.18}$$

As we can see above, the computational complexity is already extremely high with only 2 hidden layers. The number of paths from output to input weight grows exponentially as the neural network gets deeper [Aggarwal,

2018]. Due to the exponential path growth and the impracticality of computing the gradient using calculus, scientists use a technique called dynamic programming instead. Without detailing the specifics of dynamic programming, Aggarwal [2018] summarizes the beauty of this technique nicely.

> "Using dynamic programming to efficiently aggregate the product of local gradients along the exponentially many paths in a computational graph results in a dynamic programming update that is identical to the multi-variable chain rule of differential calculus."

4.4 Pros and Cons of Neural Networks for Propensity Score Estimation

Neural networks, while often thought of as black boxes, offer many benefits in propensity score estimation. First, neural networks are a fully non-parametric class of models. This means there is no need to specify interactions, the order and transformations, or the distribution of variables a priori. Just like the tree-based learners discussed in chapter 3, the absence of prior assumptions prevents researchers from incorrectly specifying false relationships between variables. The lack of prior assumptions is especially important in situations where researchers do not have expert domain knowledge and may lack information regarding relationships between explanatory variables. Secondly, as Westreich et al. [2010] notes, neural networks are highly skilled at dealing with high dimensional data. More specifically, when treatment can be accurately predicted considering lots of explanatory variables (high-dimensional data) but can't be accurately predicted using only a few variables, a neural network should be the model of choice.

While neural networks boast many advantages, there are notable drawbacks. First, training a neural network is more of an art than a science, meaning there are few universal rules for parameters like number of hidden nodes and learning rate [Westreich et al., 2010]. As such, choosing parameter values which avoid issues such as local minima and model overfitting is extremely challenging to the non-expert. Secondly, after fitting a neural network, the resulting weights have no accessible interpretation; the lack of interpretation of weights is drastically different than the highly interpretive results of β_0 and β_1 in logistic regression. While propensity score estimation

prioritizes accuracy over interpretability, it is still a noteworthy drawback. For instance, we may be interested in digging down into the impact of a confounding variable on treatment assignment, however a neural network's lack of interpretability is an insurmountable barrier to do so.

Chapter 5

Simulation and Results

The goal of this chapter is to simulate data with a known causal treatment effect and use propensity score estimation theory to uncover the causal treatment effect. Moreover, I look at sample size, strength of causal treatment effect, correct specification of variable relationships, and unspecified confounding variables to investigate under what conditions logistic regression, regression trees, random forests, generalized boosted models, and neural networks perform best.

In the following simulations, I compare the estimated average treatment effects for each of the five propensity score estimation techniques (logistic regression, regression trees, random forests, generalized boosted models, and neural networks). All average treatment effects, $\widehat{ATE}$, and corresponding standard errors, $SE(\widehat{ATE})$, are computed using Equations 1.7 and 1.8. The simulations are conducted using the R package *simcausal*.

In conducting the simulations, I specify all variables present in the model and their respective distributions. Moreover, I specify the treatment assignment mechanism and the structural equation that defines the causal treatment-response relationship. In all simulations, I generate **50 different data sets** for a given simulation setting and use propensity score estimation to obtain an estimated average treatment effect for **each** data set. Then, I average the estimated average treatment effects over all 50 data sets to minimize estimation variability. In the first simulation, I compare the setting in which each of the 50 data sets has a sample size of $N = 100$ to $N = 10,000$. In all other simulations, each of the 50 generated data sets has a sample size of $N = 100$ due to computational limitations.

5.1 Treatment Assignment Mechanism

In a randomized controlled trial, treatment assignment is determined by the researchers, and thus the underlying treatment assignment mechanism is known. However, in observational studies, the underlying treatment assignment mechanism is unknown to researchers. Since we are in a simulation setting, I will define the treatment assignment mechanism and then 'hide' the treatment mechanism when calculating the average treatment effect; in essence, I want to see if propensity score estimation and stratification can account for the withheld information within the treatment assignment mechanism.

I define the underlying equation which determines treatment assignment as follows where e, f_1, f_2 are constants. Let (X_1, X_2) be distributed according to a multivariate normal distribution where $\mu = [0.5, 1]$ and $\Sigma = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$. Let $V \sim N(0, 1)$.

Let p denote the probability of treatment. As such,

$$Z = \begin{cases} 1 \text{ (treatment)}, & p = \frac{e^{e+f_1X_1+f_2X_2+V}}{1+e^{e+f_1X_1+f_2X_2+V}} \\ 0 \text{ (no treatment)} \end{cases} \tag{5.1}$$

Unless otherwise specified, let $e = 0.5, f_1 = 0.25, f_2 = 0.75$. I can re-write the treatment selection equation with the constant values defined above as:

$$Z = \begin{cases} 1 \text{ (treatment)}, & p = \frac{e^{0.5+0.25X_1+0.75X_2+V}}{1+e^{0.5+0.25X_1+0.75X_2+V}} \\ 0 \text{ (no treatment)} \end{cases} \tag{5.2}$$

It is important to note that Equation 5.2 explicitly defines the relationship between treatment assignment and covariates, X_1, X_2. However, equation 5.2 does not tell us anything about the relationship between the response variable and covariates.

5.2 Simulation 1: Vary Treatment Effect Size and Sample Size

Let Y denote the response variable of interest, X_1, X_2 denote pre-treatment covariates, Z denote a treatment indicator variable, and U, V denote random

noise. Let $(X_1, X_2), U, V$ be independent. Independence means that knowing the value of one random variable conveys no information about the value of another.

For **Simulation 1**, the relationship between the response variable Y and the predictor variables, Z, X_1, X_2, is defined as follows where b, c_1, c_2, d are constants. It is important to recognize that a, b, c_1, c_2, d are population parameters and thus **unknown** outside of the simulation framework. As scientists, the goal is to uncover the population level parameters of interest through unbiased estimation.

$$Y = a + bZ + c_1X_1 + c_2X_2 + dU \tag{5.3}$$

Note, equation 5.4 explicitly defines the dependence of Y on covariates X_1, X_2. The dependence of both treatment, Z, **and** response, Y, on covariates, X_1, X_2, is what confounds the relationship between treatment and response. The confoundedness of treatment and response initially limits our ability to make causal treatment effect claims.

In Equation 5.3, b is the underlying causal treatment effect as treatment receipt for subject i ($Z_i = 1$) necessarily causes Y to increase by b. Through propensity score stratification and average treatment effect estimation, we try to arrive at an unbiased estimate for b, the true causal treatment parameter. For Simulation 1, let $a = 1, c_1 = 0.1, c_2 = 2, d = 1$. I will vary the treatment effect b, so I will leave it arbitrary. We can plug these values into Equation 5.3 to obtain:

$$Y = 1 + bZ + 0.5X_1 + 2X_2 + U \tag{5.4}$$

In the graph below, the x-axis denotes parameter value ATE under which the data was generated; in other words, the x-axis denotes the value of b in Equation 5.4. The y-axis represents the estimated ATE, $\widehat{ATE}$. Note, the diagonal black line denotes perfect unbiased estimation of ATE. Each point on the graph represents $\widehat{ATE}$ over 50 simulated data sets. The vertical line protruding from each point denotes a single standard error in either direction of $\widehat{ATE}$.

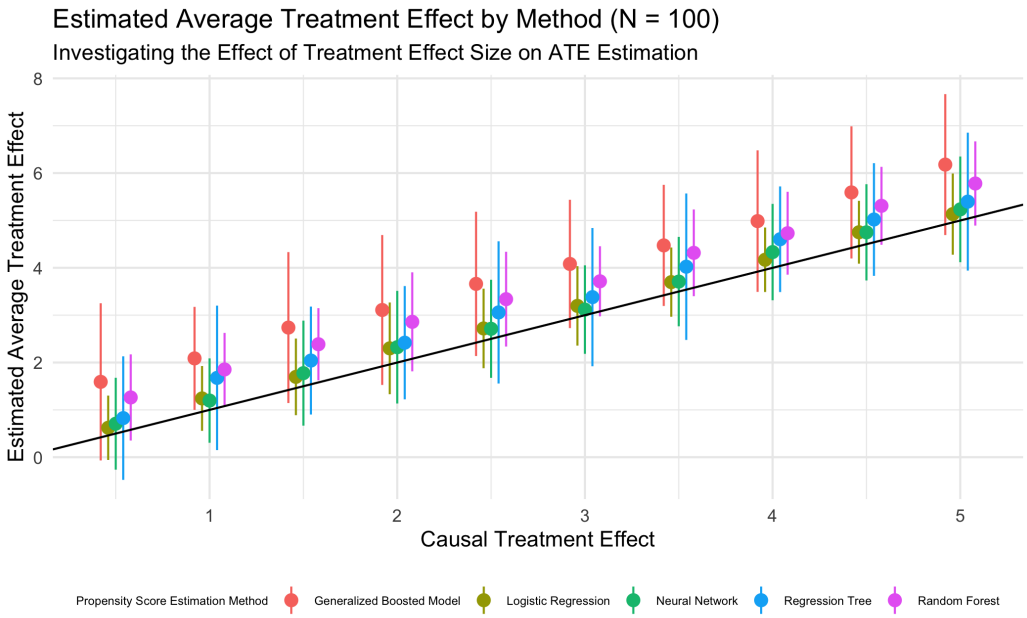

Figure 5.1: Varying the causal treatment effect size with N=100. Neural networks and logistic regression compete for the least biased $\widehat{ATE}$. Note, the black diagonal line denotes perfect unbiased $\widehat{ATE}$.

In Figure 5.1, we see that neural networks and logistic regression produce similar treatment effect biases while generalized boosted models, random forests, and regression trees perform significantly worse. Interestingly, as the treatment effect size approaches 5, all methods except generalized boosted models and random forests are able to uncover the treatment effect with similar small biases.

Next, I once again vary the parameter value ATE with an increased sample size of $N = 10,000$. I am interested in increasing the sample size 100-fold due to my pre-concieved notion that neural networks perform increasingly well (perhaps even outperform logistic regression) as sample size increases.

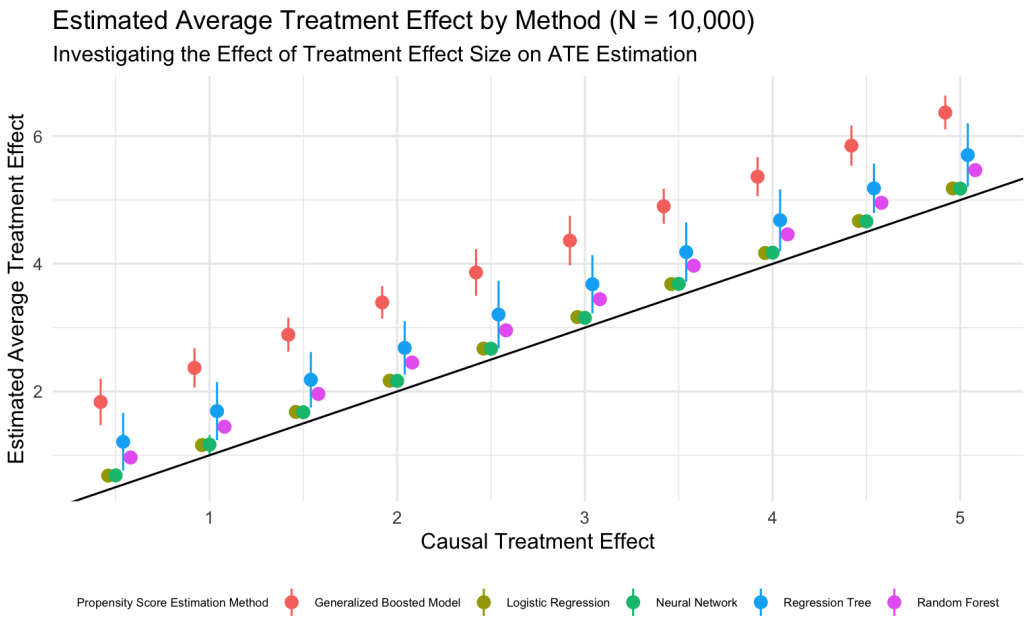

Figure 5.2: Varying the causal treatment effect size with N=10,000. For large treatment effects, neural networks just barely outperform logistic regression. Notably, when $ATE = 1$, neural networks are slightly more variable than logistic regression. Lastly, all standard error bars shrink dramatically compared to $N = 100$; this is expected as sample size and standard error are inversely proportional.

As shown in Figure 5.2, neural networks seem to outperform logistic regression when the sample size is large, though the difference is minimal.

5.3 Simulation 2: Vary Pre-treatment Covariate Strength

In this simulation, I investigate the effect of increasing the coefficients attached to the pre-treatment covariates. I hypothesize that the causal treatment effect will be more difficult to estimate when the pre-treatment covariates have a larger impact on the response variable. Why? In my intuition, stronger pre-treatment covariates will cloud the relationship between treat-

ment and response, thereby making it more difficult to estimate ATE.

The treatment selection mechanism is the same as in Simulation 1 (5.2). That is,

$$Z = \begin{cases} 1 \text{ (treatment)}, & p = \frac{e^{0.5+0.25X_1+0.75X_2+V}}{1+e^{0.5+0.25X_1+0.75X_2+V}} \\ 0 \text{ (no treatment)} \end{cases} \tag{5.5}$$

However, the response relationship is now different. More specifically, the ATE will stay constant ($b = 2$) while I vary the coefficients attached to the pre-treatment covariates (c_1, c_2). The constant, a, will stay fixed at 1. Mathematically, the response relationship is of the following form:

$$Y = 1 + 2Z + c_1X_1 + c_2X_2 + U \tag{5.6}$$

Note: The distributions of Z, X_1, X_2, U are the same as defined in Simulation 1 (5.2).

In varying c_1, c_2, I test the following values: $c_1 = c_2$ where $c_1 \in$ [0.05, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. In reality, c_1 and c_2 need not be equal. However, for simplicity sake, I chose to vary the strength where $c_1 = c_2$.

The graphical results are as follows. Note, $ATE = 2$, as denoted by the horizontal black line.

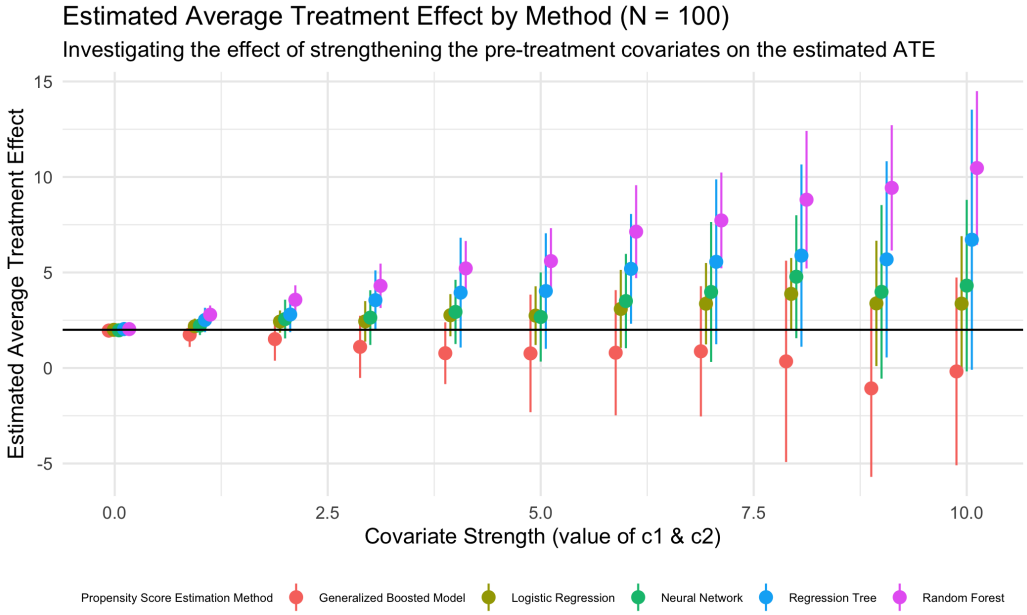

Figure 5.3: Here, I investigate how well each propensity score estimation method is able to uncover the ATE as the strength of pre-treatment covariates increases. As the strength of the pre-treatment covariates increases, logistic regression, neural networks, and GBMs produce smaller biases in $\widehat{ATE}$ than random forests and regression trees.

5.4 Simulation 3: Model Misspecification

In the prior simulations, logistic regression consistently estimated the treatment effect with the least bias of all propensity score estimation methods. However, all variable relationships and functional forms were correctly specified in those simulations; in other words, the underlying treatment assignment mechanism matched the logistic model specification. As mentioned in Section 2.3, parametric models including logistic regression require proper variable specification to achieve optimal predictive performance. As such, the purpose of Simulation 3 is to investigate the effect of model misspecification on the ability of a propensity score estimation method to produce an unbiased estimate for the ATE.

Model misspecification can occur when building a model to estimate the propensity scores (i.e., logistic regression, CART methods, or neural networks). Misspecification includes incorrectly specifying the functional form of a covariate, excluding an interaction between covariates, or excluding a covariate all together. Notably, all propensity score estimation methods will suffer should the researcher leave out an important pre-treatment covariate. However, only logistic regression will suffer if a researcher incorrectly specifies the functional form of a covariate or excludes an interaction between covariates. Logistic regression uses maximum likelihood estimation to estimate the β coefficients under the pre-specified model. As such, incorrect model assumptions can lead to sub-optimal models. In contrast, non-parametric models including regression trees, random forests, boosted CART, and neural networks look to the data to determine optimal functional forms of and interactions between variables, and thus will be unaffected by misspecification of covariate functional form and interaction.

5.4.1 Simulation 3a: Unspecified Interaction Term

First, I add an interaction term to the underlying treatment assignment mechanism and purposefully leave the interaction out of the propensity score model. More specifically, the treatment assignment mechanism is of the following form where f_3 is an unspecified constant:

$$Z = \begin{cases} 1 \text{ (treatment)}, p = \frac{e^{0.5+0.25X_1+0.75X_2+f_3X_1X_2+V}}{1+e^{0.5+0.25X_1+0.75X_2+f_3X_1X_2+V}} \\ 0 \text{ (no treatment)} \end{cases} \tag{5.7}$$

The response equation has no interactions and is of the same form as in Section 5.2.

$$Y = 1 + 2Z + 0.5X_1 + 2X_2 + U \tag{5.8}$$

Now, this is the important part. In specifying the logistic regression model, I do **not** including the interaction term, $X_1 \cdot X_2$. More specifically, the logistic regression model is of the following form:

$$\ln\left(\frac{e_i}{1-e_i}\right) = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 V_i \tag{5.9}$$

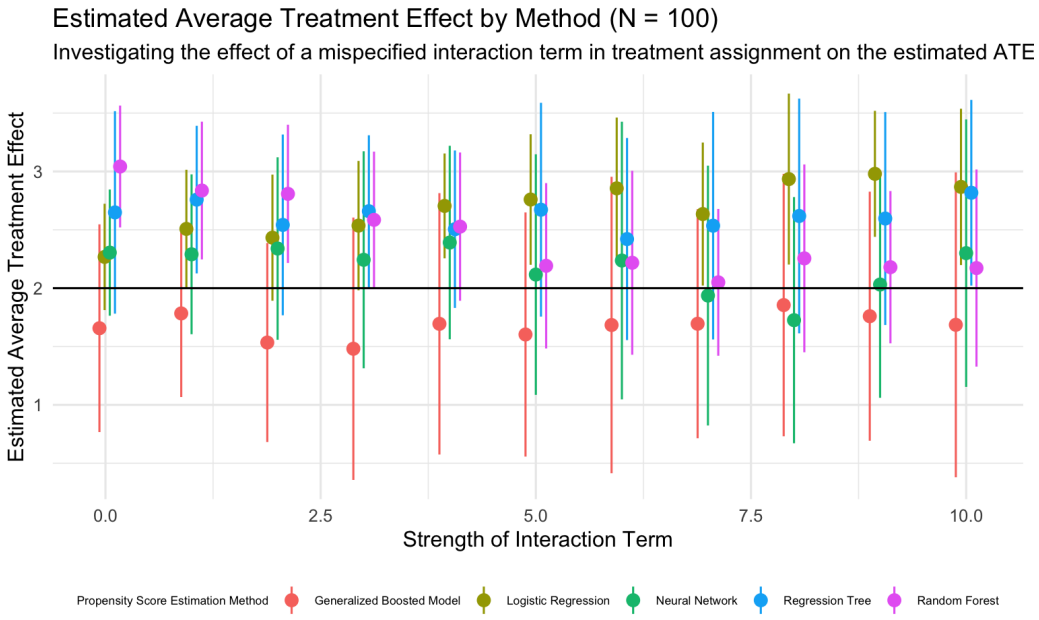

Figure 5.4: Here, I investigate the effect of an interaction term present in the underlying treatment assignment mechanism however excluded from the propensity score model. As shown, the bias of $\widehat{ATE}$ obtained via logistic regression increases fairly drastically as the withheld interaction term increases in strength. In contrast, the bias of $\widehat{ATE}$ obtained via random forest decreases as the strength of the interaction increases. In this circumstance, it is clear that models whose structure can easily incorporate unspecified interactions produce significantly lower ATE biases.

In Figure 5.4, logistic regression performs increasingly worse as the strength of the interaction term increases thereby illustrating the main pitfall of logistic regression as a method for propensity score estimation. In contrast, the $\widehat{ATE}$ obtained from regression trees, generalized boosted models, and neural networks propensity score estimation methods do not drastically change with the increase in interaction term. Interestingly, when we use random forests to model propensity scores, the bias of the average treatment effect decreases as the strength of interaction increases. Prior literature by Setoguchi et al. [2008] agrees with the stability of neural networks and the decrease in performance of logistic regression when there is an unspecified interaction term

present in the underlying treatment assignment mechanism.

5.4.2 Simulation 3b: Covariate Exclusion

In propensity score estimation, another form of model misspecification is the failure to specify all pre-treatment covariates; failure to do so should impact all models. In this simulation, I investigate the effect of leaving out a pre-treatment covariate from the propensity score estimation model on the bias of $\widehat{ATE}$.

In order to have adequate covariates to model while simultaneously leaving one out, I introduce another covariate to the model. Consider a covariate, W. Let $W \sim$ Bernoulli(p = plogis(-0.5) ≈ 0.38).

The underlying treatment selection is defined as follows where f_4 is an unspecified constant:

$$Z = \begin{cases} 1 \text{ (treatment)}, p = \frac{e^{0.5+0.25X_1+0.75X_2-f_4W+V}}{1+e^{0.5+0.25X_1+0.75X_2-f_4W+V}} \\ 0 \text{ (no treatment)} \end{cases} \tag{5.10}$$

The response equation is defined as follows:

$$Y = 1 + 2Z + 0.5X_1 + 2X_2 - 0.3W + U \tag{5.11}$$

The logistic model is defined as follows. Note, W is **not** specified in the model.

$$\ln\left(\frac{e_i}{1-e_i}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 V \tag{5.12}$$

The random forests, generalized boosted models, and neural networks are trained using only X_1, X_2, V. Again, we essentially withhold W from the propensity score model.

To test how excluding a covariate affects ATE estimation, I vary the coefficient preceding W, f_4. More specifically, I calculate $\widehat{ATE}$ using the four propensity score estimation techniques (logistic regression, regression trees, generalized boosted models, and neural networks) for each of the following values of f_4: $f_4 \in [0.05, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]$.

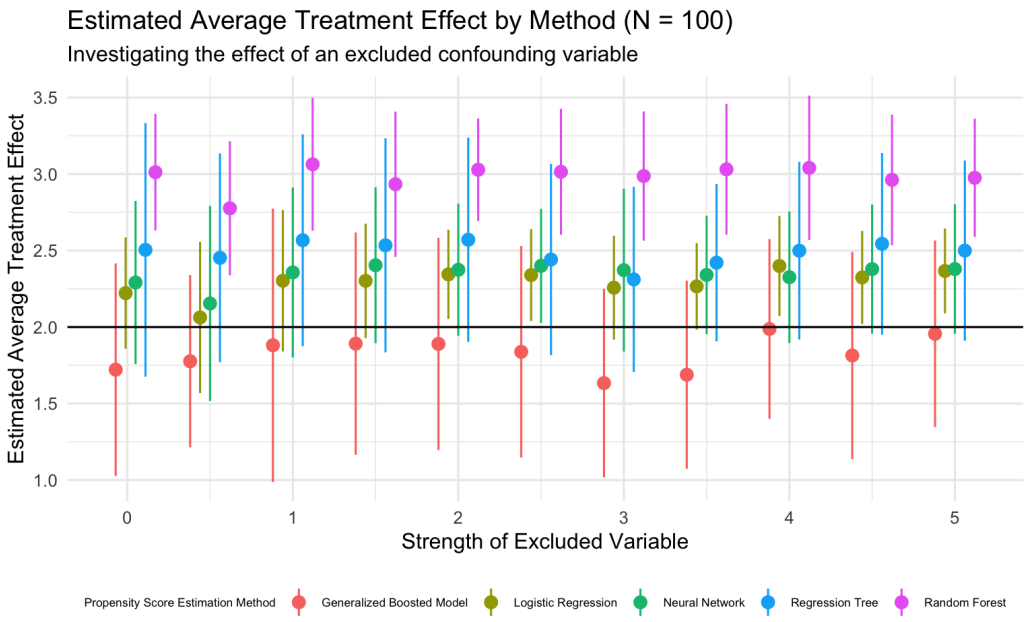

Figure 5.5: Here, there exists a covariate in the treatment assignment mechanism which is not present in the propensity score model. We see in the figure above that, as the strength of the unspecified covariate increases, GBMs yield smaller biases for the ATE than alternative methods. Additionally, logistic regression and neural networks yield fairly low biases for the ATE, however notably larger biases compared to GBMs.

As shown above in Figure 5.4, interestingly, GBMs mostly yield the smallest bias for $\widehat{ATE}$ as compared to other propensity score estimation techniques. However, once again we see that logistic regression and neural networks yield similar low-biased results. Notably, the $\widehat{ATE}$ produced from the GMB model is increasingly variable as compared to logistic regression and neural networks.

5.5 Conclusions and Takeaways

Historically, randomized controlled trials have stood as the gatekeeper to a researcher's ability to declare causal treatment-response relationships. In

a randomized controlled trial, there are, on average, no pre-existing differences between the treatment groups; the covariate balance between treatment groups is a result of a randomized allocation of treatment. While randomized controlled trials open doors for causal claims, they are often impractical or unethical. As such, researchers are often confined to observational studies where there exist systematic differences in pre-treatment covariates between treatment groups. Instead of abandoning ship, propensity score stratification offers a statistical method by which we can obtain an unbiased estimate for the average treatment effect [Rosenbaum and Rubin, 1984]. In the most broad sense, propensity score stratification divides the subjects into strata according to pre-treatment similarity. So long as we have pre-treatment covariate balance between treatment groups within each strata, we can estimate an unbiased treatment effect for each strata. Finally, Rosenbaum and Rubin [1984] show that averaging the individual strata treatment effects yields an unbiased average causal effect of treatment on response.

While the theory of propensity score stratification is fascinating, effective estimation of the propensity scores is vital to thereby obtain strata with pre-treatment covariate balance. In propensity score estimation, we are trying to predict the probability by which a subject will receive treatment given a vector of pre-treatment characteristics. Historically, propensity scores have been estimated via logistic regression, a parametric model that requires researchers to pre-specify relationships between and the functional form of covariates. However, without sufficient domain knowledge and statistical modeling capabilities, it is very difficult to correctly specify a logistic model. As such, I motivate the use of regression trees, random forests, generalized boosted models, and neural networks as alternative methods for propensity score estimation; these are non-parametric models and thus do not require pre-specification of covariate functional forms and interactions. To investigate the ability of each model to produce an unbiased estimation for the average treatment effect, I turn to simulation.

In all simulations, I, as the simulator, determine the underlying treatment assignment mechanism and treatment-response relationship. To determine the most effective propensity score estimation method, I test the effect of sample size, pre-treatment covariate strength, model misspecification, and covariate exclusion on ability to uncover the average treatment effect. As shown in Figure 5.1, logistic regression and neural networks outperform the alternatives when the propensity score model is perfectly specified with a sample size $N = 100$. When we increase the sample size to $N = 10,000$ and

maintain perfect specification propensity score model, we again see logistic regression and neural networks outperform alternative methods.

Next, I investigate the effect of increasing the influence of pre-treatment covariates on treatment assignment. As the pre-treatment covariates become increasingly important in determining one's response, logistic regression produces the least biased ATE estimation. However, GBMs and neural networks both produce an $\widehat{ATE}$ with only a slight increased bias compared to that of logistic regression as the covariate strength increases.

I then turn my focus to the effect of propensity score model misspecification on the ability of an estimation method to uncover the average treatment effect. First, I consider the presence of an interaction term in the underlying treatment assignment mechanism but the exclusion of such interaction term in propensity score model building. In the situation of an unspecified interaction term, neural networks, random forests, and GBMs produce increasingly low biases for the ATE as the strength of the interaction term increases. Notably, logistic regression does not perform well under the circumstances of an unspecified interaction term. Second, I consider the presence of a covariate in the underlying treatment effect mechanism but absence of such covariate in propensity score model building. In the setting of an excluded covariate, GBMs produce the least biased ATE estimate, however neural networks and logistic regression perform only slightly worse.

So, given the results above, if someone were to ask me "What propensity score estimation method is most effective at uncovering the average treatment effect?" I would reply with excitement, "Well, it depends!" More specifically, if the researcher is confident in their domain knowledge and ability to model interactions between variables and the correct functional forms of variables, I suggest using logistic regression for propensity score estimation. However, without sufficient domain and/or statistical knowledge, I suggest using neural networks for propensity score estimation.

Bibliography

J Audrain-McGovern and NL Benowitz. Cigarette smoking, nicotine, and body weight. *Clinical pharmacology and therapeutics*, 90(1), 2011. doi: 10.1038/clpt.2011.105.

Paul R. Rosenbaum and Donald B. Rubin. Reducing bias in observational studies using subclassification on the propensity score. *Journal of the American Statistical Association*, 79(387):516–524, 1984. doi: 10.1080/01621459.1984.10478078.

Paul R. Rosenbaum and Donald B. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 04 1983. doi: 10.1093/biomet/70.1.41.

Markus Neuhäuser, Matthias Thielmann, and Graeme D. Ruxton. The number of strata in propensity score stratification for a binary outcome. *Archives of Medical Science*, 14(3):695–700, 2018. ISSN 1734-1922. doi: 10.5114/aoms.2016.61813. URL `http://dx.doi.org/10.5114/aoms.2016.61813`.

Bambang Widjanarko Otok, Marsuddin Musa, Purhadi, and Septia Devi Prihastuti Yasmirullah. Propensity score stratification using bootstrap aggregating classification trees analysis. *Heliyon*, 6(7), 2020. doi: https://doi.org/10.1016/j.heliyon.2020.e04288.

Daniel Westreich, Justin Lessler, and MJ Funk. Propensity score estimation: neural networks, support vector machines, decision trees (cart), and meta-classifiers as alternatives to logistic regression. *Journal of Clinical Epidemiology*, 63(8):826–833, 2010. doi: https://doi.org/10.1016/j.jclinepi.2009.11.020.

Bas B.L. Penning de Vries, Maarten van Smeden, and Rolf H.H. Groenwold. Propensity score estimation using classification and regression trees in the presence of missing covariate data. *Epidemiologic Methods*, 7, 2018. doi: https://doi.org/10.1515/em-2017-0020.

Leo Breiman. Bagging predictors. *Machine Learning*, 24, 1995. doi: 10.1007/BF00058655.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

Leo Breiman. Random forests. *Machine Learning*, 45, 2001. doi: 10.1023/A:1010933404324.

Daniel McCaffrey, Greg Ridgeway, and Andrew Morral. Propensity score estimation with boosted regression for evaluating causal effects in observational studies. *Psychological methods*, 9:403–25, 01 2005. doi: 10.1037/1082-989X.9.4.403.

David A. Freedman and Richard A. Berk. Weighting regressions by propensity scores. *Evaluation Review*, 32(4):392–409, 2008. doi: 10.1177/0193841X08317586.

Charu C. Aggarwal. *Neural Networks and Deep Learning*. Springer, 2018. doi: 10.1007/978-3-319-94463-0.

Soko Setoguchi, Sebastian Schneeweiss, M. Alan Brookhart, Robert J. Glynn, and E. Francis Cook. Evaluating uses of data mining techniques in propensity score estimation: a simulation study. *Pharmacoepidemiology and Drug Safety*, 17(6):546–555, 2008. doi: 10.1002/pds.1555.