Machine Learning and the Propensity Score

Theory and Applications in Agricultural Economics

Mitchell Cameron

2024-08-09

Table of contents

Preface

This project is submitted in partial fulfillment of the requirements for the Master of Applied Science in Statistics at the University of Otago. My academic background in economics and politics sparked an enduring interest in causal inference, particularly its role in shaping evidence-based policymaking. However, my focus has evolved to include machine learning — a field that, while not traditionally central to economics, offers powerful tools for refining causal analysis.

The motivation for this project stems from my desire to bridge the gap between propensity score methods and machine learning. While the literature is rich with simulation studies, there is a noticeable lack of comprehensive, tutorial-style resources that guide readers through the application of machine learning to propensity score estimation. This project aims to fill that void, providing a practical and accessible exploration of these techniques with approachable theoretical discussion and coded examples in R. As a dedicated user of R, all the packages and methodologies discussed in this project exist in the R ecosystem. Although comparable tools exist in other languages and software.

The intended audience for this work includes individuals with a foundational understanding of causal inference and machine learning, particularly those interested in enhancing propensity score models. However, my discussion extends beyond propensity scores; much of what is covered is relevant to anyone using machine learning for probability prediction.

Acknowledgments

I would like to express my deepest gratitude to several individuals whose support, guidance, and encouragement have been invaluable throughout the course of this project.

First and foremost, I extend my sincere thanks to Associate Professor Matthew Parry, my supervisor. His Socratic method of teaching has fostered not only my academic growth but also my critical thinking abilities. I am particularly grateful for his flexibility and understanding as I navigated through various iterations and changes in the direction of this project. His patience and positivity, even as the project's focus shifted, were invaluable, and I deeply appreciate his guidance throughout this journey.

I am also grateful to Dr Conor Kresin, whose knowledge of causal inference provided helpful background for this project. Additionally, my thanks go to Dr Falco J. Bargagli Stoffi, whose correspondence was incredibly helpful during earlier iterations of this project.

To my fellow students — Bethany, Jess, Shalini, Sarah, Maryam, Anna, Alex, Jackson, Teddy, Steven, and Chris — thank you for the camaraderie, the shared challenges, collective encouragement, and practical advice.

Lastly, but by no means least, I want to thank my girlfriend, Dani for her patience, support, and understanding throughout this process. I also want to express my gratitude for her meticulous help with proofreading, which has greatly improved the clarity and quality of this project.

Presentation Tools

Quarto is an open-source publishing system that enables the creation of dynamic documents, reports, presentations, books, and websites using R, Python, or Julia. It integrates code, markdown,

and graphics seamlessly, making it ideal for reproducible research and communication. I the Quarto Book template to create which is hosted on GitHub.

1 Introduction and Background

todo;

- proof appendix

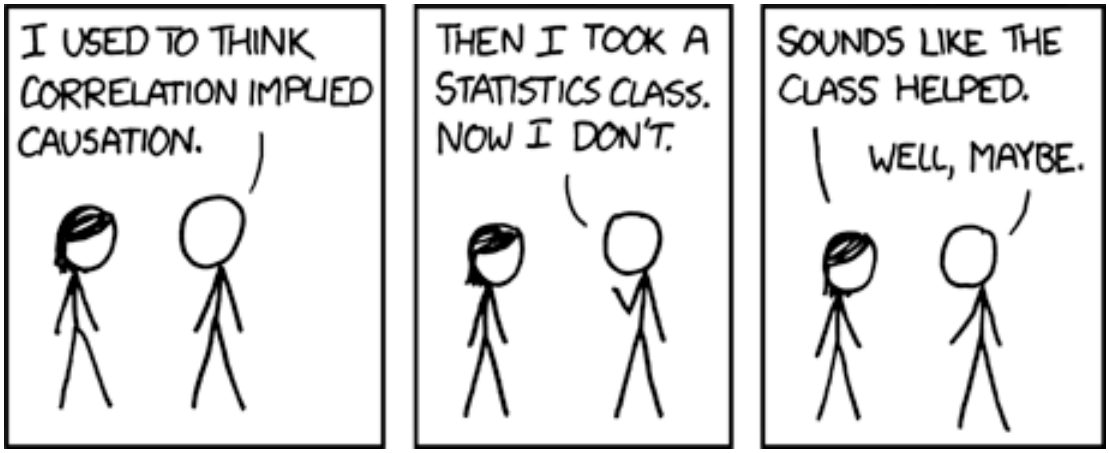

Figure 1.1: The comic plays with the difference between causation vs. correlation. Original: https://xkcd.com/552/

1.1 What is Causal Inference?

Causal inference is a field of study that focuses on identifying and estimating causal relationships between *things*. It goes beyond correlation by establishing a cause-and-effect relationship. Causal inference methods often utilise counterfactual reasoning to estimate the causal effect of an exposure

or treatment on an outcome. Such counterfactual reasoning is used unbeknownst every day. For example, if someone misses their bus and thinks, "If I had left home five minutes earlier, I wouldn't have missed it," they are engaging in counterfactual reasoning. In everyday life, policy-making, medicine, and business, understanding the size and nature of a cause is essential for decision making and avoiding misleading conclusions. In this background chapter I discuss key ideas in causal inference such as the potential outcomes framework, common estimands, and assumptions.

1.2 Layout

Chapter 1 provides a foundational introduction to causal inference, which is essential for understanding the context of this project. This section is designed to provide a concise background for readers who may not be familiar with causal inference, ensuring they have the necessary foundation to follow the rest of the project.

Chapter 2 introduces the central focus of this project: the use of machine learning to estimate propensity scores. The section begins with a traditional introduction to propensity scores, explaining their role in balancing covariates between treatment and control groups to reduce estimator bias. This leads into a discussion on the limitations of conventional propensity score estimation methods, which often rely on logistic regression. These limitations motivate the use of machine learning algorithms for propensity score estimation. The section then provides a theoretical comparison of common machine learning algorithms such as random forests, bootstrap aggregation (bagging), and gradient boosting machines. The goal of this section is to provide readers with an intuitive understanding of how machine learning can enhance propensity score estimation, setting the stage for practical applications later in the project.

Chapter 3 presents a comprehensive tutorial for implementing machine learning techniques in the estimation of propensity scores. This section is highly practical, walking the reader through the

software implementations available for estimating and assessing propensity scores. The National Supported Work (NSW) program dataset is used as a running example throughout the tutorial. This dataset is commonly used in causal inference studies due to its simplicity and well-documented treatment effect, making it an ideal example candidate. By the end of this section, readers should be able to apply these methods to their own datasets, replicating the steps and analyses presented.

Chapter 4 provides a detailed replication study of Jena et al. (2012), a paper that examines the impact of fair trade certification on farmers' livelihood in Ethiopia. This section builds on the original findings of the aforementioned authors by applying machine learning-based propensity score methods. The replication study is designed to demonstrate the practical advantages of using machine learning for propensity score estimation in a real-world setting. Specifically, my replication re-examines the causal effect on per capita income. This comparison highlights how machine learning can potentially lead to more accurate and reliable estimates of treatment effects.

Chapter 5 provides a comprehensive overview of the findings from this project. It synthesizes the key insights gained from the theoretical discussions, the practical tutorial, and the replication study. This section also outlines potential avenues for future research, emphasizing the importance of continued exploration of machine learning methods in causal inference.

Appendix A offers supplementary material that supports the main content of the project. This includes explanations of the datasets used, along with coded examples of loading and manipulating the data in R. Additionally, Appendix B presents custom functions developed for this project, which are used to present results and facilitate analyses throughout the project.

1.3 Potential Outcomes Framework

The Potential Outcomes Framework, also known as the Rubin Causal Model, was introduced by Rubin (1974) and builds upon the work of Splawa-Neyman (1923). The framework dominates how researchers think about causal inference by formalising counterfactual reasoning. Rubin defines a causal effect as a defined comparison between two states of the world. For each individual, there are two potential outcomes: one if they receive the treatment and one if they do not. The causal effect is the difference between these two potential outcomes. Hence we have two *potential outcomes*, one with the treatment and one without.

The framework is highly flexible and adaptive, extending beyond traditional notions of "treatment" in medical or experimental contexts. It can apply to any kind of intervention, exposure, or condition that could influence an outcome, whether it's a medical treatment, policy change, environmental exposure, or even abstract events like decisions or natural occurrences. Philosophically, the framework aligns with a view of the world that considers reality through alternative scenarios or *what-ifs*.

Consider a binary treatment variable, let the treatment for an observation be a random variable, T, with a realisation $t_i \in \{0, 1\}$ under control and treatment. The absence of treatment is refer to as the *control* state. Let $Y_i(1)$ and $Y_i(0)$ be the two potential outcomes for observation i under treatment and control. Let the individual treatment effect (ITE) be defined as the difference between the two potential outcomes:

$$\tau_i = Y_i(1) - Y_i(0). \tag{1.1}$$

> **Note 1: Fundamental Problem of Causal Inference**
>
> There is a clear problem that only the outcome under either treatment or control is observable. If our observations are on people, then it is logically impossible for an individual to simultaneously both receive and not receive the treatment. For example, if someone takes medication to relieve a headache and their headache improves, it could never know what would have happened if they did not take the medication. This leads to the commonly discussed *fundamental problem of causal inference* - it is impossible to observe both potential outcomes. A counterfactual, the counter to the observed outcome, is infeasible and can never be practicably known.

Let the observed outcome for i be denoted $y_i(1)$ and $y_i(0)$ under treatment or control. Many causal inference methods involve finding or estimating a counterfactual to compare outcomes to sove some variation of Equation 1.1. Let an estimated potential outcomes for i be denoted $\hat{y}_i(1)$ and $\hat{y}_i(0)$ under treatment or control.

1.4 Estimands

In causal inference, there are multiple parameters of interest called and estimand. The preferred estimand depends on the motivating example, discipline, or intended interpretation of a result.

The most basic estimand is the *average treatment effect* or the ATE denoted τ_{ATE} which is the average amount of effect on all individuals in the population regardless of whether they receive the treatment or not. This can be written as:

$$\begin{aligned} \text{ATE} &= E[\tau_{ATE}] \\ &= E[Y(1) - Y(0)] \\ &= E[Y(1)] - E[Y(0)]. \end{aligned} \tag{1.2}$$

Under certain conditions, such as a randomised control trial, Equation 1.2 can be an estimated using the explicit equation:

$$\widehat{\text{ATE}} = \hat{\tau}_{ATE} = \frac{1}{N_t}\sum_{i=1}^{n}(y_i \mid t_i = 1) - \frac{1}{N_c}\sum_{i=1}^{n}(y_i \mid t_i = 0), \tag{1.3}$$

where N_t and N_c are is the number of treated and control observations. Essentially, Equation 1.3 is just a difference in the mean outcome between the two groups.

The second parameter of interest is the *average treatment effect on the treated* or ATT and is the difference (contrast) between the potential outcomes of those who actually receive the treatment. In other words, considering observations where $t_i = 1$, what is the effect of the treatment? This can be written as:

$$\begin{aligned} \text{ATT} = \tau_{ATT} &= E[\tau \mid T = 1] \\ &= E[Y(1) - Y(0) \mid T = 1] \\ &= E[[Y(1) \mid T = 1] - E[Y(0) \mid T = 1]. \end{aligned} \tag{1.4}$$

The final parameter is the *average treatment effect on the control* or ATC which is similar to the ATT but on those who are actually under control. The ATC is the contrast between the two potential outcomes for individuals which are actually in the control. This is also known as the average treatment effect on the untreated or the ATU. It can be written as:

$$\begin{aligned} \text{ATC} = \tau_{ATC} &= E[\tau \mid T = 0] \\ &= E[Y(1) - Y(0) \mid T = 0] \\ &= E[[Y(1) \mid T = 0] - E[Y(0) \mid T = 0] \end{aligned} \tag{1.5}$$

For the estimated ATT and ATC, no explicit expression exist. Estimation is completed using G-methods to obtain contrasts of potential outcomes (see Naimi, Cole, and Kennedy 2017).

1.5 Assumptions in Causal Inference

Assumptions are made for the potential outcomes framework to be logically coherent and for estimands to be identifiable. Firstly, *independence* must be assumed, implying the potential outcomes are independent of T. This assumption is also known as unconfoundedness, ignorability, or selection on observables, and means there is no confounding relationship between the treatment and potential outcomes. This matters as confounding variables can create a spurious relationship between the treatment and the outcome, leading to biased estimates of the treatment's effect. Hence, the treatment assignment should be random, allowing an unbiased estimate. Mathematically independence can be stated as:

$$Y(1), Y(0) \perp\!\!\!\perp T. \tag{1.6}$$

Independence implies exchangeability meaning the individuals in the treatment and control groups could be swapped and the potential outcomes are still the same. A weaker assumption is conditional independence that states that assignment into treatment is random conditioned on some X:

$$Y(1), Y(0) \perp\!\!\!\perp T \mid X. \tag{1.7}$$

The assumption requires that covariates must be known and measurable which may not always hold. Independence motivates the use of randomisation in experimental contexts as this should

guarantee independence. Chapter 2 discusses conditional independence and uses propensity scores to *condition* on covariates.

A second assumption is *positivity*. This means that for each i, the condition probability when $X = x$ of being in either the treatment or control group is strictly between 0 and 1. In other words, $\Pr(T = 1 \mid X = x) > 0$ and $\Pr(T = 0 \mid X = x) > 0$. This ensures that all observations have at least some chance of receiving either the treatment or control. If not, it is theoretically impossible to obtain both potential outcomes and so the treatment effect cannot be estimated.

Building on positivity, the third assumption is *common support*. This implies the treatment and control groups overlap in terms of their characteristics. Overlap is crucial because it ensures that for every person in the treatment group, there are similar individuals in the control group—similar in terms of age, gender, income, and other factors. Mathematically, for all of i, if the conditional probability of being treated, $\Pr(T = 1 \mid X = x)$, is near to 1, and $\Pr(T = 0 \mid X = x)$ is near to 0, then there are no compatible cases and there is no *common support*. Without compatible cases, it is not possible to satisfy exchangability and so treatment effect estimates are likely to be biased.

The fourth assumption is *consistency* between the potential outcome and observed outcome. For every i, the observed outcome under treatment equals the potential outcome under treatment. Additionally, the observed outcome under control equals the potential outcome under control. Mathematically, $y_i(1) = Y_i(1)$ and $y_i(0) = Y_i(0)$ that leads to a switching equation which defines y_i as a function of the potential outcomes:

$$y_i = T_i Y_i(1) + (1 - T_i) Y_i(0). \tag{1.8}$$

Notice the logic of this equation, when $T = 1$ then $y_i = Y_i(1)$ as the second term becomes zero. Similarly, when $T = 0$ then $y_i = Y_i(0)$ as the first term becomes zero.

The final key assumption is called the *stable unit treatment value assumption* or SUTVA. This is a complex way of stating that there is no interference between observations. More specifically, neither potential outcome is affected by the treatment status of any other individual. To borrow terminology from economics, there are no externalities or spillover effects from one observations' treatment status to another observations' potential outcomes.

> **Note 2: Why Assumptions Matter**
>
> In causal inference, especially when working with observational data, it is critical that these assumptions are considered. If these assumptions do not hold, any model, regardless of the modelling assumptions, will not have a causal interpretation. Unfortunately, there are no tests that can confirm if these causal assumptions hold and thus researchers must understand the context and data generating process in which they operate.

2 Propensity Scores and Machine Learning

2.1 A Conventional Approach: Propensity Scores and Balance

In a randomised control trial (RCT), researchers believe treatment and control groups are similar because of randomisation. In this case, the similar groups are compatible and should not have systematic differences. In observational data, the exposure to a treatment is unlikely to be random, implying there may be systematic differences between groups. Systematic differences refer to consistent variations or disparities between groups in the study. These differences are not due to random chance but rather indicate a pattern or trend, perhaps due to selection-bias. As groups are not comparable, Equation 1.3 leads to a biased estimate of the treatment effect.

For example, consider the causal question: *"How much does obtaining a bachelors degree increase lifetime earnings?"*. Individuals who complete a bachelor's degree are not selected at random for university programs (treatment) and may have different observable attributes than those who do not attend a university (control). Perhaps those who attend university have higher academic abilities, higher motivation, or grew up with parents with higher income. Because of these systematic group covariate differences, a simple comparison of mean income could lead to attributing university attendance as the *cause* of higher incomes when the effect is confounded by the differences in covariates between groups. In this example, the confounding covariates are academic ability, motivation, and parental income that impact the probability of someone obtaining a bachelors degree.

This discussion introduces the idea of *covariate balance* which is a key concept behind underlying propensity score methods.

> **Note 3: What is Covariate Balance**
>
> Covariate balance is the idea that covariates are approximately equivalent across treatment and control groups. If the distribution of each covariate is the same across each group, then the covariates are *balanced* and a meaningful comparison between groups can be made. Equally, similar covariates across groups implies exchangability as the two groups should be similar (thus can be exchanged). There is a conceptual link between covariate balance, unconfoundedness, and exchangeability meaning that Equation 1.6 is satisfied when covariates are balanced.

In the bachelor's degree example, suppose that comparable treatment and control individuals are matched together to create balanced pairs. Between these pairs, covariates are balanced such as the same academic ability, motivation, parental income, geographic residence etc. The covariates are said to be *conditioned on* by matching individuals on these covariates. Comparing the balanced matched pairs should result in a robust estimate of a bachelors degree's impact on earnings because the individuals are exchangeable and satisfy Equation 1.7. However, practically this matching is difficult to perform as exact matches cannot be made as the number of covariates increases. For example, finding two people with the same gender is simple but finding two people with the same gender, age, education, income, motivation, location, experience, and race is nearly impossible. Thus, there is a *dimensionality* problem as the dimension of the number of covariates increases.

Rosenbaum and Rubin (1983) offer a valuable tool for analysing observational data called the propensity score. The propensity score is the probability of treatment assignment conditioned on observed covariates. Essentially, the propensity score reduces the dimension of the number of covariates to a single dimension to avoid the dimensionality problem. Let the propensity score be denoted as $e(X)$ and be expressed as:

$$e(X) = P(T = 1|X = x). \tag{2.1}$$

A prediction of the conditional probability of treatment on covariates is a good summary of the covariate's effect on receiving the treatment. The covariate imbalance between bachelors degrees and controls arose from people self-selecting themselves into a bachelors degree programme because of these covariates. For example, people with higher motivation and academic ability are more likely to go to university. If it is the covariates that impact the probability of going to university, then a prediction of the probability of going to university based on these covariates should summarise the covariate effects.

It is hoped that conditioning on this propensity score should balance the data and meet the conditional independence assumption stated in Equation 1.7. There are many sources that offer a comprehensive guide to propensity score methods such as (Cunningham 2021, chap. 4) who provides applications and coded examples in R, Python, and Stata.

> **Note 4: Balance and Propensity Scores**
>
> Note that an RCT will satisfy Equation 1.6 as randomisation implies the potential outcomes are independent of the treatment assignment. Propensity score methods aim to satisfy Equation 1.7 as the potential outcomes are independent of the treatment status conditioned on some covariates. Thus, $Y(1), Y(0) \perp\!\!\!\perp T \mid X$ is satisfied by $Y(1), Y(0) \perp\!\!\!\perp T \mid \widehat{e(X)}$.
> Conditioning on the propensity score aims to replicate an RCT in observational data by balancing covariates between groups. When observations are conditioned on their propensity score, differences in outcomes can be confidently attributed to the treatment itself, rather than to pre-existing differences in covariates.

Two common methods that use propensity scores are propensity score matching (PSM) and inverse

propensity weighting (IPW).[1] PSM creates matched sets with similar propensity scores. IPW creates a balanced pseudo-population, where observations are weighted on the inverse of the propensity score. The pseudo-population is created by up-weighting observations with a low propensity score and down-weighting observations with a high propensity score.

At a high level, the conditioned property of the propensity score is translated into a model by using PSM or IPW. King and Nielsen (2019) provide a notable criticism of propensity score matching, which is a very interesting read. In the following examples, IPW is used due to theoretical advantages and ease of software implementation.

2.1.1 Assessing balance

Balance assessment is an important step to ensure that conditioning on the propensity score has been successful. A commonly recommended measure of covariate balance is the standardised mean difference or **SMD**. This is the difference in the mean of each covariate between treatment groups standardised by a standardization factor so that it is on the same scale for all covariates.

SMDs close to zero indicate good balance. P. Austin (2011) notes that 0.1 is a common *threshold* for determining if a variable is balanced. This threshold is a guideline to the approximate region that indicates a covariate is balanced and should not be interpreted as a binary rule. Additionally, a variance ratio below *2* is generally acceptable. For brevity, my analysis only considers the SMD.

2.1.2 Propensity Score Modelling with Logistic Regression

A conventional propensity score model uses logistic regression to predict a probability between 0 and 1. Models may be specified to include interaction terms and polynomial terms to capture com-

[1] IPW is also commonly known as inverse probability of treatment weighting (IPTW). IPTW uses propensity scores and can equally be called IPW.

plex trends in the data. There are a range of approaches for specifying a propensity score model, but the process is a heuristically driven art rather than science. (see Brookhart et al. 2006; Heinrich 2010). One suggestion is to include two-way interaction terms between covariates and squared terms and then remove terms which are not statistically significant. Notably, many researchers do not discuss the specification of propensity models in their papers. P. C. Austin (2008) reviews 47 papers that use propensity scores and find few perform adequate model selection, assess balance, or apply correct statistical tests.

It is important to note that the true propensity score is never observable. A propensity score that is close to the theoretically true probability is well calibrated. Poorly calibrated propensity scores may result in poor balance and biased estimation of the treatment effect. Propensity scores may be poorly calibrated as covariates may be omitted by error, poorly measured, or be unobservable. Logistic regression may not predict calibrated scores if the true relationship is non-linear or involves complex interactions between covariates. Another important note is that the propensity model itself does not have an informative *causal* interpretation. In logistic regression, the coefficients are the log-odds of the treatment assignment for each variable which is not informative of the desired estimand.

The first application of machine learning in causal inference was to predict propensity scores. Despite this, logistic regression still appears to be the most common model for predicting propensity scores.

2.2 Probability Machines: Probability Theory and Machine Learning

Supervised machine learning usually focuses on classifying observations into groups, or predicting continuous outcomes. Probability prediction is a hybrid of these tasks, aiming to predict the contin-

uous probability an observation belongs to a certain class. Machine learning methods that predict probabilities are sometimes called probability machines.

Probability machines are valuable in applications requiring calibrated probability predictions. For example, probability machines can predict loan defaults or other adverse events in finance. In marketing, they estimate the likelihood of customer response to a campaign. Gamblers and bettors want robust probability predictions to enhance their betting strategies. Probability machines can be applied wherever calibrated probability predictions are needed.

Probability machines offer many advantages over parametric methods like logistic regression:

1. **Improved Calibration**: Probability machines often provide better-calibrated predictions by capturing complex data relationships.

2. **Flexible Modelling**: Unlike parametric methods like logistic regression, probability machines don't rely on assumptions of additivity or linearity, allowing them to model intricate relationships that parametric models miss.

3. **Efficient Feature Selection**: These machines automatically select features, making them ideal for high-dimensional datasets where manual selection is impractical.

4. **Handling Missing Data**: Probability machines handle missing data robustly, minimizing the need for extensive data reprocessing and imputation.

5. **Simplified Data Exploration**: By exploring complex data structures in a data-driven way, probability machines simplify model specification. For instance, tree-based models remain unaffected by adding squared or interaction terms, streamlining the modeling process.

In causal inference, probability machines can predict better calibrated propensity scores and better estimate treatment effects. This discussion aims to clarify the use of probability machines in causal

inference given the unique requirements of propensity score specification. Probability machines are theoretically complex and there are unanswered questions in this space.

> **Note 5: A Particularly Important Method: Classification and Regression Trees**
>
> Moving forward, a particularly important model is the Classification and Regression Trees. L. Breiman et al. (1984) introduces method, commonly known as CART, that partitions data according to a splitting criterion, resulting in an "if this, then that" interpretation. CART models are also widely known as a decision trees. The splits are recursive, meaning splits are applied upon previous splits, such as trees breaking into branches and then leaves. The splits are also *greedy* as each potential split only considers information available at that split instead of past or future splits. Each parent node is split to create two child nodes and the final nodes of a CART model are called terminal nodes.
>
> For example, when classifying pets into cats versus dogs, the first split might be*"if barks"* and the second is *"heavier than 5 kg"*. The tree says *If it barks and is heavier than 5 kg, then it is a dog*.
>
> A single classification tree typically uses the Gini index to determine its splits. Each split aims to maximise node purity, meaning the nodes contain the highest possible proportion of one class. The Gini index measures impurity, with lower Gini values indicating higher purity. Intuitively, the aim of a classification tree's loss function is to minimise the misclassification rate of observations. By selecting splits that reduce the Gini index, the tree minimises classification error and increases accuracy.

2.2.1 Choice of Loss Function and Probability Prediction

The loss function measures the difference between a model's predictions and the actual target values, serving as a measure of a model's performance. Different loss functions influence a model's

behaviour so the choice of loss function is important. Classification models predict the category that each observation belongs to not the probability of each class. For instance, in fraud detection, banks use classifiers to distinguish between fraudulent and routine transactions. Many classification loss functions minimise classification errors and improve accuracy as this results in the best classification. A loss function like the Gini index is effective for classification problems but it is unclear if this applies to probability problems. In other words, minimizing misclassification error may not lead to accurate probability predictions.

At a high level, to classify an observation, x_i as an A or B, a model needs to determine if $\Pr(x_i = A)$ is less than or greater than 0.5. If $\widehat{\Pr}(x_i = A) > 0.5$, then it is more likely to be an A and if $\widehat{\Pr}(x_i = A) < 0.5$ then it is more likely to be a B. Thus, if x_i is an A, it is trivial if $\widehat{\Pr}(x_i = A)$ is 0.51 or 0.99 as this makes no difference to the classification as an A. But the difference between $\widehat{\Pr}(x_i = A) = 0.51$ and 0.99 is extreme for a probability machine. It is important to understand that classification models are optimised for classification accuracy rather than probability prediction. This distinction affects the performance of ensemble methods like random forests or bagging ensembles that use classification trees for probability prediction.

2.2.2 Bagging and Random Forest as Probability Machines

> **Note 6: A Quick Note on Ensemble Learning**
>
> Ensemble learning refers to a general framework of machine learning that combines multiple simple models to create a better overall model. The philosophy behind ensemble learning is rooted in the wisdom of crowds principle, where the collective decisions from multiple models often outperform that of individual models. Often, ensemble learning methods use multiple CART models.
>
> Bagging ensembles, random forests, and gradient boosting machines are all machine learning

methods that combine CART models and are all examples of ensemble learning algorithms. These methods are introduced in the following discussion.

Consider an ensemble method called the *lazy ensemble* that combines multiple CART trees. Each tree is fitted on the same data without any cross validation. In this ensemble, class probabilities are determined through a *vote count* method. Within the lazy ensemble, each tree makes a class prediction based on the majority class in a terminal node. For instance, if x_i lies in a terminal node where 80% of the observations are classified as an A, that *individual tree* will classify x_i as A. The ensemble's *overall prediction* for x_i is derived from the proportion of trees that classify x_i as A or B. Thus, the ensemble *counts votes* for each class across the ensemble. Let T be the total number of trees and b_t be the t-th tree in the ensemble. Let $\mathbb{I}(b_t(x_i) = A)$ be the indicator function that returns 1 when b_t predicts that observation x_i belongs to class A. The probability of class A for observation x_i is calculated as:

$$\widehat{\Pr}(x_i = A) = \frac{1}{T}\sum_{t=1}^{T}\mathbb{I}(b_t(x_i) = A). \tag{2.2}$$

Olson and Wyner (2018) note bias towards predictions of 0 or 1 when trees in an ensemble framework are highly correlated and a vote count method is used. In the lazy ensemble, each tree is identical and perfectly correlated implying that each tree will make the the same class prediction for each x_i. For such an ensemble, the predicted probabilities will will be exactly 0 or 1 using a vote count. Of course a lazy ensemble of identical trees would never be used but the intuition still applies in the real world where ensembles may have some degree of correlation. The larger the correlation, the more the probability predictions will exhibit a *divergence bias* towards 0 and 1. Notably, divergence bias is not problematic in classification applications, as a larger number of trees correctly classifying the observation is encouraging.

To reduce tree correlation and improve upon the lazy ensemble, a bagging ensemble (see Leo Breiman 1996) trains each tree on a randomly selected bootstrapped sample of the data. Random forests (see Leo Breiman 2001) further reduce tree correlation by considering a random number of variables at each split, commonly referred to as $mtry$ in software implementations. Note that these ensemble methods typically use a *vote count method* in the same way as the lazy ensemble. When $mtry$ is equal to to number of predictors, the model considers all variables at each split and the random forest is equal to a bagging ensemble. A lower $mtry$ should reduce the correlation between trees and decrease divergence bias as the structure of the tree is modified by the selected variables at each split. However, a lower $mtry$ also introduces other theoretical problems.

Consider the scenario where the binary outcome (treatment and control) of the ensemble is strongly related to a single predictor and weakly related to other noisy predictors. If $mtry$ is low then each split may not consider the strong predictor and more commonly splits on weak or noisy predictors. Each predictor has a chance of $\frac{mtry}{\text{number of predictors}}$ of selection at each split implying a lower $mtry$ decreases the chance of a splitting on the strong predictor. Splits on the weak or noisy predictors may not result in a meaningful increase in node purity and successive splits may result in impure terminal nodes that poorly predict the class of x_i in each tree. Such an ensemble may have highly unstable probability predictions.

Additionally, consider there is a class imbalance and the majority of observations are classified as A not B. The terminal nodes of each tree within an ensemble are more likely to contain the majority class. Consequently, there is a *majority class bias* as each tree in the ensemble is more likely to misclassifying an observation as an A because the terminal nodes have a higher proportion of A due to the higher proportion of A's in the data overall.

> **Note 7: Class Imbalance and Machine Learning**
>
> When there is a difference in the number of observations in each class, this is called class imbalance. It is important to note that majority class bias exists in conventional machine learning classification tasks. Bagging ensembles and random forest are well known to be sensitive to class imbalance meaning that class predictions are biased towards the majority (see Bader-El-Den, Teitei, and Perry 2019).
>
> However, the class imbalance problem is particularly notable when predicting probabilities. The probability prediction from a vote count method is very sensitive to a change in the votes from each tree. Suppose that balanced data results in $80/100$ trees classifying B as B and imbalanced data (more A than B) reduces correct classifications of B to $60/100$. This results in a 20% margin of error in probability estimates but the classification remains as B.

Individually, a low $mtry$ can lead to unstable probability predictions and class imbalance can create bias towards the majority class. But probability machines are particularly effected when there is both a low $mtry$ *and* class imbalance. Because successive noisy splits (relating to a low $mtry$) result in impure child nodes, the effects of majority class bias are exaggerated. Without the ability to separate the classes, the majority class will dominate terminal nodes. If the ensemble was able to split on informative covariates each time ($mtry$ is higher), then it should still be able to create pure splits even when there is some class imbalance. In other words, if there is a small class imbalance, reducing $mtry$ may reveal majority class bias not visible at higher $mtry$'s. Equally, if there is low $mtry$, then even a small class imbalance can lead to majority class bias.

As a general philosophy, ensemble learning methods based on classification trees are poor at predicting probabilities. If x_i has a known membership of A, and an unknown $\Pr_{\text{true}}(x_i = A) = 0.6$, the ensemble might classify x_i correctly 90% of the time leading to $\widehat{\Pr}(x_i = A) = 0.9$. As a probability machine, the ensemble has overestimated the probability by 30% even though a 90% classification accuracy is excellent. To predict $\Pr_{\text{true}}(x_i = A) = 0.6$, an ensemble needs to incorrectly classify

x_i in 40% of its trees. However, bagging ensembles and random forests are designed to maximise classification accuracy and there is no incentive for the model to intentionally achieve a specific misclassification rate that aligns with the true probability.

To exemplify these theoretical points, the National Supported Work (NSW) programme is a commonly discussed dataset in causal inference. The data results from a randomised controlled trial with 445 total participants, 185 in the program group, and 260 in the control group, so the true probability of treatment for each individual can be calculated as $185/445 = 0.42$ or 42%. Further information about this data is found in Appendix A. Randomisation should ensure that the probability of treatment is independent of the predictors and so all predictors should be noisy or weak. Although Figure 2.2 and Table 3.1 suggest some covariates do have a greater impact on the probability of participating in the programme, which echoes research by Smith and Todd (2005) who suggest randomisation is imperfect in the NSW dataset. Thus, the "best" calibrated probability model will have a distribution centred near 0.42 with some noise due to imperfect randomisation.

Figure 2.1 shows both divergence bias and majority class bias using `randomForest()` to fit both the random forest and bagging ensemble. Recall that a bagging ensemble is a random forest model when $mtry$ is equal to the number of predictors and so specifying `mtry = 7` in the `randomForest()` function fits a bagging ensemble. Additionally, logistic regression using the `glm()` function provides a meaningful comparison.

Figure 2.1 shows the logistic regression model has identified a central tendency and most propensities are between 0.25 and 0.75 which roughly aligns with the true probability. The bagging ensemble has clear evidence of divergence and the majority of predictions are outside 0.25 and 0.75 which is likely related to tree correlation. For the random forest with $mtry = 1$, a significant number of the treatment and control observations are centred near the control area ($T = 0$) with a wide range of other predictions. Recall that the control group is the majority class. Reducing $mtry$ from 7 to 1 reveals the majority class bias reinforcing the theoretical discussion that a combination of low $mtry$

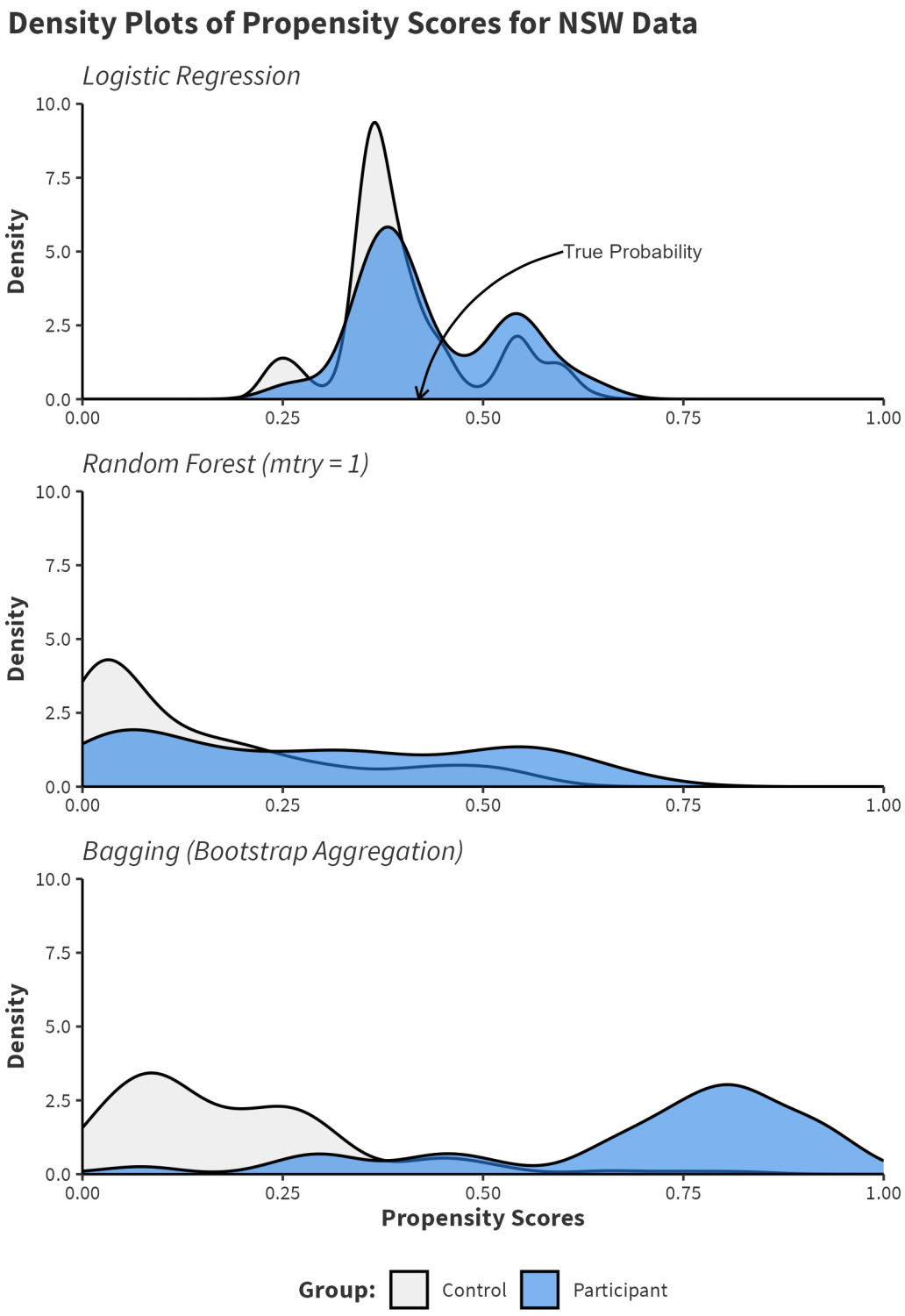

Figure 2.1: Compares the density estimates of the propensity scores for control and participant groups in the National Supported Work programme. `randomForest()` fits a random forest with `mtry = 1` and bagging ensemble with `mtry = 7`. The default values of `ntree = 500` and `nodesize = 1` are used. A logistic regression model is included for a comparison.

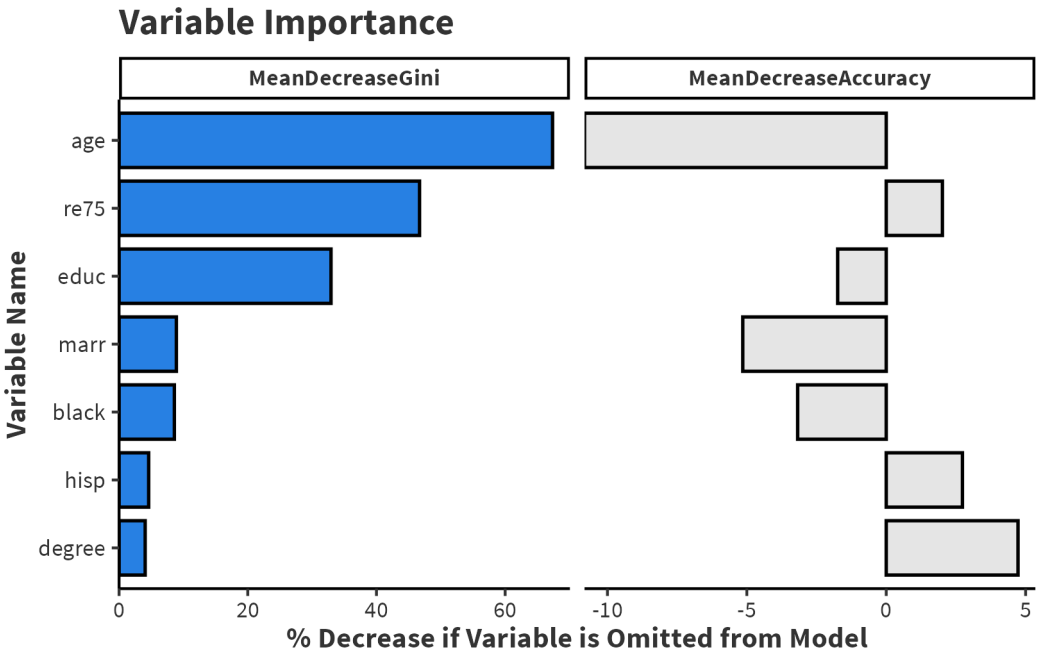

Figure 2.2: Compares the variable importance assigned to each variable from a bagging ensemble fitted on data from the National Supported Work programme. `randomForest()` fits a baggin ensemble with `mtry = 7` with default `ntree = 500` and `nodesize = 1`.

and class imbalance is especially troubling. The model over predicts the majority class and has unstable predictions otherwise. Both random forest and bagging ensembles have performed poorly compared to the true probability of 0.42%.

The tuning of $mtry$ faces double jeopardy and is another important area of discussion in probability machines. The selection of $mtry$ is typically carried out in with a classification loss function such as accuracy or out-of-bag error. Olson and Wyner (2018) compares tuning $mtry$ measured by classification accuracy and mean square error of known simulation probabilities and finds that the optimal value of $mtry$ for classification differs from probability prediction.[2] In other words, if a grid search finds that $mtry = 3$ is optimal for a classification task, this does not imply that $mtry = 3$ is optimal for predicting probabilities. Putting this together, $mtry$ is a double-edged sword, typically controlled with an imperfect method.

Random forests and bagging ensembles seem to be troubled as probability machines but this does not mean that bagging and random forest cannot perform well. In various simulation studies, they perform excellently as discussed in Section 2.3. Perhaps the nature of the data is informative for the potential success of a random forest or bagging ensemble.

Heuristically, divergence bias and majority class bias will most affect a probability machine when there is considerable overlap of true probabilities between groups. Recall the meaning of *common support* and *overlap* from Section 1.5. If there is overlap and a central region of true probabilities, then the effects of divergence bias may be very pronounced. Similarly, common support may make it even harder to increase purity in child nodes, as the covariates will lack clear split points. When combined with weak predictors relating to a low $mtry$, the terminal nodes of each tree may be relatively impure leading to a majority class bias. Alternatively, if true probabilities exist near 0 or 1 and there is a clear separation of class, divergence bias may trivially effect probability estimation as the probabilities already exist in that region. If there is a clear separation of class, then weak predictors

[2] Note that tuning $mtry$ for the mean square of probability prediction is only possible by design of the simulation study and is not possible in applications, as the true probability is unknown.

relating to a low $mtry$ may still create meaningful splits and pure terminal nodes. It is worth noting that propensity score methods require datasets with overlap to meet the assumptions required to determine causality.

2.2.3 Gradient Boosting Machines as Probability Machines

Moving beyond classification trees in random forests or bagging ensembles, Friedman (2001) introduced the *Gradient Boosting Machine* (GBM). A GBM sequentially constructs CART trees to correct errors made by previous trees. Employing a gradient descent process, each new tree is fit on the pseudo-residuals of the previous iteration. This means that with each iteration, the GBM takes a gradient step down the global loss function, incrementally minimizing the loss function to reach a minimum. A learning rate controls the contribution of each weak learner to the final model. By using a small learning rate, the machine learns slowly so that it can slowly descend the loss function. This allows for finer adjustments during the iterative process to better capture patterns in the data.

GBMs can be generalised to many different applications by minimizing a different loss function which can be specified as any continuously differentiable function. For binary outcomes, a GBM employs multiple regression trees and a logistic function to transform regression predictions into probabilities. Specifically, the logistic function used is:

$$\widehat{\Pr}(x_i = A) = \frac{1}{1 + \exp(-\mathsf{model}(x_i))}. \tag{2.3}$$

This logistic function is the same as in logistic regression, so a GBM with a binary class is sometimes called boosted logistic regression. The ensemble aims to minimise the Bernoulli deviance, which is equivalent to maximizing the Bernoulli log-likelihood with logistic regression. Maximizing the log-likelihood ensures that the predicted probability distribution is as close as possible to the true

probability distribution given the data. The full GBM model, $f_T(x)$ after T iterations can be written as:

$$f_T(x_i) = b_1(x_i) + \lambda \sum_{t=1}^{T} b_t(x_i). \tag{2.4}$$

Inside a base tree, each split considers all variables and makes the most informative split to descend the loss function using gradient descent. GBMs utilise many weak learners, such as a regression tree with a single split called a regression stump. However, additional splits enable the model to capture interactions between terms, which may increase probability calibration in complex or high-dimensional datasets.

By outputting probability predictions and avoiding the flaws of vote methods in other ensemble techniques, as well as allowing a probability distribution-based loss function optimal for probability prediction, GBMs stand out as a highly effective probability machine. Since GBMs predict probabilities from a logistic function, they avoid problems associated with a vote count method. The implementation and workflow to fit a GBM for propensity scores is discussed in Section 3.1.

Figure 2.3 shows the propensity scores resulting from a GBM model using the `gbm` package on the NSW data provides. A GBM is a notable performance improvement to random forest and bagging shown in Figure 2.1. Recall that a "better" model would predict probabilities near to 42% as this is the treatment/control share in the randomised NSW data.

2.2.4 Overfitting

Overfitting is a common concern when fitting machine learning models, as models can capture noise and random variations in the training data. An overfit model typically shows excellent performance on the training data but will perform poorly on new, unseen data because it cannot gen-

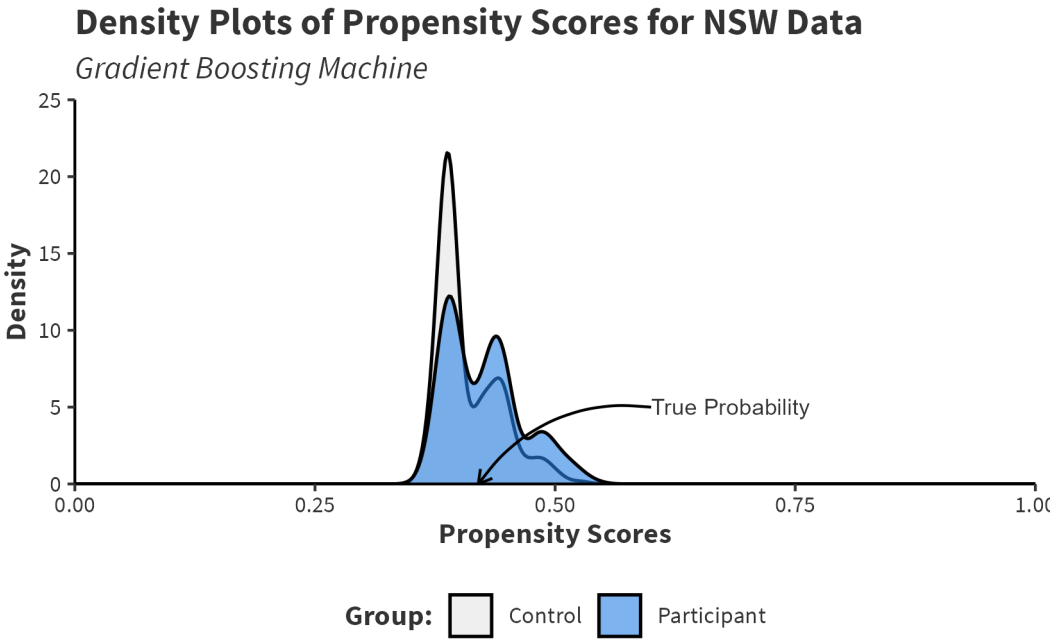

Figure 2.3: Density estimates of the propensity scores for control and participant groups in the National Supported Work programme using the `gbm` package with `distribution = "bernoulli"`, `data = nsw_data`, `n.trees = 10000`, and `shrinkage = 0.0001`.

eralise beyond the specific patterns of the training set. For instance, consider a machine learning algorithm used by a bank for fraud detection. In this scenario, an overfit model would struggle to classify transactions correctly as it has learned the noise and specific variation in the training data rather than the underlying patterns of fraud. Cross validation or test/train splitting can prevent overfitting to ensure a model can generalise to unseen data.

However, the model is not required to generalise a propensity score model as different datasets will have a different model. Instead, the emphasis of predicting propensity scores is to create balance in the data. A model is effective if it balances covariates between groups, even if it is overfit in a conventional sense.

> **Note 8: Overfitting in Logistic Regression**
>
> There is limited research on how overfitting a logistic regression model affects estimating treatment effects. In logistic regression, overfitting occurs when there are too many parameters and so the maximisation of the log-likelihood function is difficult because of noise. One study that investigates overfitting in this context is Schuster, Lowe, and Platt (2016), who suggest a general rule that the number of observations per parameter should be between 10 and 20. When overfitting occurs, the variance of the estimated treatment effect increases because noise amplifies the magnitude of the coefficients, resulting in a small bias towards 0 or 1 because of properties of the logit function. Specifically, when using (non-augmented) propensity score weighting, the estimate of the treatment effect will have high variance as propensity scores close to 0 or 1 receive artificially inflated weighting.

Lee, Lessler, and Stuart (2010) simulates a comparison of machine learning methods for propensity score prediction and finds that an overfit CART model performs better than a pruned CART model in terms of balance and treatment effect estimation bias. While not conclusive, this suggests that conventionally overfit trees are appropriate and potentially beneficial for propensity score modelling.

If overfitting was to occur, this could be interpreted as balance between groups getting worse decreases with a higher model complexity. Although various software packages use a stopping rule to prevent this. As conventional advice states, creating balance should be the aim of estimating propensity scores with overfitting being a minor concern.

2.3 Comparison of Machine Learning Algorithms: Simulation Results

A small body of simulation studies benchmarks probability machines for predicting propensity scores (see McCaffrey, Ridgeway, and Morral 2004; Setoguchi et al. 2008; Lee, Lessler, and Stuart 2010; Cannas and Arpino 2019; Tu 2019; Goller et al. 2020; Ferri-García and Del Mar Rueda 2020). Although these studies tackle the same problem, differences in simulation design and model implementation lead to a diverse range of perspectives on this issue. This variety reflects the complexity of the propensity score prediction.

Tu (2019) compares logistic regression, boosting, bagging, and random forests across different sample sizes, conditions of linearity and additivity, and treatment effect strengths. Boosting achieves the lowest bias ATE estimate in most scenarios and the lowest mean square error in all scenarios. Bagging ensembles and random forests perform poorly in both ATE estimate bias and MSE. The author notes that poor performance in bagging ensembles is likely due to correlated trees in the ensemble, leading to divergence bias. Random forests perform significantly better than bagging but both methods performed worse than boosting or logistic regression.

Despite poor theoretical properties as a probability machine, Lee, Lessler, and Stuart (2010) find that bagging results in the lowest standard error across many datasets.[3] This result is not surprising given that the bagging ensembles are trained on bootstrapped datasets, leading to lower variance

[3] In this case, the standard error is the dispersion of the standardised mean difference (effect size) across 1000 simulated datasets.

and standard error. Although, this advantage is not likely of practical interest given that the small performance gain in standard error is at the expense of a considerable increase of bias.

Additionally, Lee, Lessler, and Stuart (2010) finds that logistic regression performs well in simple data structures with comparable bias to boosting and random forest, but with larger standard errors. In complex data structures, boosting shows low bias and outperforms logistic regression while maintaining low standard errors. Consequently, the study concludes that boosted CART achieves the best 95% coverage in all simulation scenarios, with 98.6% coverage.[4]

Cannas and Arpino (2019) also undergo a simulation study to assess machine learning methods for propensity score prediction. They compare logistic regression, CART, bagging ensembles, random forest, boosting, neural networks, and naive bayes and find that random forest, neural networks, and logistic regression perform the best. Notably, the simulation design only performs hyperparameter tuning for CART, random forest, and neural networks but not either of their boosting implementation.[5] This is a weakness of their study design and thus their findings may be more informative of the relative performance of tuned versus untuned models. Although, the finding that random forest performs well when tuned is significant.

Goller et al. (2020) adds diversity to the simulation study literature by exploring an economics context, experimenting with imbalances between treated and control observations, and incorporating LASSO and probit models.[6] [7] Probit regression achieves the best covariate balance, with LASSO also performing well. In contrast, the random forest model performs poorly, showing imbalance

[4] In this context, the coverage is the proportion of times that the true treatment effect is within the 95% confidence interval across the number of simulations. This author implements 1000 simulations of each scenario.

[5] Cannas and Arpino (2019) provide a replication package for their simulation study online and their hyperparameter tuning is process transparent. The authors fit two GBMs using the `twang` and `gbm` package in R. The hyperparameter values provided to these untuned boosting models are contrary to heuristics and may lead boosting to perform poorly regardless of theoretical benefits discussed in Section 2.2.3.

[6] Goller et al. (2020) calculates the bias of the treatment effect using the average of the estimates from logistic regression, random forest, and LASSO models as the *true* treatment effect. Thus, the covariate balance table offers a clearer view of each method's performance.

[7] Tibshirani (1996) introduces LASSO regularization, short for Least Absolute Shrinkage and Selection Operator, is a technique used in linear regression to prevent overfitting by penalising the absolute size of the coefficients. It adds a penalty term to the ordinary least squares objective function, meaning that some coefficients may "shrink" to zero.

statistics with several orders of magnitude higher than those of probit or LASSO. To perform feature selection, a probit model with many interactions and polynomial terms is specified, and a LASSO penalty shrinks covariate coefficients to zero. Probit regression stands out for its superior covariate balance, while LASSO also delivers satisfactory results. The random forest model underperforms with significantly higher imbalance statistics compared to probit and LASSO.

Based on a review of the literature, the findings can be distilled into five important points:

1. Probability machines can predict propensity scores with excellent performance and their implementation should be considered in most scenarios. Although, a logistic regression approach may be preferred because of simplicity while still providing adequate performance in simple data structures.

2. In cases of non-linearity or non-additivity in the data, probability machines often achieve better covariate balance and lower bias of treatment effect estimates than logistic regression. This is significant as propensity scores are frequently used in observational studies with complex data structures (Rosenbaum and Rubin 1983).

3. Bagging ensembles perform poorly, a finding replicated across multiple studies.

4. Random forests can perform excellently when hyperparameters are satisfactorily tuned.

5. Further research should consider parametric methods with LASSO, Ridge, or Elastic Net penalties to assist in feature selection. Simulation study evidence for predicting propensity scores is limited despite attractive properties of these methods.

6. A tuned GBM stands out with strong theoretical support, excellent simulation performance, and superior software implementation and documentation. Specifically, this GBM will use the Bernoulli deviance as a loss function due to theoretical benefits. Implementations of GBMs such as AdaBoost.M1 have no simulation study evidence.

7. A good practical approach seems to be a trial-and-error approach of fitting multiple model specifications, then considering covariate balance for each model.

2.4 Code Provided for PDF Output

```
load(file = "globals.RData")
# Create Figure 2.1
library(randomForest)
library(patchwork)
library(ggplot2)
library(ragg)

theme_set(custom_ggplot_theme)

set.seed(88)
nsw_formula <- as.formula(as.factor(treat) ~ age + educ + re75 +
                          black + hisp + degree + marr)

logit_preds <- glm(nsw_formula, data = nsw_data,
                   family = binomial())$fitted.values

rf_mtry1_preds <- predict(randomForest(nsw_formula,
                          mtry = 1, data = nsw_data),
                          newdata = nsw_data, type = "prob")[, 2]
```

```r
bagging_model <- randomForest(nsw_formula, mtry = 7, importance = TRUE,
                              data = nsw_data)

bagged_preds <- predict(bagging_model, newdata = nsw_data, type = "prob")[, 2]

plot_pmachines <- function(pscores, plot_subtitle) {
  ggplot(nsw_data, aes(x = pscores, fill = factor(treat))) +
    geom_density(alpha = 0.6, linewidth = 0.6) +
    scale_fill_manual(values = c("#e5e5e5", "#2780e3"),
                      labels = c("Control", "Participant")) +
    labs(subtitle = plot_subtitle, x = "Propensity Scores", y = "Density",
         fill = "Group:") +
    scale_x_continuous(expand = expansion(0), limits = c(0,1)) +
    scale_y_continuous(expand = expansion(0), limits = c(0,10)) +
    custom_ggplot_theme
}

p1 <- plot_pmachines(logit_preds, "Logistic Regression") + xlab(NULL) +
  theme(legend.position = "none") +
  annotate(geom = "curve", x = 0.6, y = 5, xend = 0.42, yend = 0,
           curvature = .3, arrow = arrow(length = unit(2, "mm"))) +
  annotate(geom = "text", x = 0.6, y = 5, label = "True Probability",
           hjust = "left", color = "#333333", size = 3)

p2 <- plot_pmachines(rf_mtry1_preds, "Random Forest (mtry = 1)") + xlab(NULL) +
  theme(legend.position = "none")
```

```r
p3 <- plot_pmachines(bagged_preds, "Bagging (Bootstrap Aggregation)")


p1 / p2 / p3 + plot_annotation(
  title = "Density Plots of Propensity Scores for NSW Data", theme = custom_ggplot_theme)
# Create Figure 2.2
library(ggplot2)
library(tidyverse)

imp <- as.data.frame(importance(bagging_model))
imp <- cbind(vars = rownames(imp), imp)
imp <- imp[order(imp$MeanDecreaseGini),]
imp$vars <- factor(imp$vars, levels = unique(imp$vars))



imp %>%
  pivot_longer(cols = matches("Mean")) %>%
  ggplot(aes(y = vars, x = value, fill = name)) +
  geom_bar(stat = "identity", width = 0.8, show.legend = TRUE,
           position = position_dodge(width = 0.8), color = "black", linewidth = 0.6) +
  facet_grid(~ factor(name,
                      levels = c("MeanDecreaseGini", "MeanDecreaseAccuracy")),
             scales = "free_x") +
  scale_fill_manual(values = c("#e5e5e5", "#2780e3")) +
  scale_x_continuous(expand = expansion(c(0, 0.04))) +
  labs(title = "Variable Importance",
    x = "% Decrease if Variable is Omitted from Model", y = "Variable Name"
```

```
  ) + custom_ggplot_theme + theme(legend.position = "none")
# Create Figure 2.3
set.seed(88)
library(gbm)
nsw_gbm <- gbm(treat ~ age + educ + re75 + black + hisp + degree +
    marr, distribution = "bernoulli", data = nsw_data, n.trees = 10000,
    shrinkage = 0.0001)

boosted_preds <- predict(nsw_gbm, type = "response")

plot_pmachines(boosted_preds, "Gradient Boosting Machine") +
    scale_y_continuous(expand = expansion(0), limits = c(0, 25)) +
  annotate(geom = "curve", x = 0.6, y = 5, xend = 0.42, yend = 0,
           curvature = .3, arrow = arrow(length = unit(2, "mm"))) +
  annotate(geom = "text", x = 0.6, y = 5, label = "True Probability",
           hjust = "left", color = "#333333", size = 3) + labs(
  title = "Density Plots of Propensity Scores for NSW Data") +
  custom_ggplot_theme
save.image(file = "globals.RData")
```

3 Tutorial: Implimentation, Workflow, and Example with `WeightIt` and`gbm` in R

Based on Friedman (2001), the `gbm` package implements a *Generalized Boosting Machine*. Here, the "generalized" is because the package provides generalisations of the boosting framework to other distributions such as Bernoulli, Poisson, and Cox-proportional hazards partial likelihood of class probability predictions. Although this implementation very closely follows Friedman (2001) who introduced the gradient boosting machine. `gbm` also supports stochastic gradient boosting, which performs random bootstrap sampling for each tree using the `bag.fraction` parameter.

To fit and tune a GBM for propensity scores, wrapper packages facilitate optimal hyperparameter tuning for covariate balance. An effective approach involves fitting the model and computing balance statistics at each hyperparameter combination. Since the `gbm` package does not support this type of tuning, a wrapper package like `WeightIt` is necessary. `WeightIt` allows for hyperparameter tuning based on covariate balance and inverse propensity weighting (IPW). `WeightIt` supports hyperparameter turning of `shrinkage`, `interaction.depth`, and `n.trees`. Once the best model is identified, propensity scores are predicted inside `WeightIt`. These can be used inside `WeightIt` to perform IPW or extracted for other implementations. `WeightIt` also supports an offset meaning that logistic regression predictions are supplied to the `GBM` package.

Multiple sources, including package documentation and other research, suggest values for hyperparameters (see McCaffrey, Ridgeway, and Morral 2004; Ridgeway et al. 2024). A very low learning rate, such as 0.01 or 0.0005, allows a smooth descent of the loss function. The model should include a high number of trees, with $10,000$ or $20,000$ being a typical default value. While this may seem excessive, it is required when a low learning rate is used. A grid search process should consider many options including a very high number of trees and even though the optimal model may contain fewer trees. While GBMs often use shallow trees like stumps, allowing a few splits per tree can better model non-linearity and additivity. The `WeightIt` default allows for 3 splits. Based on anecdotal experience, 1 to 5 splits per tree is optimal, consistent with recommendations by McCaffrey, Ridgeway, and Morral (2004).

Another package, `twang`, proves functionality to tune the number of trees, but there are no inbuilt options for tuning of other hyperparameters and so accessory packages such as `caret` must be used. Although `twang` has other useful functionalities which users may wish to implement.

3.1 Hyperparameter Tuning and Workflow

The `WeigthtIt` package seems to have the best options for hyperparameter tuning and integration with a package for assessing balance called `cobalt`. The best information for this package can be found on this website or accessed with `vignette("WeightIt")` inside R after installation using `install.packages("WeightIt")`.

A workflow for hyperparameter tuning in `WeightIt` may be completed as follows:

1. Specify the `criterion` option or measure of balance, which specifies the measure of the *best model*. The available measures are any balance measure that `cobalt` can compute. A simple option to choose may be the average standardised mean difference (SMD) across all covariates

called `sdm.mean` or the smallest maximum SDM across covariates called `sdm.max`. It may be worthwhile to complete the tuning process with different tuning criteria.

2. Set the number of trees high. The package default is `n.trees = 10000` for binary treatments, but this may be too small depending on the learning rate. Typically, it is best to increase the number of trees to ensure slow learners have the opportunity to reach their minimum criterion. There is no modelling downside to a larger number of trees other than computation time as the model will predict propensity scores with a smaller `n.tree` if optimal.

3. Specify the grid search for the depth of the tree called `interaction.depth` and the learning rate called `shrinkage`. These values can be specified using `c()` such as `shrinkage = c(0.0005, 0.001, 0.05, 0.1, 0.2, 0.3)` or as integers such as `interaction.depth = 1:5`. These particular values are heuristically selected *suggestions* of good starting values. Additionally, an offset can be considered by performing a grid search across `offset = c(TRUE, FALSE)`.[1]

4. The model is fit and a grid search is performed. The tune grid and balance statistics can be retrieved with `my_weightit_object$info$best.tune`.

5. The best model should be inspected and to determine if the initial grid is appropriate. If the selection of the best model is at the boundary of a grid search, then a new grid should be created and step 3 and 4 are repeated. For example, if the initial fit is completed with `interaction.depth = 1:5` and the best fit is 5, then a new search can consider `interaction.depth = 3:7` so that the local area around 5 can be searched.

[1] In the context of gradient boosting machines (GBMs) or boosted logistic regression models, an offset refers to an additional term that is included in the model to account for a known effect or baseline value that should be factored into the prediction, but is not estimated by the model itself. In `gbm` the offset is estimated using conventional logistic regression.

6. Experiment with `bag.fraction`, which means each tree will consider a drawn proportion of observations equal to `bag.fraction`. Iteratively changing `bag.fraction` and assessing balance at each value should be practical. Consider 0.5, 0.67, and 1.

7. Assess balance of covariates and model fit. Covariate balance can be assessed with a balance table such as `bal.tab()` or visualised using `love.plot()` from `cobalt`.

8. The tuned model is stated and reported to allow replicable results. Balance tables are presented and discussed. Comparison to other methods of estimation if relevant.

9. Estimation and reporting of treatment effect.

3.2 Example: NSW Jobs Dataset Using R

For demonstration, propensity scores are estimated following the workflow discussed in Section 3.1 to estimate inverse propensity weights (IPW). The NSW jobs dataset arises from a randomised setting as described in Appendix A. Randomisation should eliminate structural differences between groups, but Rosenbaum and Rubin (1983) notes that randomisation only addresses structural balance and does not account for chance imbalance. To address this, propensity scores can mitigate any remaining chance imbalance, providing a more accurate estimate of the treatment effect. This example will include the fitting process of a GBM using `WeightIt` and a logistic regression model using `glm()`. Additionally, balance statistics will be computed leading to a robust estimate of the treatment effect.

> **Note 9: Inverse Probability of Treatment Weighting**
>
> Inverse probability of treatment weighting or inverse propensity weighting (IPW) adjusts for confounding in observational data by weighting individuals based on the inverse of their prob-

ability of receiving the treatment they actually got. This method creates a *pseudo-population* where treatment assignment is independent of observed covariates, similar to a randomised controlled trial. In this re-weighted population, the treatment and control groups should be have covariate balance, allowing for unbiased estimation of treatment effects. Essentially, IPW simulates random treatment assignment by rebalancing the sample, thereby eliminating confounding and enabling more accurate causal inferences.

3.2.1 Step 1-6: Model Fitting and Tuning

The `glm()` function will fit a conventional propensity score model with logistic regression in R. Logistic regression is performed by specifying the family to be the `binomial()`. Recall the `nsw_formula` is specified in Section 2.2.2.

```
nsw_logit_pmodel <- glm(nsw_formula, data = nsw_data,
                        family=binomial())                    ①


nsw_logit_pscores <- nsw_logit_pmodel$fitted.values           ②
```

1. Fits a logistic regression model using the `glm()` function specified to be a logistic model with `family=binomial()` using the previously created `nsw_formula`.

2. Extracts the fitted values (propensity scores) from the model.

The `weightit()` function from the `WeightIt` package will perform IPW and assign a weight to each observation such that the pseudo-population should exhibit covariate balance. The model object will be called `nsw_logit_weight`.

```
library(WeightIt)
nsw_logit_weight <- weightit(nsw_formula, data = nsw_data,                ①
                             ps = nsw_logit_pscores,                      ②
                             estimand = "ATE")                            ③
```

1. Specifies the formula and data.

2. Provides `weightit()` with the propensity scores from the logistic regression function. Note that in practice this can be completed within the `weightit()` function with `method = "glm"`. The separate estimation of the propensity scores is for illustrative purposes.

3. Specifies the estimand as the average treatment effect or ATE. For the purposes of demonstration, this is an arbitrary choice.

A GBM model for propensity scores can be specified using `method = "gbm"` inside the `weightit()` function. To ensure consistent results, running `set.seed(88)` will ensure each tree uses the same seed if `bag.fraction` less than 1. The model is fit using the heuristically suggested starting values. Note that this model may take approximately 30 second to fit as a grid search procedure is computationally intensive. Additionally, the best tuning specification is printed to assess if the initial tuning grid is appropriate.

```
set.seed(88)
nsw_boosted_weight <- weightit(nsw_formula, data = nsw_data,              ①
                               method = "gbm",                           ②
                               estimand = "ATE",
                               shrinkage = c(0.0005, 0.001, 0.05,
                                             0.1, 0.2, 0.3),             ③
                               interaction.depth = 1:5,
```

```
                                bag.fraction = 1,                    ④
                                offset = c(TRUE, FALSE),
                                criterion = "smd.mean",              ⑤
                                n.trees = 10000)
print(nsw_boosted_weight$info$best.tune)                             ⑥
```

1. Specifies the formula and data.
2. Specifies the propensity score prediction method to be a GBM and the estimand to the ATE.
3. Performs a grid search over these values of the learning rate and depth of tree.
4. Requires the model to use every observation in every tree, meaning the model will not perform stochastic gradient boosting. The function will will fit an offset and level GBM and select the specification with the best balance.
5. Defines the optimisation criteria to be the tune with the lowest average standardised mean difference (SMD). Additionally, the number of trees will be 10000 which is the package default.
6. Prints the tune details of the model with the best covariate balance.

```
  shrinkage interaction.depth distribution use.offset best.smd.mean best.tree
6       0.3                 1    bernoulli      FALSE    0.02253485      2392
```

The best balance across all tuning combinations yields an average SMD of 0.023 showing strong balance compared to the 0.1 threshold. Note averages can conceal extremes and a low average SMD does not mean all variables are balanced. A full balance table is presented in Section 3.2.2 accompanying a discussion of balance.

The best machine has a learning rate of 0.3 and contains 2392 decision stumps (trees with a depth of 1). The learning rate is on the boundary of the initial tuning grid showing that the tuning grid should be re-specified to include values near to 0.3. A reduction in the depth of tree and number of trees will reduce computation time.

The new tune grid will consider `shrinkage = c(0.25, 0.3, 0.35, 0.4, 0.45, 0.5)` as this allows the GBM to consider values between 0.2 and 0.3 and above 0.3 which were missing in the previous grid.

```
   shrinkage interaction.depth distribution use.offset best.smd.mean best.tree
11      0.45                 2    bernoulli      FALSE    0.01965492        95
```

Comparing the two iterations, there is a reduction from 0.022 to 0.02 in the SMD. The optimal tuning values are towards the centre of the tuning grid, implying that an adequate search of the local area has been completed. The best machine has a learning rate of 0.45, a tree depth of 2, and 95 trees. The learning rate is higher than expected, but this also explains why fewer trees are optimal.

Plotting the relationship between the number of trees and the average SMD is informative for the behaviour of the machine. Additionally, Figure 3.1 shows the optimal number of trees is highly variable. If the learning rate is set to `shrinkage = 0.05`, then the best balance is not achieved until near to $20,000$ trees.

For the optimal machine fit, finding that balance worsens as the number of trees increases is just as informative as knowing the correct number of trees. Provided sufficient computational performance, a wide grid search is beneficial in the long run to ensure that each model specification reaches the best balance possible.

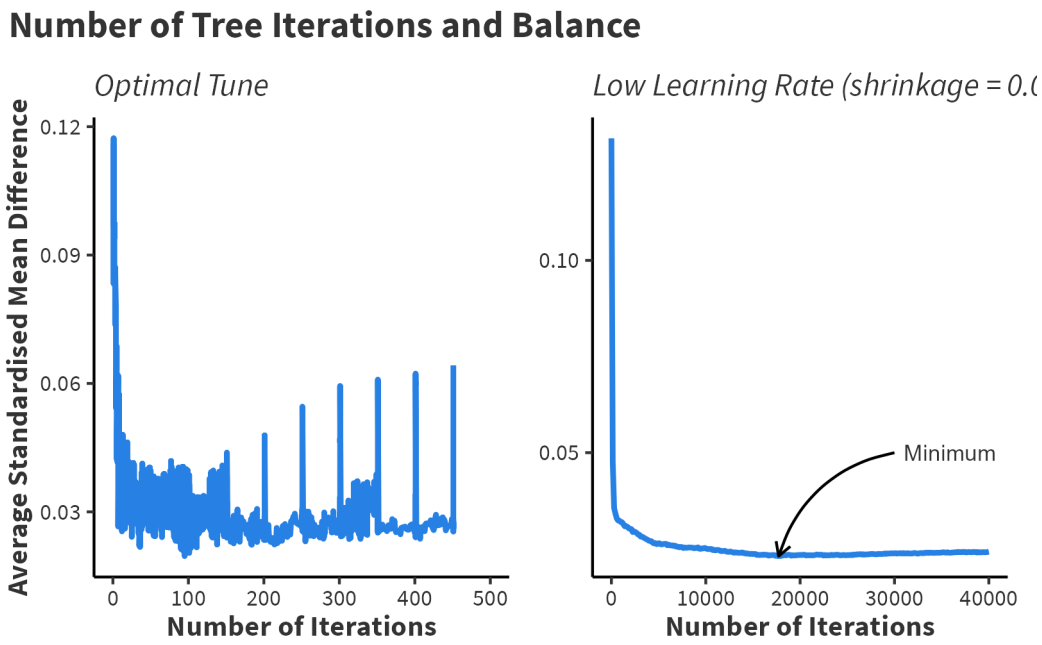

Figure 3.1: Relatoinship between standardised mean difference, number of interations, and learning rate in a GBM model. Please note the difference in horozontal scale between the two learning rates. The model is fit using `weightit` from the `WeightIt` package.

3.2.2 Step 7 and 8: Assessing Balance

> ⚠ The Importance of Discussing Balance
>
> Assessing balance is crucial because it ensures that the treated and control groups are comparable on observed covariates. This comparability is essential for reducing confounding and making valid causal inferences. Without proper balance, differences in outcomes between the groups could be due to pre-existing differences rather than the treatment itself. Balance assessment helps to verify that the propensity score model has effectively adjusted for covariates, creating a pseudo-randomised scenario. This step is vital for the reliability and validity of the study's conclusions. King and Nielsen (2019) notes that many papers that implement propensity score methods do not assess or report a balance in their studies, which can undermine the credibility of the research process and make it hard for readers to understand why results are robust.
>
> A good resource of information for assessing balance is documentation from the `cobalt` package, which can be viewed by running `vignette("cobalt", package = "cobalt")` in R.

As stated, `cobalt` provides very good integration with other related packages such as `WeightIt` for IPW and `MatchIt` for propensity score matching. Balance tables are created using `bal.tab()`.

```
library(cobalt)                                                          ①
nsw_logit_btab <- bal.tab(nsw_logit_weight,                              ②
                          data = nsw_data,
                          stats = c("mean.diffs","variance.ratios"),     ③
                          binary = "std", continuous = "std",
                          thresholds = c(mean.diffs = 0.1))              ④

nsw_logit_btab <- nsw_logit_btab$Balance[-1,-c(2,3)]                     ⑤
```

1. Loads the `cobalt` package. This assumes the package is already installed with `install.packages("cobalt")`
2. Uses the `bal.tab()` fucntion to create balance statistics for the previously created `nsw_logit_weight` model.
3. Specifies the calculation of standardised mean differences and variance ratios for each covariate. The mean differences will be standardised for binary and continuous variables.
4. Sets a threshold of balance to be 0.1 to determine if a covariate is balanced.
5. Extracts the balance table of the `nsw_logit_btab` object and removes excessive columns. This is only completed for ease of visualisation and is not typically required.

Additionally, `bal.tab()` will create balance tables for the GBM method's IPWs and the raw data. For presentation, `dplyr` combines each of the individual balance tables for presentation using `kable` and `kableExtra`.

Table 3.1 shows that both logistic regression and the GBM have reduced imbalance. The raw data exhibits imbalance across age, years of education, if someone is gispanic, and if someone has a bachelors degree. Imbalanced datasets leads to biased treatment effect estimation so the estimate of the treatment effect in the raw data may be biased. In this example, logistic regression appears to achieve the best covariate balance although GBM achieves slightly better variance ratios.

3.2.3 Step 9: Results

Finally, the treatment effect can be estimated using `lm_weightit()` from the `WeightIt` package and `avg_comparisons()` from the `marginaleffects` package. Note that the outcome variable is `re78` which is real income in 1978 meaning that the income is adjusted for inflation. Previously, the

Table 3.1: Standardised mean difference (a measure of balance) across different covariates in the National Supported Work data. The values are categorised for different propenensity score methods allowing a comparison. Balance tables are computed using `bal.tab()` from the `cobalt` package.

Variable	Type	SMD	Balanced	Variance Ratio
Raw Data				
Age	Contin.	0.1066	No	1.0278
Education	Contin.	0.1281	No	1.5513
Income 1975	Contin.	0.0824	Yes	1.0763
Black	Binary	0.0449	Yes	NA
Hispanic	Binary	-0.2040	No	NA
Degree	Binary	0.2783	No	NA
Married	Binary	0.0902	Yes	NA
Logistic Regression and IPTW				
Age	Contin.	-0.0001	Yes	0.9809
Education	Contin.	0.0012	Yes	1.2725
Income 1975	Contin.	0.0081	Yes	0.7971
Black	Binary	0.0006	Yes	NA
Hispanic	Binary	-0.0031	Yes	NA
Degree	Binary	0.0009	Yes	NA
Married	Binary	0.0045	Yes	NA
Boosting Machine and IPTW				
Age	Contin.	-0.0065	Yes	0.9086
Education	Contin.	0.0220	Yes	1.1391
Income 1975	Contin.	-0.0152	Yes	1.0134
Black	Binary	0.0028	Yes	NA
Hispanic	Binary	-0.0547	Yes	NA
Degree	Binary	0.0481	Yes	NA
Married	Binary	0.0085	Yes	NA

treatment indicator was the outcome variable because the propensity scores are a prediction of the treatment indicator.

```
nsw_boosted_lm <- lm_weightit(re78 ~ treat * (age + educ + re75 + black +   ①
                              hisp + degree + marr), data = nsw_data,
                              weights = nsw_boosted_weight$weights)         ②

library(marginaleffects)                                                    ③
nsw_boosted_result <- avg_comparisons(nsw_boosted_lm, variables = "treat")
```

1. Uses `lm_weightit()` to compute pseudo-outcomes. The formula here specifies an interaction between the treatment and all other variables. Note that $*$ indicates multiplication in R.

2. Specifies the `weights` from the `nsw_boosted_weight` object created earlier by the `weightit()` function. Intuitively, this is performing linear regression using the pseudo-population, where the pseudo-population is created weighting the data by `nsw_boosted_weight$weights`.

3. Estimates a comparison between the potential outcomes as well as standard errors for inference.

Additionally, this process is followed for the logistic regression propensity scores and the results are combined in to a table for comparison.

Table 3.2: Comparison of estimates of the average treatment for the National Supported Work data.

	Estimate	SE	P.Value	Lower.CI	Upper.CI
Logistic Regression	1610.786	668.4870	0.0160	300.5756	2920.997
GBM	1609.947	669.4201	0.0162	297.9081	2921.987

Table 3.2 shows that both estimates of the treatment effect are nearly identical at $\$1610$ with logistic regression inferring a $\$0.86$ larger treatment effect. Additionally, these results are statistically significant at the 5% level with nearly identical standard errors.

3.3 Code Provided for PDF Output

```
load(file = "globals.RData")
library(WeightIt)
nsw_logit_weight <- weightit(nsw_formula, data = nsw_data,                    ①
                             ps = nsw_logit_pscores,                          ②
                             estimand = "ATE")                                ③
# Additional weightit() GBM grid
set.seed(88)
nsw_boosted_weight2 <- weightit(nsw_formula, data = nsw_data,
                                method="gbm",
                                estimand = "ATE",
                                shrinkage= c(0.25, 0.3, 0.35, 0.4, 0.45, 0.5),
                                interaction.depth = 1:3,
                                bag.fraction = 1,
                                offset = c(TRUE, FALSE),
                                criterion = "smd.mean",
                                n.trees = 5000)

print(nsw_boosted_weight2$info$best.tune)
# Create Figure 3.1
```

```r
library(ggplot2)
library(patchwork)
low_shrinkage <- weightit(nsw_formula, data = nsw_data,
                                method = "gbm",
                                estimand = "ATE",
                                shrinkage = 0.05,
                                interaction.depth = 1,
                                offset = c(TRUE, FALSE),
                                criterion = "smd.mean",
                                n.trees = 40000)


optimal_boost_plot <- ggplot(nsw_boosted_weight2$info$tree.val,
                             aes(x = tree, y = smd.mean)) +
  geom_line(linewidth = 1, color = "#2780e3") +
  labs(subtitle = "Optimal Tune",
       x = "Number of Iterations",
       y = "Average Standardised Mean Difference") +
  xlim(0,500) + custom_ggplot_theme


lowshrinkage_boost_plot <- ggplot(low_shrinkage$info$tree.val,
                                  aes(x = tree, y = smd.mean)) +
  geom_line(linewidth = 1, color = "#2780e3") +
  labs(subtitle = "Low Learning Rate (shrinkage = 0.05)",
       x = "Number of Iterations",
       y = NULL) +
  annotate(geom = "curve", x = 30000, y = 0.05,
```

```
            xend = low_shrinkage$info$best.tree, yend = 0.0231,
            curvature = 0.3, arrow = arrow(length = unit(2, "mm"))) +
  annotate(geom = "text", x = 31000, y = 0.05, label = "Minimum",
            hjust = "left", color = "#333333", size = 3) + custom_ggplot_theme

optimal_boost_plot + lowshrinkage_boost_plot + plot_annotation(
  title = 'Number of Tree Iterations and Balance', theme = custom_ggplot_theme)
# Create balance tables
nsw_boosted_btab <- bal.tab(nsw_boosted_weight,
                            data = nsw_data,
                            stats = c("mean.diffs","variance.ratios"),
                            binary = "std", continuous = "std",
                            thresholds = c(mean.diffs = 0.1))

nsw_raw_btab <- bal.tab(nsw_formula,
                        data = nsw_data,
                        stats = c("mean.diffs","variance.ratios"),
                        binary = "std", continuous = "std",
                        thresholds = c(mean.diffs = 0.1),
                        s.d.denom = "treated")

nsw_boosted_btab <- nsw_boosted_btab$Balance[-1, -c(2, 3)]
nsw_raw_btab <- nsw_raw_btab$Balance[-c(5, 6)]
# Create Table 3.1
library(tidyverse)
library(kableExtra)
```

```r
collabels <- c("Type", "SMD", "Balanced", "Variance Ratio","Method")

rowlabels <- c("Age", "Education", "Income 1975","Black",
               "Hispanic", "Degree", "Married")

nsw_raw_btab$method <- "Raw Data"
nsw_logit_btab$method <- "IPTW: Logistic Regression"
nsw_boosted_btab$method <- "IPTW: Boosting"

combined_btab <- bind_rows(setNames(nsw_raw_btab,collabels),
                           setNames(nsw_logit_btab,collabels),
                           setNames(nsw_boosted_btab,collabels))

combined_btab$Variable <- rep(rowlabels,3)

combined_btab <- combined_btab[c(6,1,2,3,4,5)]

rownames(combined_btab) <- NULL

combined_btab$Balanced <- ifelse(
          combined_btab$Balanced == "Not Balanced, >0.1", "No", "Yes")

kbl(combined_btab[-6], digits = 4, booktabs = TRUE, align = "c",
      font_size = 10) %>%
  kable_styling(full_width = T) %>%
```

```
  row_spec(0, bold = TRUE) %>%
  column_spec(1, bold = TRUE) %>%
  column_spec(2:5, bold = F, width = "3cm") %>%
  pack_rows("Raw Data", 1, 7, label_row_css = "text-align: center;") %>%
  pack_rows("Logistic Regression and IPTW", 8, 14,
            label_row_css = "text-align: center;") %>%
  pack_rows("Boosting Machine and IPTW", 15, 21,
            label_row_css = "text-align: center;")
# Create Table 3.2
nsw_logit_lm <- lm_weightit(re78~treat*(age + educ +
                            re75 + black + hisp +
                            degree + marr), data = nsw_data,
                            weights = nsw_logit_weight$weights)

nsw_logit_result <- avg_comparisons(nsw_logit_lm, variables = "treat")

nsw_comparisons_tab <- rbind(extract_comparison_results(nsw_logit_result),
                            extract_comparison_results(nsw_boosted_result))
rownames(nsw_comparisons_tab) <- c("Logistic Regression", "GBM")

knitr::kable(nsw_comparisons_tab, digits = 4)
save.image(file = "globals.RData")
```

4 Replication Case Study

The replication study focuses on a paper titled *The Impact of Coffee Certification on Small-Scale Producers' Livelihoods: A Case Study from the Jimma Zone, Ethiopia* published in Agricultural Economics (2012) by Pradyot Ranjan Jena, Bezawit Beyene Chichaibelu, Till Stellmacher, and Ulrike Grote. This paper explores the effects of fair trade coffee certification schemes on the economic wellbeing of small-scale coffee farmers in Ethiopia, particularly examining whether these schemes contribute to poverty reduction and improved livelihoods among smallholders.

The central theme of the paper is the evaluation of certification schemes, such as Fairtrade and organic certification, as tools for enhancing the income stability and economic resilience of small-scale coffee producers. Certification is seen as a potential tool for economic growth and and environmental sustainability and so it is important to understand the impact on small-scale farmers. Table 4.1 summarises the variables used in the propensity score model.

Table 4.1: List of covaraites in Jena et al. (2012). Per capita income is the outcome and certification is the treatment. Other covariates are used in the propensity score model.

Name	Codename	Description
Per Capita Income	`percapitaincome_dry_eq1`	Average income earned per person within a farming household

Table 4.1: List of covaraites in Jena et al. (2012). Per capita income is the outcome and certification is the treatment. Other covariates are used in the propensity score model.

Name	Codename	Description
Certification (Treatment/ Control)	`certified`	If the farming household is certified (=1) or otherwise (=0)
Household Age	`age_hh`	Age of the head of the household in years
Squared Household Age	`agesq`	Age of the head of the household squared
Gender	`gender`	Gender of the head of household (male = 1 and female = 0)
Dependency Ratio	`depratio`	Household members below 14 and above 65 years divided by rest of the household member
Education Level	`edu`	Education of the head of household in years
Years of Coffee Production	`years_cofeproduction`	Years of experience in coffee farming
Log Total Land	`logtotal_land`	Logarithm of total land size in hectares
Access to Credit	`access_credit`	Household has access to credit (yes = 1, otherwise = 0)
Bad Weather	`badweat`	If the household was affected by floods/droughts during the last year (2008–2009)

Non-farm Income Access	`nonfarmincome_access`	if the household has access to nonfarm income

Jena et al. (2012) define livelihood as a combination of per capita income, total income, per capita consumption, and yield per hectare. For simplicity, this replication will only use per capita income as a dependent variable. This measure is selected as per capita income is a direct measurement of the income of those potentially impacted by certification. Additionally, the replication uses the same variables as the original paper so any difference in estimates or covariate balance can be attributed to the propensity score model.

Randomisation into certified and uncertified is not possible and it is likely that farmers who seek certification are different than farmers who don't. Thus, there is selection bias leading to structural differences between groups so a contrast in means between the certified (treated) and uncertified (control) farmers would be biased. Propensity scores are used to create covariate balance and reduce bias of the estimated treatment effect. The paper did not assess the balance of covariates. However, this provides a good opportunity to assess covariate balance in the initial paper and the repeat the analysis using a machine learning propensity model.

4.1 Replication of Original Results

Jena et al. (2012) provides a replication package including Stata code that uses Stata's psmatch2 package to perform nearest neighbour matching with replacement and common support trimming. Common support trimming means that any observations outside the commonly overlapping are are discarded. The results of the paper are be fully replicated using the `MatchIt` package inside R.

Table 4.2: Replication of results in Jena et al. (2012). Note a slight difference in standard error which should be 1.1 as the `MatchItSE` package uses a trivially different method than psmatch2 in Stata (see Abadie and Imbens 2006). Matching is performed by `matchit()` from the `MatchIt` package.

	Estimate	SE	P.Value	Lower.CI	Upper.CI
Replicated Result	-0.1538	0.9898	0.835	-1.6009	1.2934

Table 4.2 shows the replicated result obtained by Jena et al. (2012). The intriguing finding of the paper is that the average treatment effect on the treated (ATT) is negative. That is, of the farmers that become certified, their per capita income is expected to decrease by $\$0.15$ per day. Intuition and proponents of certification schemes suggest that certification leads to an increase of income. If certification negatively impacts income, it would call into question a significant effort to engage in certification and fair trade practices.

Jena et al. (2012) does not perform any discussion or consideration of balance in their paper and so it is unclear if propensity score matching results in covariate balance. The `cobalt` package creates balance tables using `bal.tab()` and a visualisation using `love.plot()`.

Table 4.3: The standardised mean difference (SMD) for each covariate in Jena et al. (2012) using a logistic regression propensity model and propensity score matching. Across each of the covariates, a balance threshold is set at 0.1 to indicate if a covariate is balanced. Binary and continuous variables are both standardised over the treatment group. SMDs are computed using `bal.tab()` from the `cobalt` package.

	Type	Diff.Adj	M.Threshold	V.Ratio.Adj
Household Age	Contin.	-0.2723	No	1.0728
Squared Household Age	Contin.	-0.2552	No	1.1430
Non-farm Income Access	Binary	0.3005	No	NA
Log Total Land	Contin.	0.2597	No	1.2969

Table 4.3: The standardised mean difference (SMD) for each covariate in Jena et al. (2012) using a logistic regression propensity model and propensity score matching. Across each of the covariates, a balance threshold is set at 0.1 to indicate if a covariate is balanced. Binary and continuous variables are both standardised over the treatment group. SMDs are computed using `bal.tab()` from the `cobalt` package.

	Type	Diff.Adj	M.Threshold	V.Ratio.Adj
Dependency Ratio	Contin.	-0.3996	No	0.9789
Bad Weather	Binary	0.2016	No	NA
Education Level	Contin.	0.2443	No	1.0338
Gender	Binary	-0.1324	No	NA
Years of Coffee Production	Contin.	-0.3400	No	0.9111
Access to Credit	Binary	0.1949	No	NA

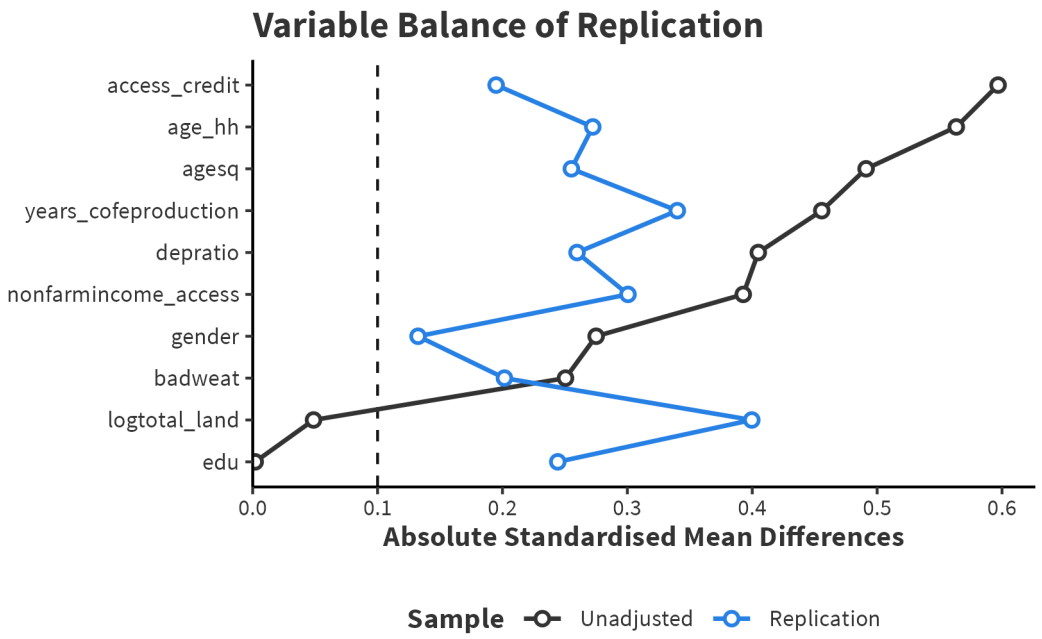

Figure 4.1: A visual representation of Table 4.3 called a love plot. Additionally, the unadjusted (raw data) SMDs are displayed for comparison. Variables are ordered by the SMD in the unadjusted data. Plot is created using `love.plot()` from the `cobalt` package.

Table 4.3 and Figure 4.1 show that propensity score matching has obtained very poor balance. Based on the 0.10 rule discussed in Section 2.1.1, not a single variable is balanced and so the estimate of the treatment effect is likely to be biased by structural differences between control and certified.

Four key variables: *age*, *gender*, *education*, and *access to credit* all exhibit poor balance. These variables are strong confounders in theory and so emphasising balance in these variables is critical to making a robust causal inference. Perhaps there is gender or age discrimination in the certification process. Perhaps, those with lesser education may struggle to obtain certification. Perhaps those who have less access to credit are unable to afford to become certified. Moving forward, these variables must exhibit better covariate balance to make a robust conclusion.

Figure 4.2 shows the effect of common support trimming. Table 4.4 shows 34 total observations are dropped of which 33 are treated and 1 are control. By dropping these observations, PSM avoids making poor matches which should lead to better covariate balance. When observations are discarded, the estimand is no longer the ATT. Instead, it is refereed to as the average treatment effect on the matched or ATM. There is a significant reduction in the effective sample size in the control group from 82 to 21 individuals.

Table 4.4: The effective sample size resulting from the use of propensity score matching in Jena et al. (2012). The effective sample size (ESS) is displayed in the unweighted (raw) and matched data as well as the number of disarded observations. Computed using `bal.tab()` from the `cobalt` package.

	Control	Certified
All (ESS)	82	164
All (Unweighted)	82	164
Matched (ESS)	21	131
Matched (Unweighted)	42	131
Unmatched	39	0

Table 4.4: The effective sample size resulting from the use of propensity score matching in Jena et al. (2012). The effective sample size (ESS) is displayed in the unweighted (raw) and matched data as well as the number of disarded observations. Computed using `bal.tab()` from the `cobalt` package.

	Control	Certified
Discarded	1	33

Overall, the propensity score matching in Jena et al. (2012) is poor and results in unbalanced covariates and a loss of estimand.

4.2 Further Modelling

To improve the poor balance achieved by the Jena et al. (2012), there are two strategies to obtain better balance. First, the propensity scores can be re-estimated using machine learning to obtain better calibrated propensity scores. Second, inverse propensity weighting (IPW) can be used instead of propensity score matching (PSM). IPW should ensure that the sample size remains the same as no observations are lost through a matching process. IPW should retain all observations and preserve the estimand as the ATT. Additionally, IPW is generally more efficient as a pseudo-population is based on precise weights compared to matched observations based on approximate similarity.

The machine learning propensity scores will be estimated using the `WeightIt` package in the same process as Chapter 3. The model will be used using `criterion = "smd.mean"` for simplicity.

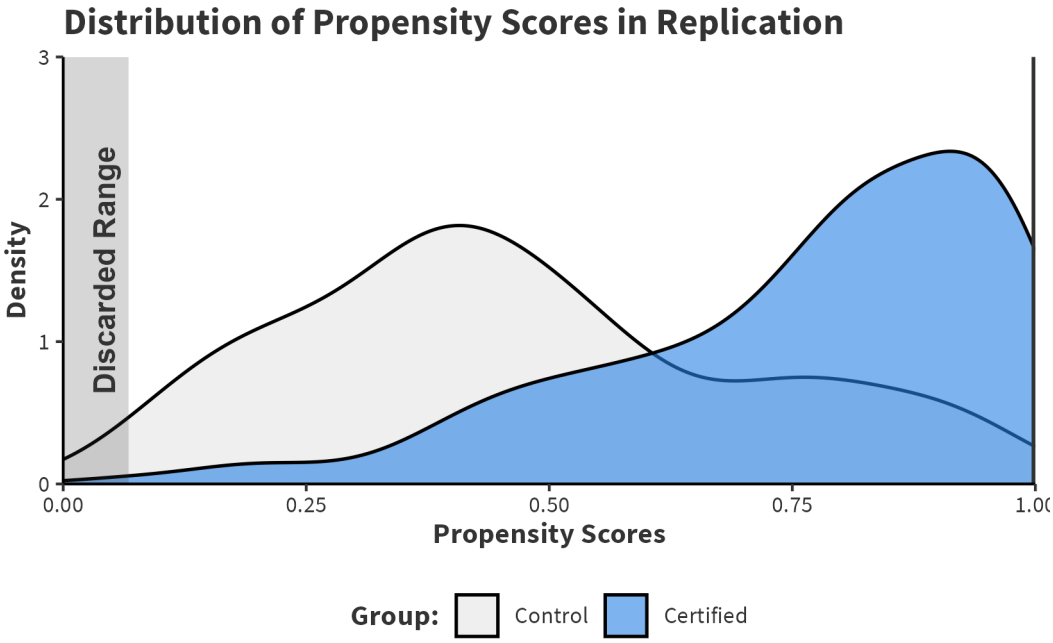

Figure 4.2: Density estimates of the propensity scores from Jena et al. (2012) using logistic regression. Propensity score matching discards some observations as displayed by the discarded range on the left. A single observation is discarded on the right.

> **Note 10: Discussion of Tuning**
>
> Initially, a tuning grid considering shrinkage values of $0.001, 0.005, 0.01, 0.05, 0.1, 0.2, textand0.3$ were considered using 10000 trees with a depth between 1 and 5. The best tuning performance was found with shrinkage of 0.2 and 13 trees which were three splits 2 deep.
>
> As such my later tuning grids considered higher learning rates and a smaller number of trees. The third and final iteration of the tuning grid searches between $0.15, 0.2125, 0.275, 0.3375,$ and 0.4, between 3 and 6 splits deep, uses an offset, and randomly selects 67% of the data at each tree.

Of course there is no guarantee that the GBM model will perform the best and so a logistic model is also fitted. An interesting comparison is between the SMDs in the matched data and in the weighted sample. Any differences between the two samples relates to the difference between PSM and IPW as the propensity scores are identical.

4.3 Comparison of Methods

As before, `cobalt` creates a balance table and love plot.

There are three notable findings:

1. PSM has performed very poorly relative to IPW even when matching dropps a significant number of observations.

2. A GBM model has resulted in better covariate balance than logistic regression for most covariates. Using a 0.1 guidline for determining balance, logistic regression leaves 5 variables unbalanced and the GBM leaves 3 variables unbalanced. Additionally, the degree of unbalance is larger for logistic regression.

Table 4.5: Comparison of standardised mean difference (SMD) using different propensity score models. Across each of the covariates, a balance threshold is set at 0.1 to indicate if a covariate is balanced. Binary and continuous variables are both standardised over the treatment group. SMDs are computed using `bal.tab()` from the `cobalt` package.

Variable	Type	SMD	Balance Threshold	Variance Ratio
Raw Data				
Household Age	Contin.	0.5634	No	0.8650
Squared Household Age	Contin.	0.4912	No	1.0070
Non-farm Income Access	Binary	-0.3928	No	NA
Log Total Land	Contin.	-0.4048	No	0.5507
Dependency Ratio	Contin.	0.0487	Yes	1.2371
Bad Weather	Binary	-0.2505	No	NA
Education Level	Contin.	-0.0020	Yes	0.7272
Gender	Binary	-0.2750	No	NA
Years of Coffee Production	Contin.	0.4557	No	1.3621
Access to Credit	Binary	0.5968	No	NA
Logistic Regression and IPTW				
Household Age	Contin.	0.2449	No	0.9275
Squared Household Age	Contin.	0.2277	No	1.0724
Non-farm Income Access	Binary	0.1701	No	NA
Log Total Land	Contin.	-0.0923	Yes	0.8564
Dependency Ratio	Contin.	0.1138	No	1.3877
Bad Weather	Binary	0.1941	No	NA
Education Level	Contin.	0.0473	Yes	0.9217
Gender	Binary	-0.0465	Yes	NA
Years of Coffee Production	Contin.	-0.0613	Yes	1.1125
Access to Credit	Binary	-0.0292	Yes	NA
Boosting Machine with IPTW				
Household Age	Contin.	0.0669	Yes	1.2690
Squared Household Age	Contin.	0.0989	Yes	1.4911
Non-farm Income Access	Binary	0.0579	Yes	NA
Log Total Land	Contin.	-0.0284	Yes	0.8760
Dependency Ratio	Contin.	-0.0623	Yes	0.7660
Bad Weather	Binary	0.1915	No	NA
Education Level	Contin.	0.1377	No	1.0735
Gender	Binary	-0.0816	Yes	NA
Years of Coffee Production	Contin.	-0.0055	Yes	0.9704
Access to Credit	Binary	0.1231	No	NA

Variable	Type	SMD	Balanced	Variance Ratio
Raw Data				
Age	Contin.	0.1066	No	1.0278
Education	Contin.	0.1281	No	1.5513
Income 1975	Contin.	0.0824	Yes	1.0763
Black	Binary	0.0449	Yes	NA
Hispanic	Binary	-0.2040	No	NA
Degree	Binary	0.2783	No	NA
Married	Binary	0.0902	Yes	NA
Logistic Regression and IPTW				
Age	Contin.	-0.0001	Yes	0.9809

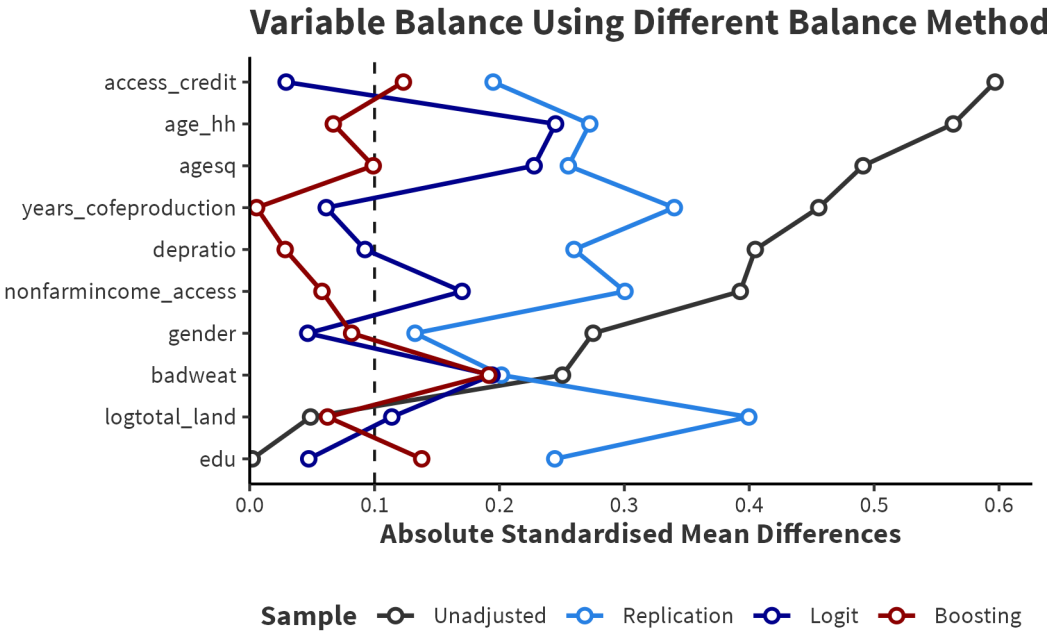

Figure 4.3: Visual representation of Table 4.5 called a love plot and displays the standardised mean difference (SMD) of covariates in Jena et al. (2012). Additionally, the unadjusted SMDs are displayed for comparison. Variables are ordered by the SMD in the unadjusted data.

3. Logistic regression has a satisfactory average SMD of 0.077. Boosting has an average SMD of 0.0498 which is excellent and narrowly meets a rigorous threshold of 0.05.

4. The covariate with the highest SMD is *household age* (0.245) in logistic regression and *bad weather* (0.191) in the GBM.

4.4 Results

Now that satisfactory covariate balance is achieved, the treatment effect can be estimated under logistic regression, the GBM, and then compared to the result in the paper. Note that the estimand in the paper is intended to be the average treatment effect (ATT) but dropped observations mean the actual treatment effect is the average treatment effect on matched (ATM) individuals. In theory, better covariate balance should lead to a better estimate of the ATT so a comparison of the estimates is interesting. As in Section 3.2.3, the results will be completed using G-computation with the `lm_weightit()` and `avg_comparisons()` functions.

Table 4.6: Estimates of the average treatment effect on the treated of certification on per capita income across different propensity score models and methods. Created using `WeightIt`, `MatchIt`, and `Cobalt` packages.

	Estimate	SE	P.Value	Lower.CI	Upper.CI
Rep. Result (Logistic with PSM)	-0.1538	0.9898	0.8350	-1.6009	1.2934
Logistic Regression and IPW	-1.5824	0.6072	0.0092	-2.7724	-0.3924
Generalized Boosting Machine and IPW	-1.0187	0.5196	0.0499	-2.0372	-0.0003

Table 4.6 shows the estimates of the treatment effect across different methods. Recall that Jena et al. (2012) estimate a an effect of -0.15 implying that daily income reduces by $\$0.15$ if a farmer becomes certified. This result is not statistically significant.

The IPW estimate is -1.58 implying that certification leads to a $\$1.58$ decrease in daily income. This coefficient is much larger than than the original paper by a magnitude of 10. Additionally, this estimate is statistically significant at the 1% level. The GBM estimate is -1.02 which predicts a decrease in daily income by $\$1.02$ when a farmer becomes certified. This finding is statistically significant at the 5% level.

The most interesting result is that the estimates become even more negative. One may expect that the result from a better balanced sample would become positive to align with theoretical motivations for certification policies. Jena et al. (2012) presented two explanations for why certification shows no positive impact. First, the authors note that the prices offered by certified cooperatives are not significantly different from those provided by non-certified cooperatives. Second, a substantial portion of coffee—about 75% is sold to private traders, who often pay higher prices to non-certified farmers. Additionally, from qualitative interviews with farmers, the authors note that policies and arrangements within different cooperatives exhibit heterogeneity so the impact of certification may relate more to the structure of the cooperatives not merely being certified.

The reason for a large difference is twofold. First, better covariate balance by using a GBM and IPW and should result in a more robust estimate. Of course better covariate balance alone does not guarantee robust results but it is a step in the right direction. Second, weighting on the inverse of the propensity scores instead of matching may significantly effect the estimate of the treatment effect especially when matching results in dropped observations.

An additional answer is the impact of reverse causality. A general problem is causal inference is that the direction of causality is not always known. While it is most intuitive that coffee certification would impact income, it is also possible that per capita income might determine their certification. Suppose that proponents of fair trade and certification are correct that certification will increase income and benefit livelihood. If farmers are aware of this, then perhaps the lowest income farmers are most likely to attempt to become certified to increase their income. Additionally, income likely

has a reverse causal relationship with many of the explanatory variables. For example, a higher income may lead to better access to credit and the accumulation of land.

In summary, the analysis demonstrates that using more advanced methods like GBM and IPW not only improves covariate balance but also leads to significantly larger and more negative estimates of the treatment effect compared to the original study. This suggests that previous estimates may have underestimated the negative impact of certification on per capita income. The findings highlight the importance of methodological rigor in estimating causal effects and raise critical questions about the broader implications of certification policies, particularly when considering potential reverse causality and the varying structures of cooperatives. This analysis underscores the need for careful interpretation of treatment effects, especially in policy-relevant research.

4.5 Code Provided for PDF Output

```
load(file = "globals.RData")
# Create Table 4.1
library(dplyr)
coffee_variable_summary <- data.frame(
  Name = c("Per Capita Income", "Certification (Treatment/ Control)",
                   "Household Age", "Squared Household Age", "Gender",
                   "Dependency Ratio", "Education Level",
                   "Years of Coffee Production",
                   "Log Total Land", "Access to Credit",
                   "Bad Weather", "Non-farm Income Access"),
  Codename = c("`percapitaincome_day_maleeq`", "`certified`", "`age_hh`",
               "`agesq`", "`gender`", "`depratio`", "`edu`",
```

```
                "`years_cofeproduction`", "`logtotal_land`", "`access_credit`",
                "`badweat`", "`nonfarmincome_access`"),
  Description = c("Average income earned per person within a farming household",
                  "If the farming household is certified (=1) or otherwise (=0)",
                  "Age of the head of the household in years",
                  "Age of the head of the household squared",
                  "Gender of the head of household (male = 1 and female = 0)",
                  "Household members below 14 and above 65 years divided by
                  rest of the household member",
                  "Education of the head of household in years",
                  "Years of experience in coffee farming",
                  "Logarithm of total land size in hectares",
                  "Household has access to credit (yes = 1, otherwise = 0)",
                  "If the household was affected by floods/droughts during the
                  last year (2008-2009)",
                  "If the household has access to nonfarm income")
)

knitr::kable(coffee_variable_summary)
# Create Table 4.2
library(MatchIt)
library(MatchItSE)
library(marginaleffects)

coffee_formula <- as.formula(certified ~ age_hh + agesq + nonfarmincome_access +
                             depratio + logtotal_land + badweat + edu + gender +
```

```
                              years_cofeproduction + access_credit)

coffee_rep_pmodel <- matchit(coffee_formula, data = coffee_data,
                              distance = "glm", method = "nearest",
                              replace = TRUE, estimand = "ATT", discard = "both")

coffee_logit_md <- match.data(coffee_rep_pmodel)

coffee_rep_fit<- lm(percapitaincome_day_maleeq ~ certified,
                    data = coffee_logit_md, weights = weights)

replicated_result <- avg_comparisons(coffee_rep_fit, variables = "certified",
                                     vcov = NULL,
                                     newdata = subset(coffee_logit_md,
                                                      certified == 1),
                                     wts = "weights")

ai_se <- abadie_imbens_se(obj = coffee_rep_pmodel,
                          Y = coffee_data$percapitaincome_day_maleeq)

replicated_result_tbl <- extract_comparison_results(replicated_result)

replicated_result_tbl$SE <- ai_se

rownames(replicated_result_tbl) <- "Replicated Result"
```

```r
knitr::kable(replicated_result_tbl, digits = 4)

# Create Table 4.3

library(cobalt)

coffee_rep_btab <- bal.tab(coffee_rep_pmodel,

                           data = coffee_data,

                           stats = c("mean.diffs","variance.ratios"),

                           binary = "std", continuous = "std",

                           thresholds = c(mean.diffs = 0.1),

                           s.d.denom = "treated")


coffee_rep_btab_ss <- coffee_rep_btab$Observations


coffee_rep_btab <- coffee_rep_btab$Balance[-1,-c(2,3)]


rowlabels <- c(

  "Household Age", "Squared Household Age", "Non-farm Income Access",

  "Log Total Land", "Dependency Ratio", "Bad Weather",

  "Education Level", "Gender", "Years of Coffee Production",

  "Access to Credit")


colnames <- c("Variable","Type", "SMD", "Balance Threshold", "Variance Ratio")


rownames(coffee_rep_btab) <- rowlabels


coffee_rep_btab[,3] <- ifelse(

          coffee_rep_btab[,3] >= "Not Balanced, >0.1", "No", "Yes")
```

```r
knitr::kable(coffee_rep_btab, digits = 4, align = "c")
nobs_coffee_dropped <- sum(coffee_rep_pmodel$discarded, na.rm=TRUE)
coffee_data$discarded <- coffee_rep_pmodel$discarded
nobs_coffee_Tdropped <- nrow(subset(coffee_data,discarded==TRUE&certified==1))
nobs_coffee_Cdropped <- nrow(subset(coffee_data,discarded==TRUE&certified==0))
# Create Figure 4.1
library(ggplot2)
love.plot(coffee_formula, data = coffee_data,
          weights = list(Replication = coffee_rep_pmodel),
          sample.names = c("Unadjusted", "Replication"),
          var.order = "unadjusted", binary = "std",
          abs = TRUE, colors = c("#333333", "#2780e3"),
          shapes = c("circle filled", "circle filled"),
          line = TRUE, thresholds = 0.1, s.d.denom = "treated") +
  labs(title = "Variable Balance of Replication",
       x = "Absolute Standardised Mean Differences", fill = "Method") +
  scale_x_continuous(breaks = seq(0,0.6,length.out=7),
                     expand = expansion(c(0, 0.05))) +
  custom_ggplot_theme
# Create Figure 4.2
discarded_scores <- coffee_rep_pmodel$distance[coffee_rep_pmodel$discarded]

discard_min <- min(discarded_scores, na.rm = TRUE)
discard_max <- max(discarded_scores, na.rm = TRUE)
```

```r
ggplot(coffee_data, aes(x = coffee_rep_pmodel$distance,
                        fill = factor(certified))) +
  geom_density(alpha = 0.6, linewidth = 0.6) +
  scale_fill_manual(values = c("#e5e5e5", "#2780e3"),
                    labels = c("Control", "Certified")) +
  labs(title = "Distribution of Propensity Scores in Replication",
       x = "Propensity Scores", y = "Density", fill = "Group:") +
  scale_x_continuous(expand = expansion(0), limits = c(0, 1)) +
  scale_y_continuous(expand = expansion(0), limits = c(0, 3)) +
  geom_vline(xintercept = discard_max, color = "#333333", linewidth = 0.8) +
  annotate("rect", xmin = 0, xmax = discard_min, ymin = -Inf, ymax = Inf,
           fill = "#333333", alpha = 0.2) +
  annotate("rect", xmin = discard_max, xmax = 1, ymin = -Inf, ymax = Inf,
           fill = "#333333", alpha = 0.2) +
  annotate("text", x = 0.02, y = 1.5,
           label = "Discarded Range", angle = 90, vjust = 1.5, size = 4,
           fontface = "bold", color = "#333333") +
  custom_ggplot_theme
# Create Table 4.4
colnames(coffee_rep_btab_ss) <- c("Control", "Certified")

knitr::kable(coffee_rep_btab_ss, digits=0, align = "c")
# Perform IPW with a GBM
library(WeightIt)
library(cobalt)
```

```r
set.seed(88)
coffee_boosted_weight <- weightit(coffee_formula, data = coffee_data,
                                  method = "gbm", distribution = "bernoulli",
                                  use.offset = T,
                                  shrinkage = seq(0.15, 0.4, length.out = 5),
                                  bag.fraction = 0.67,
                                  interaction.depth = 3:6,
                                  n.trees = 500,
                                  criterion = "smd.mean",
                                  estimand = "ATT")

coffee_boosted_btab <- bal.tab(coffee_boosted_weight, data = coffee_data,
                               stats = c("mean.diffs", "variance.ratios"),
                               binary = "std", continuous = "std",
                               thresholds = c(mean.diffs = 0.1),
                               s.d.denom = "treated")

coffee_boosted_btab <- coffee_boosted_btab$Balance[-1, -c(2,3)]
# Perform IPW with Logistic Regression
coffee_logit_weight <- weightit(coffee_formula, data = coffee_data,
                                method= "glm", estimand = "ATT")

coffee_logit_btab <- bal.tab(coffee_logit_weight, data = coffee_data,
                             formula = coffee_formula,
                             stats = c("mean.diffs", "variance.ratios"),
                             binary = "std", continuous = "std",
```

```
                              thresholds = c(mean.diffs = 0.1),
                              s.d.denom = "treated")

coffee_logit_btab <- coffee_logit_btab$Balance[-1, -c(2,3)]
# Create Table 4.5
library("data.table")
library(cobalt)
library(kableExtra)
library(tidyverse)
coffee_raw_btab <- bal.tab(coffee_formula, data = coffee_data,
                           stats = c("mean.diffs","variance.ratios"),
                           binary = "std", continuous = "std",
                           thresholds = c(mean.diffs = 0.1),
                           s.d.denom = "treated")

coffee_raw_btab <- coffee_raw_btab$Balance[,-c(5,6)]

coffee_combined_btab <- rbindlist(list(coffee_raw_btab,
                                       coffee_logit_btab,
                                       coffee_boosted_btab),
                                  use.names = FALSE)

coffee_combined_btab$Variable <- rep(rowlabels, 3)

coffee_combined_btab <- coffee_combined_btab[, c(5, 1, 2, 3, 4)]
```

```r
coffee_combined_btab[, 4] <- ifelse(
          coffee_combined_btab[, 4] >= "Not Balanced, >0.1", "No", "Yes")

kbl(coffee_combined_btab, digits = 4, booktabs = TRUE, align = "c",
    font_size = 10, col.names = colnames) %>%
  kable_styling(full_width = TRUE) %>%
  column_spec(1, bold = TRUE) %>%
  column_spec(2:5, bold = FALSE, width = "2cm") %>%
  pack_rows("Raw Data", 1, 10, label_row_css = "text-align: center;") %>%
  pack_rows("Logistic Regression and IPTW", 11, 20,
            label_row_css = "text-align: center;") %>%
  pack_rows("Boosting Machine with IPTW", 21, 30,
            label_row_css = "text-align: center;")

kbl(combined_btab[-6], digits = 4, booktabs = TRUE, align = "c",
      font_size = 10) %>%
  kable_styling(full_width = T) %>%
  row_spec(0, bold = TRUE) %>%
  column_spec(1, bold = TRUE) %>%
  column_spec(2:5, bold = F, width = "3cm") %>%
  pack_rows("Raw Data", 1, 7, label_row_css = "text-align: center;") %>%
  pack_rows("Logistic Regression and IPTW", 8, 14,
             label_row_css = "text-align: center;") %>%
  pack_rows("Boosting Machine and IPTW", 15, 21,
             label_row_css = "text-align: center;")
```

```r
# Create Figure 4.3
love.plot(coffee_formula,
          data = coffee_data,
          weights = list(Replication = coffee_rep_pmodel,
                         Logit = coffee_logit_weight,
                         Boosting= coffee_boosted_weight),
          var.order = "unadjusted", binary = "std", continuous = "std",
          abs = TRUE, colors = c("#333333", "#2780e3", "darkblue","darkred"),
          shapes = rep("circle filled", 4),
          line = TRUE, thresholds = 0.1, s.d.denom = "treated", use.grid = F) +
  labs(title = "Variable Balance Using Different Balance Methods",
       x = "Absolute Standardised Mean Differences",
       fill = "Method") +
  scale_x_continuous(breaks = seq(0,0.6,length.out=7),
                     expand = expansion(c(0, 0.05))) +
  custom_ggplot_theme
# Create Table 4.6
coffee_att_formula <- update.formula(as.formula(
  paste("~", paste(attr(terms(coffee_formula), "term.labels"),
                   collapse = " + "))),
  percapitaincome_day_maleeq ~ certified * .)



coffee_logit_fit <- lm_weightit(coffee_att_formula,
                    data = coffee_data, weightit = coffee_logit_weight)
```

```r
coffee_boosted_fit <- lm_weightit(coffee_att_formula,
                                  data = coffee_data,
                                  weightit = coffee_boosted_weight)



coffee_logit_att <- avg_comparisons(coffee_logit_fit, variables = "certified")

coffee_boosted_att <- avg_comparisons(coffee_boosted_fit,
                                      variables = "certified")

coffee_comparisons_tab <- rbind(replicated_result_tbl,
                                extract_comparison_results(coffee_logit_att),
                               extract_comparison_results(coffee_boosted_att))

rownames(coffee_comparisons_tab) <- c("Rep. Result (Logistic with PSM)",
                                      "Logistic Regression and IPW",
                                      "Generalized Boosting Machine and IPW")

knitr::kable(coffee_comparisons_tab, digits = 4)
save.image(file = "globals.RData")
```

5 Conclusion and Summary

In conclusion, propensity score methods are useful causal inference tools when working with observational data. While logistic regression is commonly used, machine learning approaches, can improve the calibration of propensity scores and lead to better covariate balance. Thus, a better estimate of the treatment effect can be obtained. Particularly, gradient boosting machines perform well with strong theoretical properties.

The impact of fair trade certification for coffee producers in developing countries is an interesting problem in causal inference. Replicating Jena et al. (2012) with machine learning propensity scores results in a 10 fold increase in the estimate treatment effect of certification on per capita income which is a notable finding. The significant change in the covariate balance under a machine learning propensity score is a testament to the value of machine learning for propensity scores in this observational situation.

Moving forward, a critical area of research at the intersection of machine learning and causal inference is the exploration of treatment effect heterogeneity, which refers to the variation in treatment effects across different individuals or subgroups. This concept is particularly relevant in fields like targeted medicine and policy, where understanding how different groups respond to a treatment or policy can lead to more effective and equitable outcomes.

Existing methods to estimate heterogeneous treatment effects include causal trees and causal forests, which are designed to detect and quantify this heterogeneity. Additionally, metalearners, such as T-learners and S-learners, are powerful frameworks that adapt machine learning algorithms to estimate treatment effects for different subgroups. A particularly interesting area for future research is making these machine learning algorithms more interpretable, so they can be readily applied in real-world decision-making. Causal rule ensembles, for example, combine the predictive power of machine learning with the interpretability of rule-based models, making it easier to understand the underlying reasons for treatment effect heterogeneity.

References

Abadie, Alberto, and Guido W. Imbens. 2006. "Large sample properties of matching estimators for average treatment effects." *Econometrica* 74 (1): 235–67. https://doi.org/10.1111/j.1468-0262.2006.00655.x.

Allaire, JJ, Yihui Xie, Christophe Dervieux, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, et al. 2024. *rmarkdown: Dynamic Documents for r*. https://github.com/rstudio/rmarkdown.

Arel-Bundock, Vincent, Noah Greifer, and Andrew Heiss. Forthcoming. "How to Interpret Statistical Models Using marginaleffects in R and Python." *Journal of Statistical Software*, Forthcoming.

Austin, Peter. 2011. "An introduction to propensity score methods for reducing the effects of confounding in observational studies." *Multivariate Behavioral Research* 46 (3): 399–424. https://doi.org/10.1080/00273171.2011.568786.

Austin, Peter C. 2008. "A critical appraisal of propensity-score matching in the medical literature between 1996 and 2003." *Statistics in Medicine* 27 (April): 2037–49. https://doi.org/10.1002/sim.3150.

Bader-El-Den, Mohammed, Eleman Teitei, and Todd Perry. 2019. "Biased Random Forest for Dealing with the Class Imbalance Problem." *IEEE Transactions on Neural Networks and Learning Systems* 30 (7): 2163–72. https://doi.org/10.1109/TNNLS.2018.2878400.

Barrett, Tyson, Matt Dowle, Arun Srinivasan, Jan Gorecki, Michael Chirico, and Toby Hocking. 2024. *data.table: Extension of "data.frame"*. https://CRAN.R-project.org/package=data.table.

Breiman, Leo. 1996. “Bagging predictors.” *Machine Learning* 24: 123–40. https://doi.org/10.3390/risks8030083.

———. 2001. “Random Forests.” *Machine Learning* 45: 5–32. https://doi.org/https://doi.org/10.1023/A:1010933404324.

Breiman, L, Jerome H Friedman, Richard A Olshen, and C J Stone. 1984. “Classification and Regression Trees.” *Biometrics* 40: 874. https://api.semanticscholar.org/CorpusID:29458883.

Brookhart, M. Alan, Sebastian Schneeweiss, Kenneth J. Rothman, Robert J. Glynn, Jerry Avorn, and Til Stürmer. 2006. “Variable selection for propensity score models.” *American Journal of Epidemiology* 163 (12): 1149–56. https://doi.org/10.1093/aje/kwj149.

Cannas, Massimo, and Bruno Arpino. 2019. “A comparison of machine learning algorithms and covariate balance measures for propensity score matching and weighting.” *Biometrical Journal* 61 (4): 1049–72. https://doi.org/10.1002/bimj.201800132.

Cunningham, Scott. 2021. “Matching and Subclassification.” In *Causal Inference: The Mixtape*, 175–240. Yale University Press. https://doi.org/10.2307/j.ctv1c29t27.8.

Dehejia, Rajeev H., and Sadek Wahba. 1999. “Causal Effects in Nonexperimental Studies: Reevaluating the Evaluation of Training Programs.” *Journal of the American Statistical Association* 94 (448): 1053–62. https://doi.org/10.1080/01621459.1999.10473858.

Ferri-García, Ramón, and María Del Mar Rueda. 2020. “Propensity score adjustment using machine learning classification algorithms to control selection bias in online surveys.” *PLoS ONE* 15 (4): 1–19. https://doi.org/10.1371/journal.pone.0231500.

Friedman, Jerome H. 2001. “Greedy Function Approximation: A Gradient Boosting Machine.” *The Annals of Statistics* 29 (5): 1189–1232. https://www.jstor.org/stable/2699986.

Goller, Daniel, Michael Lechner, Andreas Moczall, and Joachim Wolff. 2020. “Does the estimation of the propensity score by machine learning improve matching estimation? The case of Germany’s programmes for long term unemployed.” *Labour Economics* 65 (March). https://doi.org/10.1016/j.labeco.2020.101855.

Greifer, Noah. 2024a. *cobalt: Covariate Balance Tables and Plots*. https://CRAN.R-project.org/package=cobalt.

———. 2024b. *WeightIt: Weighting for Covariate Balance in Observational Studies*. https://CRAN.R-project.org/package=WeightIt.

Heinrich, Carolyn. 2010. "A Primer for Applying Propensity-Score Matching." *Development*, no. August: 59. http://www.iadb.org/document.cfm?id=35320229.

Ho, Daniel E., Kosuke Imai, Gary King, and Elizabeth A. Stuart. 2011. "MatchIt: Nonparametric Preprocessing for Parametric Causal Inference." *Journal of Statistical Software* 42 (8): 1–28. https://doi.org/10.18637/jss.v042.i08.

Huntington-Klein, Nick, and Malcolm Barrett. 2021. *causaldata: Example Data Sets for Causal Inference Textbooks*. https://CRAN.R-project.org/package=causaldata.

Jena, Pradyot Ranjan, Bezawit Beyene Chichaibelu, Till Stellmacher, and Ulrike Grote. 2012. "The impact of coffee certification on small-scale producers' livelihoods: A case study from the Jimma Zone, Ethiopia." *Agricultural Economics (United Kingdom)* 43 (4): 429–40. https://doi.org/10.1111/j.1574-0862.2012.00594.x.

King, Gary, and Richard Nielsen. 2019. "Why Propensity Scores Should Not Be Used for Matching." *Political Analysis* 27 (4): 435–54. https://doi.org/10.1017/pan.2019.11.

LaLonde, Robert J. 1986. "Evaluating the Econometric Evaluations of Training Programs with Experimental Data Author." *The American Economic Review* 76 (4): 604–20.

Lampach, Nicolas, and Ulrich B. Morawetz. 2016. "Credibility of propensity score matching estimates. An example from Fair Trade certification of coffee producers." *Applied Economics* 48 (44): 4227–37. https://doi.org/10.1080/00036846.2016.1153795.

Lee, Brian K., Justin Lessler, and Elizabeth A. Stuart. 2010. "Improving propensity score weighting using machine learning." *Statistics in Medicine* 29: 337–46. https://doi.org/10.1002/sim.3782.

Liaw, Andy, and Matthew Wiener. 2002. "Classification and Regression by randomForest." *R News* 2 (3): 18–22. https://CRAN.R-project.org/doc/Rnews/.

McCaffrey, Daniel F., Greg Ridgeway, and Andrew R. Morral. 2004. "Propensity score estimation with boosted regression for evaluating causal effects in observational studies." *Psychological Methods* 9 (4): 403–25. https://doi.org/10.1037/1082-989X.9.4.403.

Naimi, Ashley I., Stephen R. Cole, and Edward H. Kennedy. 2017. "An introduction to g methods." *International Journal of Epidemiology* 46 (2): 756–62. https://doi.org/10.1093/ije/dyw323.

Olson, Matthew A., and Abraham J. Wyner. 2018. "Making Sense of Random Forest Probabilities: a Kernel Perspective," 1–35. http://arxiv.org/abs/1812.05792.

Pedersen, Thomas Lin. 2024. *patchwork: The Composer of Plots*. https://CRAN.R-project.org/package=patchwork.

R Core Team. 2024. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Ridgeway, Greg, and GBM Developers. 2024. *gbm: Generalized Boosted Regression Models*. https://CRAN.R-project.org/package=gbm.

Ridgeway, Greg, Dan Mccaffrey, Andrew Morral, Matthew Cefalu, Lane Burgette, and Beth Ann Griffin. 2024. "Toolkit for Weighting and Analysis of Nonequivalent Groups: A Tutorial for the R TWANG Package." https://doi.org/10.7249/tl136.1.

Rosenbaum, Paul R., and Donald B. Rubin. 1983. "The central role of the propensity score in observational studies for causal effects." *Biometrika* 70 (1): 41–55. https://doi.org/10.1017/CBO9780511810725.016.

Rubin, Donald B. 1974. "Estimating Causal Effects of Treatments in Experimental and Observational Studies." *Journal of Educational Psychology* 66 (5): 688–701. https://doi.org/10.1002/j.2333-8504.1972.tb00631.x.

Schuster, Tibor, Wilfrid Kouokam Lowe, and Robert W. Platt. 2016. "Propensity score model overfitting led to inflated variance of estimated odds ratios." *Journal of Clinical Epidemiology* 80: 97–106. https://doi.org/10.1016/j.jclinepi.2016.05.017.

Setoguchi, Soko, Sebastian Schneeweiss, Alan M. Brookhart, Robert J. Glynn, and Francis E. Cook.

2008. “Evaluating uses of data mining techniques in propensity score estimation: a simulation study.” *Pharmacoepidemiology and Drug Safety* 17 (March): 546–55. https://doi.org/10.1002/pds.

Smith, Jeffrey A., and Petra E. Todd. 2005. *Does matching overcome LaLonde’s critique of nonexperimental estimators?* Vol. 125. 1-2 SPEC. ISS. https://doi.org/10.1016/j.jeconom.2004.04.011.

Splawa-Neyman, Jerzy. 1923. “On the application of probability theory to agricultural experiments. Essay on principles. Section 9.” PhD thesis. https://doi.org/10.1214/ss/1177012031.

Tibshirani, Robert. 1996. “Regression Shrinkage and Selection via the Lasso.” *Journal of the Royal Statistical Society* 58 (1): 267–88. https://www.jstor.org/stable/2346178.

Tu, Chunhao. 2019. “Comparison of various machine learning algorithms for estimating generalized propensity score.” *Journal of Statistical Computation and Simulation* 89 (4): 708–19. https://doi.org/10.1080/00949655.2019.1571059.

Ushey, Kevin, and Hadley Wickham. 2024. *renv: Project Environments*. https://CRAN.R-project.org/package=renv.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D’Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. “Welcome to the tidyverse.” *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Xie, Yihui. 2014. “knitr: A Comprehensive Tool for Reproducible Research in R.” In *Implementing Reproducible Computational Research*, edited by Victoria Stodden, Friedrich Leisch, and Roger D. Peng. Chapman; Hall/CRC.

———. 2015. *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. https://yihui.org/knitr/.

———. 2024. *knitr: A General-Purpose Package for Dynamic Report Generation in r*. https://yihui.org/knitr/.

Xie, Yihui, J. J. Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown.

Xie, Yihui, Christophe Dervieux, and Emily Riederer. 2020. *R Markdown Cookbook*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown-cookbook.

Zhu, Hao. 2024. *kableExtra: Construct Complex Table with "kable" and Pipe Syntax*. https://CRAN.R-project.org/package=kableExtra.

R Version Control

Please note that future updates to these packages may impact replication of results.

Package	Version	Citation
base	4.4.1	R Core Team (2024)
causaldata	0.1.3	Huntington-Klein and Barrett (2021)
cobalt	4.5.5	Greifer (2024a)
data.table	1.15.4	Barrett et al. (2024)
gbm	2.2.2	Ridgeway and Developers (2024)
kableExtra	1.4.0	Zhu (2024)
knitr	1.48.1	Xie (2014); Xie (2015); Xie (2024)
marginaleffects	0.21.0	Arel-Bundock, Greifer, and Heiss (Forthcoming)
MatchIt	4.5.5	Ho et al. (2011)
MatchItSE	1.0	(**MatchItSE?**)
patchwork	1.2.0	Pedersen (2024)
randomForest	4.7.1.1	Liaw and Wiener (2002)
renv	1.0.7	Ushey and Wickham (2024)
rmarkdown	2.27	Xie, Allaire, and Grolemund (2018); Xie, Dervieux, and Riederer (2020); Allaire et al. (2024)
tidyverse	2.0.0	Wickham et al. (2019)

Package	Version	Citation
WeightIt	1.2.0	Greifer (2024b)

A Datasets

A.1 National Supported Work Data

The National Supported Work (NSW) Demonstration Job Training Program dataset originates from a large-scale social experiment conducted in the 1970s in the United States aimed at evaluating the impact of job training on employment and earnings among disadvantaged groups, including ex-addicts, ex-offenders, youth dropouts, and long-term unemployed women. The data contains a wide range of covariates including as age, education, pre-treatment earnings, marital status, and race.

The study is a randomized controlled trial (RCT) design which is rare for jobs and employment data. Participants were randomly assigned to either a treatment group, which received job training, or a control group, which did not. This randomization is notable as it as it simplifies the calculation of a treatment effect.

Initially LaLonde (1986) used the NSW dataset to compare experimental and non-experimental estimators of the treatment effect. His findings highlighted significant discrepancies between the two, underscoring the importance of randomization in estimating causal effects. This study has been widely cited and forms the basis for many discussions on the validity of non-experimental methods.

Following this, Dehejia and Wahba (1999), who revisited LaLonde's analysis and compared many different contemporary methods with varying results.

For these reason it is commonly used in the literature as a toy dataset. It serves as a practical example for students learning about causal inference, allowing them to understand and apply different econometric methods.

```
library('causaldata')
data("nsw_mixtape", package = "causaldata")
nsw_data <- as.data.frame(nsw_mixtape)

nsw_data$data_id <- seq(1,length(nsw_data$data_id))

nsw_data$degree <- abs(nsw_data$nodegree-1)

nsw_data$nodegree <- NULL
```

A.2 Coffee Data from Jena et al. (2012)

The data used in the study by Jena et al. (2012) focuses on smallholder coffee farmers in Ethiopia. It includes a comprehensive survey of coffee-producing households, capturing various socioeconomic and agricultural variables. Key data points include household income, coffee production levels, prices received for coffee (both certified and non-certified), costs associated with certification, and access to markets. Additionally, the dataset encompasses demographic information such as household size, education levels, and access to resources like credit and extension services. This rich dataset allows for a detailed analysis of the impact of coffee certification on the livelihoods of

these farmers, providing insights into both the benefits and challenges associated with certification programs.

The data is best accessed from Lampach and Morawetz (2016) where the data is available in the supplimentary information: https://www.tandfonline.com/doi/full/10.1080/00036846.2016.1153795. This data is also included on this project's github under `datasets/`.

```
library(haven)
coffee_data <-  read_dta("datasets/Jena_etAl_LampachMorawetz.dta")


coffee_data <- zap_formats(coffee_data)


coffee_data <- coffee_data[-c(56,84,156 ),]
```

B Functions

```
extract_comparison_results <- function(results) {
  extracted_results <- data.frame(
    Estimate = results$estimate,
    SE = results$std.error,
    P.Value = results$p.value,
    Lower.CI = results$conf.low,
    Upper.CI = results$conf.high
  )

  return(extracted_results)
}
```

C Theming

The following ggplot2 theme is used to stylise plots.

```
library(ggplot2)


custom_ggplot_theme <- theme_classic(base_size = 11,
                                     base_family = "Source Sans Pro") +
  theme(
      text = element_text(color = "#333333"),
      plot.background = element_blank(),
      panel.background = element_blank(),
      axis.text = element_text(color = "#333333"),
      axis.title = element_text(color = "#333333", face = "bold"),
      legend.text = element_text(color = "#333333"),
      legend.title = element_text(color = "#333333", face = "bold",),
      plot.title = element_text(size = 14, face = "bold", color = "#333333"),
      plot.subtitle = element_text(size = 12, face = "italic",
                                   color = "#333333"),
      strip.text = element_text(face = "bold",color = "#333333"),
      legend.position="bottom")
```

```
theme_set(custom_ggplot_theme)
```