

IRTG 1792 Discussion Paper 2021-005

CATE Meets ML: Conditional Average Treatment Effect and Machine Learning

Daniel Jacob *



* Humboldt-Universität zu Berlin, Germany

This research was supported by the Deutsche Forschungsgesellschaft through the International Research Training Group 1792 "High Dimensional Nonstationary Time Series".

<http://irtg1792.hu-berlin.de>
ISSN 2568-5619



International Research Training Group 1792

CATE meets ML *

The Conditional Average Treatment Effect and Machine Learning

Daniel Jacob

daniel.jacob@hu-berlin.de

April 24, 2021

Abstract

For treatment effects - one of the core issues in modern econometric analysis - prediction and estimation are two sides of the same coin. As it turns out, machine learning methods are the tool for generalized prediction models. Combined with econometric theory, they allow us to estimate a personalized treatment effect - the conditional average treatment effect (CATE). In this tutorial, we give an overview of novel methods, explain them in detail, and apply them via Quantlets in real data applications. We study the effect that microcredit availability has on the amount of money borrowed and if 401(k) pension plan eligibility has an impact on net financial assets, as two empirical examples. The presented toolbox of methods contains meta-learners, like the DR, R-, T- and X-learner, and methods that are specially designed to estimate the CATE like causal BART and generalized random forest. In an additional simulation study we further compare the methods to observe patterns and similarities.

Keywords Causal Inference, CATE, Machine Learning, Tutorial

*Financial support of the European Union's Horizon 2020 research and innovation program "FIN-TECH: A Financial supervision and Technology compliance training programme" under the grant agreement No 825215 (Topic: ICT-35-2018, Type of action: CSA), the European Cooperation in Science & Technology COST Action grant CA19130 - Fintech and Artificial Intelligence in Finance - Towards a transparent financial industry and the Deutsche Forschungsgemeinschaft's IRTG 1792 grant is gratefully acknowledged.

1 Introduction

Estimation and prediction of treatment effects are important tasks for every economist and financial econometrician since treatment effects are often the basis for policy and business decisions. As an illustration, let us look at an idea of microcredits, dating back to Muhammad Yunus, a Nobel Price winner, who discovered in 1976 that very small loans could make a disproportional difference to a poor person. Microcredits work as shown in Figure 1.1. They can increase investments since such credit is easy to get and pay back. Business activity is hence more flexible and could be improved. Increasing gains from a business could increase the household income and further allow for more savings which can be invested in, for example, education.



Figure 1.1: The theory of microcredits.

This specific example was recently applied by [Crépon, Devoto, Duflo, and Parienté \(2015\)](#) who studied the setting where certain villages in Morocco get access to microcredit (the treatment group) while others don't (the control group). As economists, one is interested in the effect that microcredit availability has on the amount of loans which could be an indicator of how demanded such microcredits are. Since we observe certain characteristics for each household, we can condition on such observed variables to see if there is heterogeneity in the effect from microcredit. Figure 1.2 shows an example of what we want to do. The goal is to find subgroups based on characteristics where we believe that the treatment effect is different. As an example, we can partition the households by age and compare young vs. older household members in terms of their effect on microcredit. In both subgroups, we need to make sure that we observe people that are treated and others that did not receive treatment. We can estimate the average treatment effect (ATE) for the young household members, for

example, by taking the difference of their mean outcome given treatment status. We repeat this for the subgroup of older households. Recent methods to estimate the ATE using nonparametric methods on the whole sample include target maximum likelihood estimation (TMLE) (van der Laan, 2010) and double machine learning (Chernozhukov, Chetverikov, Demirer, Duflo, Hansen, Newey, and Robins, 2018a). If the data has many covariates (let us say it has high-dimensionality) and if we don't know which specific subgroup we should focus on, as is the case here, we can use methods that are presented in this tutorial. These methods estimate a treatment effect for each observation based on their covariates, the conditional (on covariates) average treatment effect (CATE). In a further step, we can then look at the heterogeneity and try to link characteristics that are drivers for different treatment effects.

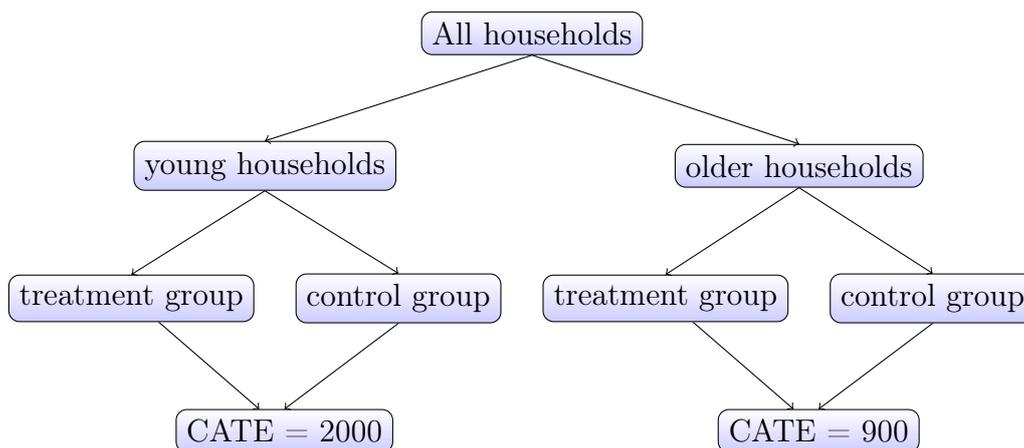


Figure 1.2: CATE example for microcredits

The high-dimensionality of a dataset does not necessarily mean that one has more covariates than observations by default. However, if we are unsure about the structural form, we could include interaction and quadratic terms, and soon the number of dimensions increases. For example, if we have 1000 observations and 30 covariates, then by only including quadratic interactions the amount of covariates increases to 495. Including up to cubic terms leads to a dimension of 5455. If we further assume that only a few covariates are dependent on the outcome and the treatment (often called the approximate sparsity assumption), the task transfers into a selection problem where

standard parametric models are limited and we might want to use machine learning (ML) methods. The reason why this is the case is either that we have more covariates than observations or that the functional forms are complex and we don't know which interaction terms to include in a linear model.

Machine learning is not easily defined. It contains many algorithms with the main focus of prediction (regression), classification, and grouping tasks like clustering. While clustering, as a form of dimensionality reduction, only uses covariates but not outcomes with labels, we call this branch unsupervised ML. The counterpart is called supervised ML. Supervised ML, in general, uses a set of covariates to predict an observed outcome. When talking about prediction we mean the following: Construct an estimator $\hat{\mu}(x)$ of $E[Y|X_i = x]$ using Y and a set of covariates from some training set and predict the values of Y from an independent test set. The goal is to minimize deviations between the true outcome and predicted outcomes from the test set. Note that this is in contrast to the term forecasting. The only assumption so far is that the observations are independent and that the joint distribution of X and Y in the training set is the same as that of the test set. To achieve the goodness of fit (for example minimize the average of the mean-squared error), in an independent test set many alternative models are estimated and the model that maximizes a criterion is selected. We will talk about cross-validation - a concept for model selection - later. The key is that the functional form is mostly determined as a function of the data. Regularization together with systematic model selection may be the main advantages of ML methods. When we talk about ML in this tutorial, we mean supervised machine learning models that are used to make predictions. For a detailed discussion about machine learning in economics see [Mullainathan and Spiess \(2017\)](#) and [Athey \(2019\)](#).

How do we get from prediction to causal inference? A simple, pure prediction approach to get the CATE is to estimate two conditional mean functions, one for the treated observations and one for the non-treated (the control group). For each observation, we can predict the outcome under treatment and control by plugging each observation into both functions. Taking the difference between the two outcomes results in the CATE. Mapping the support of X on Y is a classic regression task for which

machine learning methods are well suited to find generalizable predictive patterns. Since we are only interested in getting a good prediction of the conditional mean, we do not need to know the underlying structural form of this function which enables vanilla ML methods to be sufficient. We call such functions, where the parameters are not of immediate interest, a nuisance function. While the above example of estimating the CATE is quite simple and intuitive, we will see that there are more efficient or automated methods to estimate heterogeneous treatment effects. We will also see that while prediction models are easy to evaluate, causal parameters are not. This is mainly since the objective is different. In prediction, we can optimize a goodness of fit criterion since we observe the true outcome. The causal parameter, however, is never observed in any dataset. As in econometrics, we need to carefully design methods that aim to estimate such parameters of interest, apply statistical theory and expand the set of assumptions to interpret a parameter as causal.

This tutorial is structured as follows. First, we provide an overview of the potential outcome framework and state the necessary assumptions to interpret our parameter of interest as a causal parameter. We then explain different methods that we consider, methods that are very flexible in the choice of the ML algorithm, and methods that are developed to estimate the CATE, mostly relying on tree-based algorithms. As in classical ML, we make use of sample splitting to limit overfitting and allowing for less restrictive assumptions on the nuisance functions. We cover explanations on why and how to do sample splitting and cross-validation. Next, we investigate two empirical datasets, the microcredit example, and the 401(k) pension plan survey. Last, we include a simulation study where we generate the true treatment effect. This allows us to directly compare all different methods in terms of accuracy. Whenever possible we provide and link to Quantlets [Q](#) that are ready-to-use code snippets to implement the discussed methods (the Quantlets are all written in R). The files are not only a replication code for the empirical analysis and the simulation study but contain functions to implement novel methods that aim to estimate the CATE directly. During this tutorial, we will use the terms model, method, and algorithm interchangeably.

Figure 1.3 gives an idea of how a causal structure may look. In the first graph, only the treatment has an impact on the outcome while the second graph also includes covariates that might make the treatment effect dependent on some characteristics. The same is true for the third graph but now the covariates also influence the treatment probability. We say that such a setting is from an observational study since the researcher has no control of the treatment assignment. The first two settings can be seen as a randomized controlled trial (RCT) but only in the second and third can we hope to observe treatment heterogeneity and hence estimate the CATE.

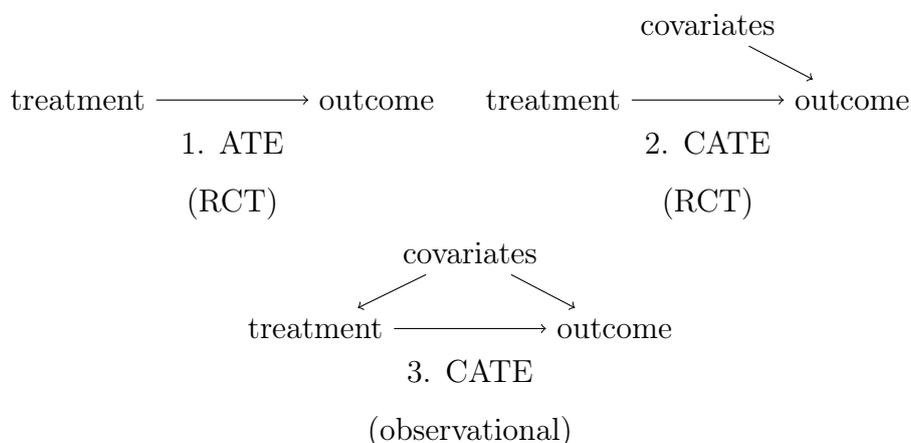


Figure 1.3: Simple causal diagrams - from ATE to CATE.

2 Methods

Let us start with an introduction of the potential outcome framework for which we use the following notations: Each observation has two potential outcomes, Y^1 and Y^0 of which we only observe one, namely the former if someone was treated or the latter if not. The binary treatment indicator is $D \in \{0; 1\}$ and we denote observed covariates $X \in \mathbb{R}$. To interpret the estimated parameter as a causal relationship, the following assumptions are needed; see, for example, [Rubin \(1980\)](#):

1. Conditional independence (or conditional ignorability/exogeneity or conditional unconfoundedness):

$$(Y_i^1, Y_i^0) \perp\!\!\!\perp D_i | X_i.$$

2. Stable Unit Treatment Value Assumption (SUTVA) (or counterfactual consistency):

$$Y_i = Y_i^0 + D_i(Y_i^1 - Y_i^0).$$

3. Overlap Assumption (or common support or positivity):

$$\begin{aligned} \forall x \in \text{supp}(X_i), \quad 0 < P(D_i = 1 | X_i = x) < 1, \\ P(D_i = 1 | X_i = x) \stackrel{\text{def}}{=} e(x). \end{aligned} \tag{2.1}$$

4. Exogeneity of covariates:

$$X_i^1 = X_i^0.$$

Assumption 1 together with Assumption 4 is very natural since they state that the treatment assignment is independent of the two potential outcomes and that the covariates are not affected by the treatment. Assumption 2 ensures that there is no interference, no spillover effects, and no hidden variation between treated and non-treated observations. Assumption 3 states that no subpopulation defined by $X_i = x$ is entirely located in the treatment or control group, hence the treatment probability needs to be bounded away from zero and one. Equation (2.1) is called the propensity score.

Now we define the conditional expectation of the outcome for the treatment or control group as

$$\mu_d(x) = \mathbf{E}[Y_i | X_i = x, D_i = d] \quad \text{with } D \in \{0, 1\}.$$

If we don't use any subscript, we refer to this function as the general conditional expectation.

Our parameter of interest is the CATE ($\tau(x)$), which is formally defined as:

$$\tau(x) = \mathbf{E}[Y_i^1 - Y_i^0 | X_i = x] = \mu_1(x) - \mu_0(x). \quad (2.2)$$

Equation 2.3 shows how the two conditional mean functions can represent the two potential outcomes and hence, by taking the difference, lead to the CATE.

$$\begin{aligned} \tau(x) &= \mu_1(x) - \mu_0(x) \\ &= \mathbf{E}[Y_i | D_i = 1, X_i = x] - \mathbf{E}[Y_i | D_i = 0, X_i = x] \\ &= \mathbf{E}[Y_i^1 | D_i = 1, X_i = x] - \mathbf{E}[Y_i^0 | D_i = 0, X_i = x] \\ &= \mathbf{E}[Y_i^1 | X_i = x] - \mathbf{E}[Y_i^0 | X_i = x] \\ &= \mathbf{E}[Y_i^1 - Y_i^0 | X_i = x] \end{aligned} \quad (2.3)$$

This estimator is of special interest in many areas like medicine or policy actions since it tells us if there are differences in the treatment effect in the population and how big these differences are. It could be, for example, that the average treatment effect of a policy is +2, containing half of the people with a treatment effect of +6 and the other half of -2. Instead of treating everyone, we should only treat people that have a positive effect from the policy (if positive means better). If this is not possible, let us say due to legal or ethical reasons, the policy should not be implemented at all. The CATE will tell us the exact distribution of the effects and, at best, allows us to identify subgroups. To estimate the CATE, we are not primarily interested in the coefficient from regressing X on Y , nor are we interested in the coefficients from the propensity score model. What we want instead is to have a good approximation of the function and hence good estimates from e.g. $\mu_1(x)$ and $\mu_0(x)$. This is why ML methods are well suited for the job.

When reviewing recently proposed methods for the estimation of the CATE, we can categorize them into two groups. The first group contains methods that are built on off-the-shelf machine learning methods (such as the lasso, random forest (RF), Bayesian additive regression trees (BART), boosting methods, or neural networks). Since the base learners are not designed to estimate the CATE directly, the literature calls them meta-learners or generic ML algorithms. The second group of methods alters existing machine learning methods in a way that they can be used to estimate the CATE directly (examples are causal boosting by Powers, Qian, Jung, Schuler, Shah, Hastie, and Tibshirani (2018), causal forest by Athey, Wager, and Tibshirani (2019) or Bayesian regression tree models for causal inference by Hahn, Murray, and Carvalho (2020)). See Künzel, Sekhon, Bickel, and Yu (2019) for a comparison between meta-learners like the S-, T-, and X-learner as well as the causal forest in a simulation study. Knaus, Lechner, and Strittmatter (2020) compare meta-learners like inverse probability weighting (IPW) estimator, doubly-robust (DR), modified covariate method (MCM), R-learner, and different versions of the causal forest in an empirical Monte Carlo study while Nie and Wager (2020) compare their R-learner with the S-, T-, X- and U-learner as well as causal boosting. Regarding the base learners (the ML methods), Künzel et al. (2019) use a random forest and BART. Knaus et al. (2020) use RF and lasso while Nie and Wager (2020) use boosting and the lasso for the estimation of the nuisance functions and the treatment effects. In high dimension, the use of machine learning methods, such as boosting or random forests to estimate the propensity score, works quite well as McCaffrey, Ridgeway, and Morral (2004) and Wyss, Ellis, Brookhart, Girman, Jonsson Funk, LoCasale, and Stürmer (2014) show. The estimation of probabilities given a large set of covariates is nothing less than a prediction problem in where ML methods are superior. In Table 2.1 we list popular methods by category, including links to the Quantlets. The references refer to recent papers that use these methods and provide theoretical properties.

Table 2.1: Methods to estimate CATE

Category	Method	Reference	Quantlet
Meta-Learner	DR-learner	Kennedy (2020)	
	IPW-learner	Horvitz and Thompson (1952)	
	R-learner	Nie and Wager (2020)	
	S-learner	Hill (2011)	
	T-learner	Hansotia and Rukstales (2002)	
	X-learner	Künzel et al. (2019)	
Modified ML	Causal BART	Hahn et al. (2020)	
Methods	Causal Boosting	Powers et al. (2018)	
	Causal Forest	Athey et al. (2019)	

2.1 Meta-Learners

In the following, we briefly describe the considered meta-learners. We follow the definition of meta-learners by [Künzel et al. \(2019\)](#) and describe them as methods to estimate the CATE using ML methods that are built for regression or classification tasks only. The S- and T-learner, for example, can use any vanilla ML method to predict the conditional outcome. The prediction models can then be used to estimate the conditional average treatment effects. The second class of methods uses additional information from the propensity score. They contain the DR- and X-learner. Again, the conditional mean functions, as well as the propensity score, can be estimated using a broad range of ML methods. The aforementioned methods are also called transformed outcome methods. The idea is to generate a pseudo-outcome using the estimated nuisance functions in the first step. This can be seen as an approximation of the conditional average treatment effect. The pseudo-outcome, which we show in Table 2.2, is an unbiased estimate of the CATE given that the nuisance parameters are known (e.g. if we would know the true propensity score). In a second step, the pseudo-outcome is mapped on the covariates to get the final estimate and to make predictions on new observations. The reason for prediction is that the observed data after a treatment

assignment includes the outcome, covariates, and the treatment assignment variable. If we want to classify new observations, we only observe the covariates. Hence we need a model that maps the covariates on the estimated treatment effect. The mapping also serves as a smoother since it could be the case that some pseudo-outcome values are quite extreme (e.g. if the propensity score estimate is very low or high). The last method that we examine in this category is the R-learner. However, it does not generate a pseudo-outcome in the classical sense as it needs algorithms that can modify the loss function. Still, many ML methods can be used which is why we include the R-learner into the category of meta-learners. Currently, R-packages are available for the R-, S-, T-, U-, and X-learner (`install_github("xnie/rlearner")`) and the M-, S-, T-, and X-learner (`install_github("soerenkuenzel/causalToolbox")`). Causal analysis via the potential outcome framework and causal graph theory for Python can be found in [Sharma, Kiciman, et al. \(2019\)](#). For heterogeneous treatment effect analysis via machine learning in Python see [EconML \(2019\)](#). The list of methods above is not a complete list of methods in this subject. For example, we do not talk about methods for instrumental variables, multiple-treatment, difference-and-difference methods, or regression discontinuity designs. We also note that there are other methods to estimate treatment effects in cross-sectional settings. For example, one of the first methods developed to control for confounding bias is the inverse probability weighting (IPW) estimator by [Horvitz and Thompson \(1952\)](#). In Algorithm 9 we show how to implement this method. Some methods use neural networks to estimate heterogeneous treatment effects. See, for example, the recent work by [Farrell, Liang, and Misra \(2021\)](#) who use a deep neural network for semiparametric inference and develop nonasymptotic high probability bounds.

2.1.1 Single- (S-learner) and two-model learner (T-learner):

Let us first start with a very simple and intuitive method, the T-learner. It is a two-step approach where the conditional mean functions $\mu_1(x) = \mathbb{E}[Y^1|X_i = x]$ and $\mu_0(x) = \mathbb{E}[Y^0|X_i = x]$ are estimated separately with any generic machine learning algorithm. The difference between the two functions results in the CATE as shown in

Table 2.2 and as seen in equation 2.3. One problem with the T-learner is that it aims to minimize the mean squared error for each separate function rather than the mean squared error of the treatment effect. By splitting the sample into two groups there is only information on one group. This might be problematic if the two functions shrink different covariates which are important in both groups. This is especially the case in an RCT. See, for example, [Künzel et al. \(2019\)](#); [Kennedy \(2020\)](#) for settings when the T-learner is not the optimal choice. An alternative is to model only one function and include the treatment assignment into this function. This approach is called the S-learner. See for example, [Hill \(2011\)](#) and [Foster, Taylor, and Ruberg \(2011\)](#) for early examples of proposing the S-learner. Algorithm 7 in the Appendix describes how to implement the S-learner while algorithm 1 shows the implementation for the T-learner.

Algorithm 1: T-learner

Input: $Z_i = \{Y_i, D_i, X_i\}_{i \in N}$

- 1 Split sample Z into K random subsets
- 2 **for** k in $\{1, \dots, K\}$ **do**
- 3 **assign** Sample $S_a = Z \cup S_k$ and S_k
- 4 **regress** $Y_i^0 = \hat{\mu}_0(X_i^0) + \hat{U}_i^0$, with $i \in S_a | D = 0$
- 5 **regress** $Y_i^1 = \hat{\mu}_1(X_i^1) + \hat{U}_i^1$, with $i \in S_a | D = 1$
- 6 **estimate** $\hat{Y}_i^0 = \hat{\mu}_0(X_i)$, with $i \in S_k$
- 7 **estimate** $\hat{Y}_i^1 = \hat{\mu}_1(X_i)$, with $i \in S_k$
- 8 **create** $\hat{\tau}_k(X_i) = \hat{\mu}_1(X_i) - \hat{\mu}_0(X_i)$
- 9 **end**
- 10 **combine** $\hat{\tau}(X_i) = \{\hat{\tau}_1, \hat{\tau}_k, \dots, \hat{\tau}_K\}$

2.1.2 Doubly-Robust learner (DR-learner):

A more efficient method than the T-learner can be the DR-learner. It builds on the T-learner and adds a version of the inverse probability weighting (IPW) scheme on the residuals of both regression functions $\{Y^d - \hat{\mu}_d(x)\}$. We can think of it as combining two different models and hence avoid drawbacks like the minimization goal from the T-learner and a potentially high variance from an IPW model when some propensity

scores are small. The doubly-robust learner takes its name from a double robustness property which states that the estimator remains consistent if either the propensity score model or the conditional outcome model is correctly specified (Lunceford and Davidian, 2004). The DR-learner creates a pseudo-outcome which is an unbiased estimator for the CATE:

$$\hat{\psi}_{DR} = \hat{\mu}_1(x) - \hat{\mu}_0(x) + \frac{D \{Y - \hat{\mu}_1(x)\}}{\hat{e}(x)} - \frac{(1 - D) \{Y - \hat{\mu}_0(x)\}}{(1 - \hat{e}(x))}. \quad (2.4)$$

Equation 1.17 in the Appendix shows the double-robustness property by rewriting equation 2.4. Whenever the propensity score or the conditional mean function is correctly specified the doubly-robust estimator converges to $E[Y^1] - E[Y^0]$.

The first part in equation 2.4 can be seen as a regression adjustment parameter (this is the difference between the two conditional mean functions). The second part, which makes use of the propensity score, can be seen as an inverse probability weighting estimator applied to the residual from the conditional mean functions. The DR-learner is very flexible in the choice of the ML method. Estimating the nuisance parameters we can use any ML method. Since the pseudo-outcome is already an unbiased estimator for the CATE the loss-function in the final regression task is to minimize the mean squared error. This allows using the same range of ML methods as in the first step. Recently, this estimator has gained popularity to estimate the CATE, especially in high-dimensional settings. See, for example, the work by Fan, Hsu, Lieli, and Zhang (2019). Most recently, Kennedy (2020) find that for estimating the CATE, the finite-sample error-bound from the DR-learner at most deviates from an oracle error rate by the product of the mean squared error of the propensity score and the conditional mean estimator. As can be seen from equation 2.4, extreme propensity score estimates can lead to large pseudo-outcome estimates. Hence it is necessary to look at the distribution of the propensity score and, if necessary, apply methods to make the overlap assumption more realistic.

Algorithm 2: DR-learner

Input : $Z_i = \{Y_i, D_i, X_i\}_{i \in N}$

- 1 Split sample Z into K random subsets
- 2 **for** k in $\{1, \dots, K\}$ **do**
- 3 **assign** Sample $S_a = Z \cup S_k$ and S_k
- 4 **regress** $D_i = \hat{e}(X_i) + \hat{V}_i$, with $i \in S_a$
- 5 **regress** $Y_i^0 = \hat{\mu}_0(X_i^0) + \hat{U}_i^0$, with $i \in S_a | D = 0$
- 6 **regress** $Y_i^1 = \hat{\mu}_1(X_i^1) + \hat{U}_i^1$, with $i \in S_a | D = 1$
- 7 **estimate** $\hat{D}_i = \hat{e}(X_i)$, with $i \in S_k$
- 8 **estimate** $\hat{Y}_i^0 = \hat{\mu}_0(X_i)$, with $i \in S_k$
- 9 **estimate** $\hat{Y}_i^1 = \hat{\mu}_1(X_i)$, with $i \in S_k$
- 10 **create** $\hat{\psi}_{DR,k} = \hat{\mu}_1(x) - \hat{\mu}_0(x) + \frac{D\{Y - \hat{\mu}_1(x)\}}{\hat{e}(x)} - \frac{(1-D)\{Y - \hat{\mu}_0(x)\}}{(1 - \hat{e}(x))}$
- 11 **store** $\hat{\psi}_{DR,k}$ for $i \in S_k$
- 12 **end**
- 13 **combine** $\hat{\psi}_{DR} = \{\hat{\psi}_{DR,1}, \hat{\psi}_{DR,2}, \dots, \hat{\psi}_{DR,K}\}$
- 14 *Cross-fitting:*
- 15 **for** *oob* in (1:2) **do**
- 16 **if** *oob* = 1: $S_{oob} = Z_i$ with $i \in \{1, \dots, N/2\}$ and $S_{train} = Z_i \cup S_{oob}$
- 17 **if** *oob* = 2: $S_{train} = Z_i$ with $i \in \{1, \dots, N/2\}$ and $S_{oob} = Z_i \cup S_{in}$
- 18 **for** l in 1:5 **do**
- 19 **split** S_{train} in $\{S_1, S_2, \dots, S_5\}$
- 20 **regress** $\hat{\psi}_i = \hat{t}_{DR}(X_i) + W_i$, for $i \in S_l$
- 21 **estimate** $\tilde{\tau}_l(X_i) = \hat{t}_{DR}(X_i)$, with $i \in S_{oob}$
- 22 **end**
- 23 **average** $\hat{\tau}_{oob}(X_i) = E[\tilde{\tau}(X_i)]$
- 24 **end**
- 25 **row bind** $\hat{\tau}(X_i) = \{\hat{\tau}_1, \hat{\tau}_2\}$

2.1.3 R-learner:

The R-learner makes use of the idea of orthogonalization to cancel out any selection bias that may arise in observational studies from observed covariates. Here, the residuals from the regression of Y on X are regressed on the residuals from the regression of D on X and weighted by the squared residuals, $\{D - \hat{e}(x)\}^2$. This is similar to the double

machine learning approach from [Chernozhukov et al. \(2018a\)](#) where their estimator of interest is the ATE. [Nie and Wager \(2020\)](#) develop a general class of two-step algorithms for the estimation of the CATE. The R-learner, as from residualized and as an homage to Peter M. Robinson, makes explicit use of machine learning methods. Achieving Neyman orthogonality using a residuals-on-residuals (or debiasing) approach has a long history in econometrics (see the Frisch–Waugh–Lovell theorem from the 1930s for linear regression) and mainly builds on the work by [Robinson \(1988\)](#) who replaces the linear parts by non-parametric kernel regression. The CATE from the R-learner is obtained by the following minimization task:

$$\hat{\tau}(\cdot) = \operatorname{argmin}_{\tau} \left\{ \frac{1}{n} \sum_{i=1}^n [\{Y_i - \hat{\mu}^{(-i)}(X_i)\} - \{D_i - \hat{e}^{(-i)}(X_i)\} \tau(X_i)]^2 + \Lambda_n\{\tau(\cdot)\} \right\}. \quad (2.5)$$

The superscript $(-i)$ indicates the sample splitting. The conditional mean functions are trained without the i -th observations and evaluated only for i . We will explain certain sample splitting procedures later. The term $\Lambda_n\{\tau(\cdot)\}$ can be interpreted as a regularizer on the complexity of the $\tau(\cdot)$ function. In practice, this regularization term could be explicitly given as in penalized regression or implicitly introduced, e.g., as provided by a carefully designed deep neural network. The main difference to the pseudo-outcome estimators (the DR- and X-learner) is that the R-learner needs to alter the loss-function of the ML method. Even if ψ_R with weights equal to 1 as an estimator for $\tau(x)$ it can suffer from high variance if the nuisance functions are not known. The variance is mainly caused by the propensity score since $\{D - \hat{e}(x)\}$ is in the denominator. This is where the weighting comes into play. Observations that have a high variance are weighted by the squared of $\{D - \hat{e}(x)\}$ and hence are less important. The weights for each observation directly influence the loss function (e.g. in boosting methods they manipulate the gradient). Therefore, applying the R-learner needs ML methods that have the option of altering the loss-function through weighting. The following methods have this option: lasso and ridge regression (`glmnet`), boosting (included in the DMatrix format) (`XGBoost`), neural network (`nnet`) and the random

forest (**ranger**). Note that the ranger package seems to be the only implementation of weights for a random forest. The weights are applied on the whole training sample and observations with larger weights will be selected with higher probability in the bootstrap (or subsampled) samples for the trees.

Algorithm 3: R-learner

Input: $Z_i = \{Y_i, D_i, X_i\}_{i \in N}$

- 1 Split sample Z into K random subsets
- 2 **for** k in $\{1, \dots, K\}$ **do**
- 3 **assign** Sample $S_a = Z \cup S_k$ and $S_m = S_k$
- 4 **regress** $D_i = \hat{e}(X_i) + \hat{V}_i$, with $i \in S_a$
- 5 **regress** $Y_i = \hat{\mu}(X_i) + \hat{U}_i$, with $i \in S_a$
- 6 **estimate** $\hat{D}_i = \hat{e}(X_i)$, with $i \in S_m$
- 7 **estimate** $\hat{Y}_i = \hat{\mu}(X_i)$, with $i \in S_m$
- 8 **create** $\hat{\psi}_R = \frac{(Y_i - \hat{\mu}(X_i))}{(D_i - \hat{e}(X_i))}$ and $w_i = (D_i - \hat{e}(X_i))^2$
- 9 **store** $\hat{\psi}_{R,k}$ and $w_{i,k}$ for $i \in S_k$
- 10 **end**
- 11 **combine** $\hat{\psi}_R = \{\hat{\psi}_{R,1}, \hat{\psi}_{R,k}, \dots, \hat{\psi}_{R,K}\}$, $w_R = \{w_1, w_k, \dots, w_K\}$
- 12 *Cross-fitting:*
- 13 **for** *oob* in (1:2) **do**
- 14 **if** *oob* = 1: $S_{oob} = Z_i$ with $i \in \{1, \dots, N/2\}$ and $S_{train} = Z_i \cup S_{oob}$
- 15 **if** *oob* = 2: $S_{train} = Z_i$ with $i \in \{1, \dots, N/2\}$ and $S_{oob} = Z_i \cup S_{in}$
- 16 **for** l in 1:5 **do**
- 17 **split** S_{train} in $\{S_1, S_2, \dots, S_5\}$
- 18 **regress** $\hat{\psi}_i = \hat{t}_R(X_i) + W_i$ and weight by w_R , for $i \in S_l$
- 19 **estimate** $\tilde{\tau}_l(X_i) = \hat{t}_R(X_i)$, with $i \in S_{oob}$
- 20 **end**
- 21 **average** $\hat{\tau}_{oob}(X_i) = \mathbb{E}[\tilde{\tau}(X_i)]$
- 22 **end**
- 23 **row bind** $\hat{\tau}(X_i) = \{\hat{\tau}_1, \hat{\tau}_2\}$

2.1.4 X-learner:

Künzel et al. (2019) propose the X-learner which estimates a treatment effect separately for the control and the treatment group. This might be especially helpful in situations where the proportion of the two groups is highly imbalanced. The X-learner has several steps. The first step is identical to the T-learner, namely estimating the two conditional mean functions. In the second step, we predict the counterfactual outcome using the two functions. If a person is treated and hence the observed outcome is Y^1 we subtract the estimated counterfactual. If a person is in the control group we use the estimated counterfactual outcome and subtract the observed outcome (Y^0). This results in two imputed treatment effects:

$$\hat{\psi}_X^1 \stackrel{\text{def}}{=} Y^1 - \hat{\mu}_0(x^1) \quad \text{for } D_i = 1, \quad (2.6)$$

$$\hat{\psi}_X^0 \stackrel{\text{def}}{=} \hat{\mu}_1(x^0) - Y^0 \quad \text{for } D_i = 0. \quad (2.7)$$

These imputed effects are now used in a third step to regress them individually on the covariates to obtain $\hat{\tau}_0(x)$ (the CATE for the control group) and $\hat{\tau}_1(x)$ (the CATE for the treatment group). The final estimator combines the two estimators plus some weights, $g(x)$:

$$\hat{\tau}(x) = g(x)\hat{\tau}_0(x) + \{1 - g(x)\}\hat{\tau}_1(x).$$

In a randomized controlled trial, the two estimates should not differ significantly. If there are confounding variables and if the support of the treatment variable given covariates differs among the treatment status we would expect the two estimates to be different. Hence, a natural weighting function for $g(x)$ could be the propensity score. We would use $1 - \hat{e}(x)$ for the treatment group and $\hat{e}(x)$ for the control group estimate, respectively.

Algorithm 4 describes the procedure in detail and takes sample-splitting into account.

Algorithm 4: X-learner

Input: $Z_i = \{Y_i, D_i, X_i\}_{i \in N}$

- 1 Split sample Z into K random subsets
- 2 **for** k in $\{1, \dots, K\}$ **do**
- 3 **assign** Sample $S_a = Z \cup S_k$ and S_k
- 4 **regress** $D_i = \hat{e}(X_i) + \hat{V}_i$, with $i \in S_a$
- 5 **regress** $Y_i^0 = \hat{\mu}_0(X_i^0) + \hat{U}_i^0$, with $i \in S_a | D = 0$
- 6 **regress** $Y_i^1 = \hat{\mu}_1(X_i^1) + \hat{U}_i^1$, with $i \in S_a | D = 1$
- 7 **estimate** $\hat{D}_i = \hat{e}(X_i)$, with $i \in S_k$
- 8 **estimate** $\hat{Y}_i^0 = \hat{\mu}_0(X_i)$, with $i \in S_k$
- 9 **estimate** $\hat{Y}_i^1 = \hat{\mu}_1(X_i)$, with $i \in S_k$
- 10 **create** $\hat{\psi}_X^1 \stackrel{\text{def}}{=} Y^1 - \hat{\mu}_0(x^1)$ for $i \in S_k$
- 11 **create** $\hat{\psi}_X^0 \stackrel{\text{def}}{=} \hat{\mu}_1(x^0) - Y^0$ for $i \in S_k$
- 12 **store** $\hat{\psi}_X^1, \hat{\psi}_X^0$ and $\hat{e}(X_i)$ for $i \in S_k$
- 13 **end**
- 14 **combine** $\hat{\psi}_X^1 = \{\hat{\psi}_{X,1}^1, \hat{\psi}_{X,k}^1, \dots, \hat{\psi}_{X,K}^1\}$, $\hat{\psi}_X^0 = \{\hat{\psi}_{X,1}^0, \hat{\psi}_{X,k}^0, \dots, \hat{\psi}_{X,K}^0\}$,
 $\hat{e}(X_i) = \{\hat{e}_1(X_i), \hat{e}_k(X_i), \dots, \hat{e}_K(X_i)\}$
- 15 **regress** $\hat{\psi}_X^1 = \hat{t}^1(X_i) + W_i^1$, with $i \in Z$
- 16 **regress** $\hat{\psi}_X^0 = \hat{t}^0(X_i) + W_i^0$, with $i \in Z$
- 17 **estimate** $\hat{\tau}_k^1(X_i) = \hat{t}^1(X_i)$, with $i \in Z$
- 18 **estimate** $\hat{\tau}_k^0(X_i) = \hat{t}^0(X_i)$, with $i \in Z$
- 19 **average** $\hat{\tau}_k(X_i) = \hat{e}(X_i)\hat{\tau}_k^0 + (1 - \hat{e}(X_i))\hat{\tau}_k^1$

2.1.5 Summary of meta-learners:

We summarise the considered meta-learners in Table 2.2 where $\hat{\psi}$ states the pseudo-outcome or estimator for each of the learners. The last column counts the number of nuisance functions needed to estimate the pseudo-outcome or estimator. In brackets, we state the total number of models needed to get the final CATE estimate. Note that the X-learner is regressed only for the treated observations and again only for the observations in the control group. This is why we need two more additional models for the final estimate.

The estimators from Table 2.2 can be represented as a weighted minimization problem which solves the following:

Table 2.2: Summary of meta-learners

Method	Estimator/Pseudo-outcome	Weights (w_i)	# of Models
S-learner	$\hat{\psi}_S = \hat{\mu}(x, d = 1) - \hat{\mu}(x, d = 0)$	1	1 (2)
T-learner	$\hat{\psi}_T = \hat{\mu}_1(x) - \hat{\mu}_0(x)$	1	2 (3)
DR-learner	$\hat{\psi}_{DR} = \hat{\psi}_T + \frac{D \{Y - \hat{\mu}_1(x)\}}{\hat{e}(x)} - \frac{(1-D) \{Y - \hat{\mu}_0(x)\}}{(1 - \hat{e}(x))}$	1	3 (4)
R-learner	$\hat{\psi}_R = \frac{\{Y - \hat{\mu}(x)\}}{\{D - \hat{e}(x)\}}$	$\{D - \hat{e}(x)\}^2$	2 (3)
IPW-learner	$\hat{\psi}_{IPW} = \frac{DY}{\hat{e}(x)} - \frac{(1-D)Y}{(1 - \hat{e}(x))}$	1	1 (2)
X-learner	$\hat{\psi}_X^1 \stackrel{\text{def}}{=} Y^1 - \hat{\mu}_0(x^1)$ $\hat{\psi}_X^0 \stackrel{\text{def}}{=} \hat{\mu}_1(x^0) - Y^0$	1	3 (5)

Notes: Considered meta-learners that estimate the CATE. # of Models counts the number of nuisance functions to estimate the pseudo-outcome. Numbers in brackets count the total number of models to train to get the final CATE estimate or to make predictions.

$$\min_{\tau} \left\{ N^{-1} \sum_{i=1}^N w_i \{ \hat{\psi}_i - \tau(x) \}^2 \right\}.$$

2.1.6 The choice of ML algorithms for meta-learners:

The accuracy of the CATE estimation depends on the accuracy of the nuisance functions and hence on the choice of the ML method. To minimize the dependence of the ML methods on our estimates, we do not assign specific machine learning methods for the estimation but consider a range of different popular methods. To choose which ML method to use for each nuisance function as well as for any additional functions,

we use a stacking method. In such a setting, not only one ML method may be chosen but an ensemble of methods that are stacked together with different weights. We use the SuperLearner package as proposed by [Polley, Rose, and Van der Laan \(2011\)](#). It also enables us to choose different models for each nuisance function and setting. The package offers a general class of prediction methods to be considered by the ensemble. From the 42 different algorithms, we select gradient boosted trees (`xgboost`), neural network (`nnet`) and random forest (`ranger`) for our analysis. Note that the R-learner needs to include weights to minimize the R-loss in the algorithm, so we need to make sure that the ML methods we use have this possibility included. While many researchers include the lasso in their simulations or empirical analysis, we do not use this approach. The reason is that the lasso algorithm would ideally need to assume a parametric form. This means that if we believe that there are interaction and or polynomial effects from X on Y , we would need to include such transformations. There are extensions like the adaptive lasso that expand the feature space by including such additional factors. The computation time does however increase the more features we include. These are the main reasons why we do not include the lasso in our analysis.

We use 10-fold cross-validation to estimate the performance of all machine learning models. Cross-validation is a resampling procedure used to evaluate ML models on a finite data sample. Depending on the ML model, the data can be fit perfectly and hence produce a high variance (overfitting). This is, however, on the training sample and the model can behave poorly on unseen data. Hence, we have to validate our models. We could use a part of the data for validation. Since there is never enough data, removing a part of it poses a potential for underfitting (we might lose trends in the data or important patterns). What we require instead is a method that provides enough data for training the model and also leaves enough data for validation. K-fold cross-validation does exactly that. This approach involves randomly splitting the set of observations into K groups, or folds, of approximately equal size. The model is fit on folds 2 to K while the first fold is used as a validation set. It is also important that any preparation of the data before fitting the model occur on the training sample that is used for cross-validation within the loop rather than on the broader data sample.

This also applies to any hyperparameter tuning, for example, the number of trees, the minimum observations within a node, learning rates, or shrinkage parameters. There is no formal rule for the choice of K but usually, it is set to 5 or 10. These values have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance. The reason is the following: The larger K , the smaller the difference in size between the (original) training set and the resampling subset ($K - 1$ folds). As this difference decreases, the bias of the technique becomes smaller. This means that the bias is smaller for $K = 10$ than for $K = 5$. A special case of cross-validation is the leave-one-out cross-validation (LOOCV). In this case, K is set to the sample size and only one observation is the validation sample. In all procedures, the I resampled estimates of performance are summarized (e.g. by the mean and the standard error).

Since we apply multiple models to estimate the nuisance functions, we create a weighted average among all models. Using stacking, we can find the optimal combination of a collection of prediction algorithms or even different settings within one model. In other words, we build a linear model that uses the outcome variable of the validation set as the dependent variable and all different base learners as the input variables. For the random forest, we set the following tuning parameters: `n.trees=1000`, `min.node.size=10`.

2.2 Modified ML Methods

We now describe some methods that modify existing ML methods to estimate the CATE directly. In contrast to meta-learners that are flexible in the choice of the ML algorithm, these methods use a specific ML method (mostly tree-based algorithms). Packages or code in R are available for the causal forest (`grf`), the causal Boosting (<https://github.com/saberpowers/causalLearning>) and the causal BART (`install_github("vdorie/bartCause")`). Since causal boosting is computationally expensive, we do not consider this method in our analysis.

2.2.1 Causal Forest:

The causal forest method, part of the generalized random forest (GRF) by [Athey et al. \(2019\)](#) builds on a random forest algorithm to find neighborhoods in the covariate space. These neighborhoods are built by recursive splitting the covariates into subgroups while the criterion to do so is based on heterogeneity in treatment effects. The idea is to find leaves where the treatment effect is constant but different from other leaves. If we know that $\tau(x)$ were constant over some neighbourhood $N(x)$, we could solve a partially linear model over $N(x)$ using the residual-on-residual approach (see e.g. [Robinson \(1988\)](#)): First, we estimate $e(x) = \mathbb{E}[D_i|X_i = x]$ and second, $\mu(x) = \mathbb{E}[Y_i|X_i = x]$. We can use any non-parametric method like the lasso, random forests, boosting methods, neural networks and others. The final step is to estimate $\tau(x)$ over the neighbourhood $N(x)$:

$$\hat{\tau}(x) = \frac{\sum_{\{i: X_i \in N(x)\}} \{Y_i - \hat{\mu}(X_i)\} \{D_i - \hat{e}(X_i)\}}{\sum_{\{i: X_i \in N(x)\}} \{D_i - \hat{e}(X_i)\}^2}. \quad (2.8)$$

Note that this approach looks similar to the R-learner. [Chernozhukov et al. \(2018a\)](#) showed that when using any of the aforementioned ML methods for the estimation of the nuisance functions and then use the residual-on-residual approach to estimate the average treatment effect the following regularity condition holds:

Given that,

$$\mathbb{E}[\{\mu(X_i) - \hat{\mu}(X_i)\}^2]^{\frac{1}{2}} \ll \frac{1}{n^{1/4}}, \quad \mathbb{E}[\{e(X_i) - \hat{e}(X_i)\}^2]^{\frac{1}{2}} \ll \frac{1}{n^{1/4}}, \quad (2.9)$$

we get a central limit theorem such that $\sqrt{n}(\hat{\tau} - \tau) \Rightarrow \mathcal{N}(0, V)$. The treatment effect in the above setting, however, has to be constant. We can assume that with heterogeneous treatment effects, there are subgroups such that the constant effect assumption holds. The question of how to find such accurate subgroups is exactly where the (causal) random forest comes into play. To create leaves that consist of observations with the same (average) treatment effect, the splitting criterion has to rely on maximizing the heterogeneity in treatment effects between leaves (similar to maximizing the variance between the leaves). Here we use again the method from

equation (2.8). In observational studies where self-selection into treatment is present, the first splits might not be a good representation of the treatment effect rather than differences due to confounding variables. To overcome this problem, [Athey et al. \(2019\)](#) suggest applying local-centering. This means that we use the residuals of the outcome and treatment variable as data instead of the original values. Therefore one has to train two nuisance functions beforehand to predict the conditional mean which is used to create the residuals. While machine learning methods rely on sample splitting to avoid overfitting, the causal random forest integrates this via an honesty condition. A tree is honest if, for each training sample i , it only uses the response Y_i to estimate the within-leaf treatment effect or to decide where to place the split, but not both.

So far we have looked at how a single tree is build and how the final treatment effect can be estimated. To extend this procedure to multiple trees, let us view a forest as a weighting function:

$$\hat{\mu}(x) = B^{-1} \sum_{b=1}^B \sum_{i=1}^n Y_i \frac{1\{X_i \in L_b(x)\}}{|L_b(x)|} = \sum_{i=1}^n Y_i \underbrace{B^{-1} \sum_{b=1}^B \frac{1\{X_i \in L_b(x)\}}{|L_b(x)|}}_{\alpha_i(x)}. \quad (2.10)$$

Instead of seeing a forest as a double average over observations within a leaf and B single trees, we can integrate the first sum to be a weighted average over all X_i that fall into the leaf $L_b(x)$ and divide by the total number of observations within the leaf ($|L_b(x)|$). This weighted average tells us how often Y_i falls into a certain leaf and hence the weight that we have to apply to control for the different proportions. The weights can be represented as $\alpha_i(x)$. We can now use these weights to weigh each observation in a generalized method of moments estimator where we apply a linear model, regressing the residuals of D_i on the residuals of Y_i and weigh by α_i . This is how we get the CATE using a random forest. The algorithm is implemented in the `grf` package. See [Friedberg, Tibshirani, Athey, and Wager \(2018\)](#) for an extension of this approach to local linear forests.

Algorithm 5 describes the approach to estimating the CATE for each observation using the causal forest. The results from steps 4-7 are used for local-centering, as

described above. If not provided in the causal forest (step 8), the nuisance functions are estimated internally. We use the `regression_forest` function to estimate the nuisance parameters. This function uses the honest estimation which means the prediction is based on out-of-bag observations. We state the estimation explicitly since it might be the case that a different ML method is better suited in predicting the conditional mean or the propensity score (e.g a boosting method). When using different methods, we just need to make sure that the predictions (steps 6 and 7) are again based on either out-of-bag observations or a different subsample. Theoretically, if relying completely on the causal forest we do not need to split the sample at all since the honest condition applies to each step (the nuisance parameters and the estimation of the CATE). Since we use K fold sample splitting for all other methods we apply the same subsamples when using the causal forest.

Algorithm 5: Causal Forest

Input: $Z_i = \{Y_i, D_i, X_i\}_{i \in N}$

- 1 Split sample Z into K random subsets
- 2 **for** k in $\{1, \dots, K\}$ **do**
- 3 **assign** Sample $S_a = Z \setminus S_k$ and S_k
- 4 **regress** $D_i = \hat{e}(X_i) + \hat{V}_i$, with $i \in S_a$
- 5 **regress** $Y_i = \hat{\mu}(X_i) + \hat{U}_i$, with $i \in S_a$
- 6 **estimate** $\hat{D}_i = \hat{e}(X_i)$, with $i \notin S_a$
- 7 **estimate** $\hat{Y}_i = \hat{\mu}(X_i)$, with $i \notin S_a$
- 8 **apply** Causal Forest using $Y_i, D_i, X_i, \hat{\mu}(X_i), \hat{e}(X_i)$ with $i \in S_a$
- 9 **estimate** $\hat{\tau}_k(X_i)$ with $i \in S_k$
- 10 **end**
- 11 **combine** $\hat{\tau}(X_i) = \{\hat{\tau}_1, \hat{\tau}_k, \dots, \hat{\tau}_K\}$

2.2.2 Causal Boosting:

An alternative to random forest based causal inference is given by Powers et al. (2018) who introduces boosted trees and causal multivariate adaptive regression splines (MARS). By iteratively fitting weak learners to the residuals of a model, an approximation of the function is build. The idea is to fit a causal tree in the style of Wager and

Athey (2018) while setting the basis function $\hat{G}(x, D)$ to zero. Now we estimate the residuals by $Y_i - \epsilon \times \hat{g}_k(X_i, D_i)$ and update $\hat{G}_k = \hat{G}_{k-1} + \epsilon \times \hat{g}_k$. k defines the terminal nodes from the tree and ϵ is the learning rate parameter. After K iterations we return $\hat{G}_K(x, D)$. Estimating the CATE is done by setting D to 1 for the treated observations and 0 for the control-group observations, such that:

$$\hat{\tau}(x) = \hat{G}_K(x, 1) - \hat{G}_K(x, 0). \quad (2.11)$$

Like in the causal forest the problem remains how to control for overfitting. Especially boosting methods are prone to overfit the data since the trees are not built independently. While a random forest would benefit from using more trees over which to average, in gradient boosting the number of trees is an important tuning parameter that needs to be controlled. In supervised ML we would ideally apply cross-validation. In our case, the parameter of interest is the CATE and we do not observe the true value for each observation. Hence, cross-validation does not apply here. Instead, we can do something like the honest approach from the causal forest.

Powers et al. (2018) propose to split the data into two distinct sets. The training set is used to build the causal boosting. Using the split-points and split-variables from the training set we use the covariates from the validation set, let's call it X_v , for validation and get new estimates based on D_v and Y_v for each terminal node. This procedure is done for any of the K trees, using again the residuals (this time from the validation set tree) to reestimate the terminal nodes of the next causal tree. This allows estimating a validation error for each of the original K models. The overall validation error for a causal boosting model is given by the differences of the CATE from the original vs. the validation trees.

2.2.3 Causal BART:

While (causal) boosting relies on multiplying each sequential tree by the learning rate (ϵ), the idea developed by Chipman, George, McCulloch, et al. (2010) is to estimate a posterior distribution of the prediction by explicitly setting priors for the trees and

ensemble structure (e.g. the depth of the tree, the probability of a new split). Using a Bayesian approach allows for a broader set of information than the point estimate from regression and classification methods. The Bayesian Additive Regression Trees (BART) approach is a combination of three methods: Using gradient boosting trees, a Bayesian framing for each individual tree, and Markov chain Monte Carlo (MCMC) sampling to do backfitting (using additive and generalized additive models for posterior sampling). [Hill \(2011\)](#) proposes to use such nonparametric Bayesian models to estimate treatment effects. Given strong ignorability, one way to estimate treatment effect is to estimate the response function $\mu(X_i, D_i)$. This function is estimated in one step instead of estimating two functions. Hence, the prior is set directly for the response surface. This approach is also called the S-learner - train one function and set D_i to 1 and 0 for each observation to get estimates for both potential outcomes. [Hahn et al. \(2020\)](#) extends the idea of using a Bayesian approach to estimate treatment effects but expresses the response surface as:

$$\mathbb{E}[Y_i | X_i, D_i = d] = \mu\{X_i, \hat{e}(X_i)\} + \tau(X_i) D_i, \quad (2.12)$$

where $\hat{e}(x)$ is the estimated propensity score and the functions $\mu(\cdot)$ and $\tau(\cdot)$ are independent BART priors. The inclusion of the estimated propensity score can be seen as a covariate-dependent prior to control for confounding bias. The method is specially designed to estimate the CATE from observational studies with small effect sizes and heterogeneous effects. The package we use is built on the model by [Hill \(2011\)](#) (`install_github("vdorie/bartCause")`). A package that implements the method proposed by [Hahn et al. \(2020\)](#) is in development (`install_github("socket778/XBCF")`). This package is also available for Python. Note that the causal BART produces credible intervals as a contrast to confidence intervals. They are estimated from the posterior probability function and hence rely on the prior distribution while confidence intervals are based on data only. We will only use the term confidence interval on all methods, however, we do mean credible intervals for the causal BART and (frequentists) confidence intervals for all other methods.

Algorithm 6: Causal BART

Input: $Z_i = \{Y_i, D_i, X_i\}_{i \in N}$

- 1 Split sample Z into K random subsets
 - 2 **for** k in $\{1, \dots, K\}$ **do**
 - 3 **apply** Causal BART using Y_i, D_i, X_i with $i \in S_a$
 - 4 **estimate** $\hat{\tau}_k(X_i)$ with $i \in S_k$
 - 5 **end**
 - 6 **combine** $\hat{\tau}(X_i) = \{\hat{\tau}_1, \hat{\tau}_k, \dots, \hat{\tau}_K\}$
-

2.3 Sample splitting and cross-fitting

To aim for a consistent estimator, we need to assume certain complexity conditions on the nuisance functions. Specifically, we want them to be smooth (i.e. differentiable) and the entropy of the candidate nuisance functions to be small enough to fulfill Donsker conditions (e.g. if we assume Lipschitz parametric functions or VC classes). In high-dimensional settings ($p > n$) or when using ML methods that are complex or adaptive, the Donsker conditions might not hold; see, for example, [Robins, Zhang, Ayyagari, Logan, Tchetgen, Li, Lumley, and van der Vaart \(2013\)](#), [Chernozhukov, Escanciano, Ichimura, Newey, and Robins \(2016\)](#) and [Rotnitzky, Robins, and Babino \(2017\)](#). As [Chernozhukov et al. \(2018a\)](#) noticed, verification of the entropy condition is so far only available for certain classes of machine learning methods, such as lasso and post-lasso. For classes that employ cross-validation or for hybrid methods (like the SuperLearner), it is likely difficult to verify such conditions. Luckily, there is an easy solution available: sample splitting. When splitting the sample, we can use independent sets for estimating the nuisance functions and constructing the treatment estimation equation. By using different sets, we can treat the nuisance functions as fixed functions which allow avoiding conditions on the complexity. It also allows us to use any ML method such as random forest or boosting or even an ensemble of different methods. The split-sample approach to avoid smoothness conditions dates back at least to [Bickel \(1982\)](#) and was extended to also use cross-fitting by [Schick \(1986\)](#).

To overcome a potential loss in efficiency, since only a subset of the data is used when estimating the CATE, cross-fitting is an increasingly popular approach to combine ML

methods with semi-parametric estimation problems; see, for example, [Chernozhukov et al. \(2018a\)](#), [Newey and Robins \(2018\)](#) and [Athey and Wager \(2017\)](#). We note that there are two definitions of cross-fitting. First, it is defined in the context of estimating the CATE for all observations. For example, we split the data into two folds, subset A and M. We use fold A to train the nuisance functions and then estimate the parameter of interest using subset M. Now we switch the roles of the sets, using subset M for training and subset A for estimation. As a result, we get estimates of the CATE for all observations.

The second definition is more in the spirit of averaging CATE estimates obtained from different partitions that are used for the nuisance parameter estimation. For example, let us say we have again the data as above but also an independent test set. Now we can use the procedure as before. First, we train the nuisance function on subset A and predict on subset B to get the pseudo-outcomes. We again train a regression function based on B but predict the CATE using the test set. Now we reverse the roles of A and B and get a second prediction of the CATE for the test set. The two results are now averaged to get the final estimate. In this tutorial, we will combine the two definitions of cross-fitting. First, we estimate the CATE on all observations through reversing roles of samples. Second, we use cross-fitting as an averaging tool over K folds. When referring to cross-fitting we mainly mean the latter definition.

We give an example of the benefit from cross-fitting in Figure 2.1. We show the MSE from the true treatment effect for a single estimator and the cross-fit estimator based on a 50:50 sample split. We used the R-learner as the meta-learner and create 50 Monte Carlo replications of the data using the same data generating process (DGP) which simulates a RCT and has the following properties: $N = 2000$, $X = \mathbb{R}^{10}$, $e(X) = 0.5$, and $\tau(x) = X_1 + \mathbf{1}(X_2 > 0) + W$ with $W \sim \mathcal{N}(0, 0.5)$. Using cross-fitting decreases the MSE compared to the single estimator in about 90% of the cases. We also find that the variance is smaller compared to the single estimator.

In empirical studies we do not have an independent test set and setting aside a partition might not be efficient since we lose observations for the estimation part. In

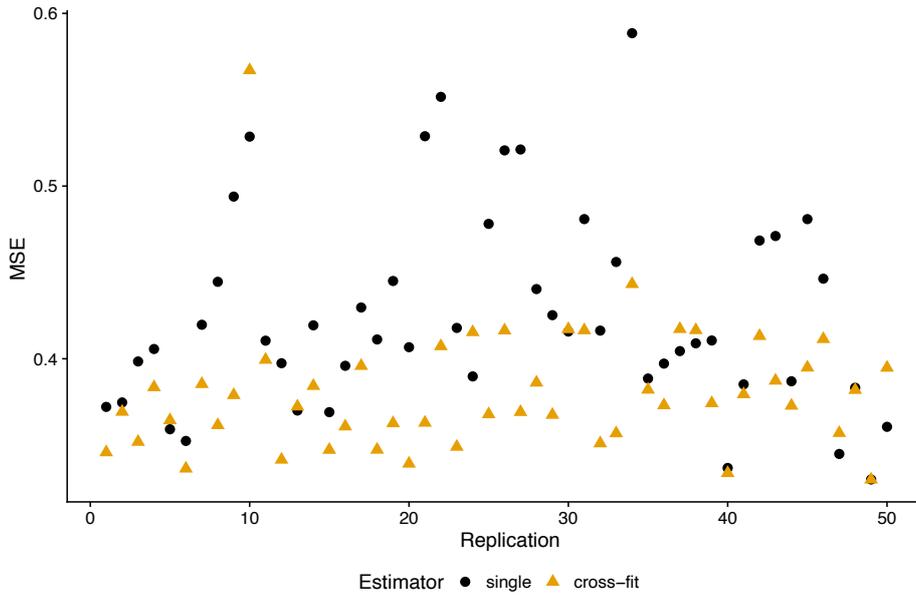


Figure 2.1: Single vs. cross-fit estimation of CATE. 

the following, we present an approach to use cross-fitting without an additional test set.

We apply 5-fold sample splitting and use 80% of the full data (denoted by Z) for training the nuisance functions (denoted by S_a) and 20% to for estimation (denoted by S_k). We propose to estimate the nuisance parameters for each of the 5 folds, using all folds but k for training and fold k to predict the conditional mean and the propensity score. We then store the estimates. As a result, we have estimates of all nuisance parameters to create the pseudo-outcomes for each observation obtained from independent samples. Hence, the above-mentioned regularity conditions should be fulfilled. Now we want to train a regression model on the pseudo-outcomes (or minimize the R-loss). Instead of using the full sample for training and prediction, we divide the sample into different parts. We assign half of the sample as the test set (denoted by S_{oob} , which is short for out-of-bag) and the other half that is used to train the regression model. Let us say we want to rely on 5-fold cross-fitting (taking the average of S_{oob} over 5 folds). We therefore split the other half of the sample into 5 folds (denoted by $S_{train} = \{S_1, S_2, \dots, S_5\}$). Using each fold to train the regression model and predict on

S_{oob} leads to 5 estimates that we average by taking the mean. Now we reverse the role of S_{train} and S_{oob} and proceed as above. We apply this procedure to the DR- and R-learner. The S- and T-learner only needs one estimation step and hence it suffices to only use two different samples (the S_a and S_k). In all other methods, we need an additional model (for example, the IPW-learner would also benefit from cross-fitting). The X-learner is quite robust even without cross-fitting. This might be because it only uses the propensity score in the last step. The advantage of the two-step sample splitting approach is that we have more observations to train the nuisance functions (in this example we have $S_a = 0.8Z$ observations instead of $0.8(Z - S_{oob})$). Figure 2.2 shows the procedure in detail. As above, we denote S_k as the fold which is used to estimate the nuisance parameters (e.g. the propensity score, the pseudo-outcomes). The estimators $\tilde{\tau}(x)$ refer to the CATE estimates obtained using S_{oob} given different folds for training. For example, $\tilde{\tau}_1(x)$ first uses S_1 for training and S_{oob} for estimation. To get estimates on the other half of the data (also denoted as S_{oob}) we use S_6 for training. Hence, $\tilde{\tau}_1(x)$ are estimates of the CATE for the whole dataset Z . A more detailed version of the cross-fitting part is shown in Figure B.2 in the Appendix.

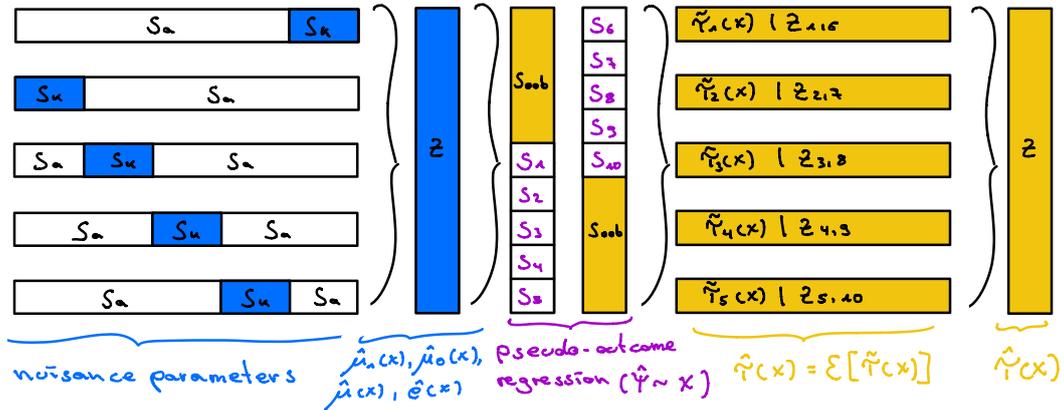


Figure 2.2: Two-step sample splitting procedure.

There might be alternative ways and averaging procedures to ensure robust estimates and prediction results. For example, we could repeat the whole procedure M times and generate different folds in the first place (the S_k folds). The result would be M

estimates for each observation of $\hat{\tau}(x)$ over which we could take the median. This might lead to more robust estimates since it takes the sample splitting uncertainty into account. See [Jacob \(2020\)](#) for a Monte Carlo study about the implications of different sample-splitting, cross-fitting, and averaging approaches for meta-learner methods. The simulation study finds that the 5 fold cross-fitting with median averaging procedure works best. Our approach mimics this procedure but changes the way to define a test set on which the cross-fitting is applied. For the meta-learners, we have to do sample splitting and cross-fitting manually while the causal forest as well as the causal boosting relies on honest estimation and does sample splitting by default. Cross-fitting, as we define it, is not implemented in any of the modified ML methods.

3 Empirical Examples

To illustrate the methods presented in the previous sections, we consider two empirical examples. In the first example, we examine the effect that microcredits have on the total amount of loans, resulting from a randomized experiment in Morocco. In the second example, we study the effect of 401(k) eligibility on accumulated assets. This example deviates from random treatment assignment and contains self-selection into a treatment. While all presented methods condition on observed pre-treatment variables to estimate heterogeneity in treatment effects they should also be able to control for confounding variables. However, methods that use the propensity score should be more suited to eliminate the selection bias. For each method, we estimate the CATE and provide confidence intervals. We also show how to link the CATE to observed covariates for further analysis. In both examples, we apply the two-step sample splitting with a cross-fitting approach for the DR- and R-learner.

3.1 Effect of microcredits on borrowing

We start with an empirical dataset to analyze the effect of microcredit availability on borrowing activities such as the amount of loans (see [Crépon et al. \(2015\)](#) for a

recent study using this dataset). Looking beyond the ATE and finding heterogeneous treatment effects is important to target specific groups and to make better policies. The allocation of treatment was randomized between 162 villages in Morocco. The villages were divided into pairs with similar observable characteristics. Then the treatment was randomly assigned to one of the pair while the counterpart was assigned to the control group. Treatment as microcredit availability in this context means that between 2006 and 2007 a microfinance institution started operating only in the treated villages. In 2009, 5,551 households were surveyed in a follow-up study. We use the results from this survey to estimate conditional average treatment effects using different methods and also show some strategies to get some insight into which characteristics are responsible for heterogeneity in treatment effects. We select the following pre-treatment covariates that are observed characteristics for each household such as the age of household's head, number of adults, number of children, total number of members in a household, indicators for households doing animal husbandry, other non-agricultural activity, household spouse responding to the survey, the education of the head and having an outstanding loan over the last year. Table 3.1 shows the mean value for some covariates. They are categorized by all observations, the treated and the control group. Given these unconditional means, we see that the amount of loans for the treatment group is much higher (2,930) than for the control group (1,802). We also see that the mean of the characteristics is quite balanced across the two groups. This reassures us that the treatment assignment was randomly selected and that there are no confounding variables that lead to self-selection into treatment. While there are small differences in some covariates, this is not concerning since all methods that we apply make use of the propensity score or condition on the covariates to estimate the treatment effect only on similar subgroups. For example, more people in the treatment group already have a loan in the last twelve months. We can estimate the probability of being in the treatment group given this variable and reweigh the treatment and control group to adjust for these differences. The dataset and R-code for the microcredit analysis can be found here [emp](#).

Table 3.1: Descriptive statistics of households (mean)

	All	Treated	Control
<i>Outcome Variable</i>			
Total Amount of Loans	2,359	2,930	1,802
<i>Baseline Covariates</i>			
Number of Household Members	3.879	3.872	3.886
Number of Children	1.266	1.261	1.272
Head Age	35.976	35.937	36.014
Declared Animal Husbandry Self-employment Activity	0.415	0.426	0.404
Declared Non-agricultural Self-employment Activity	0.146	0.129	0.164
Borrowed from Any Source	0.210	0.224	0.196
Spouse of Head Responded to Self-employment Section	0.067	0.074	0.061
Member Responded to Self-employment Section	0.044	0.048	0.041

We use three different ML algorithms to estimate the nuisance functions and to map the covariates on the pseudo-outcome (for the DR- and X-learner) or to minimize the R-loss function. Table 3.2 shows the coefficients for each ML algorithm obtained through cross-validation in the SuperLearner. The loss-function is the non-negative least squares based on the Lawson-Hanson algorithm which works for both Gaussian and binomial outcomes. We find that the neural network gets the highest weight for all functions except for the propensity score where the random forest has a slightly higher weight. Based on these weights, we only use the neural network and the random forest for the construction of the bootstrapped confidence intervals. This is mainly since we believe that even with a bootstrapped sample, the weights of the algorithms for each function will not change dramatically. Excluding the boosting algorithms decreases the computation time by about 50%.

Table 3.3 shows a summary for the heterogeneous treatment, namely the effect for the 20% least affected, the ATE, and the effect for the 20% most affected observations. Especially for the quantiles, we find differences in the estimates given the methods

Table 3.2: Weights of ML methods.

	$\hat{e}(X)$	$\hat{\mu}_0(X)$	$\hat{\mu}_1(X)$	$\hat{\mu}(X)$	DR	R
Boosting	0.00	0.00	0.00	0.00	0.00	0.11
Neural Network	0.46	0.73	0.70	0.80	0.90	0.89
Random Forest	0.54	0.27	0.30	0.20	0.09	0.00

Table 3.3: CATE results for different methods.

Category	Method	20% Least	ATE	20% Most
Meta-Learner	DR-learner	-15.4	1,119.8	3,057.6
	R-learner	84.9	1,081.1	2,237.5
	T-learner	198.4	1,152.5	2,470.3
	X-learner	869.9	1,137.2	1,379.7
Modified ML	Causal BART	593.8	1,132.3	2,304.7
Methods	Causal Forest	296.6	1,129.6	2,329.6

that we consider. This holds for the lower 20% where the effect ranges from -15 to 869 as well as for the 20% most affected with the lowest treatment effect from the X-learner with $1,379$ and the highest estimate from the DR-learner with $3,057$. The high value in the upper quantile from the DR-learner is because it predicts more extreme values at the tail of the distribution. The DR-learner also has the highest variance in terms of treatment effect with a range in number of loans from -15 to $3,057$ on average for the specific quantiles. The ATE is around $1,100$ for all methods and there is no large difference between them. Figure 3.1 shows the treatment effect for each observation, sorted by the size of the effect. We also show 95% confidence

intervals (CI). They are estimated via bootstrapping with $B = 500$ replications. Here we adopt the procedure for the construction of CI's from [Künzel et al. \(2019\)](#). We first split our entire dataset into a training and validation set. We use the training set for bootstrapping by creating a sample from the training data of the same size with replacement. For each meta-learner and each bootstrap sample from the training data, we use the test set to estimate the CATE. We repeat this procedure $K = 5$ times looping through all k subsets and define them as the test set. In total, we end up having B estimates for each observation on the whole data. Now we calculate the standard deviation ($\hat{\sigma}$) for each observation which we use to generate a lower and upper bound around the CATE estimates $[\hat{\tau}(x) - q_{\alpha/2}\hat{\sigma}; \hat{\tau}(x) + q_{1-\alpha/2}\hat{\sigma}]$. Especially for the meta-learners, we have a high variance between the bootstrapped samples indicating that even if the CATE is different, there might not be a significant heterogeneity. This is also in line with the estimates from the causal BART and the causal forest that show tighter bounds but also an almost flat CATE curve. To calculate the CATE, denoted by $\hat{\tau}(x)$, we use the whole training data, not a bootstrapped version, and proceed as described in the pseudo-code for the specific meta-learner.

Figure 3.1 shows quite similar values for at least four of the methods. Only the DR- and R-learner have heavier tails for higher CATE estimates. The T-learner has the widest confidence intervals while all other methods show a similar range. Treatment effects based on the DR- and R-learner are heterogeneous, at least for the 20% least and most affected. The most homogeneous prediction comes from the causal BART and the X-learner. Their estimates of the CATE are quite similar. The causal forest also shows an increasing slope of the point estimates but wide confidence intervals for the most affected observations. Based on these results, there is no clear evidence of treatment effect heterogeneity.

In Figure 3.1, we sorted the treatment effect by its size for each method. This does not necessarily mean that all methods have the same order. To look into the order of the CATE based on each method, we show the correlation of the CATE among them in Figure B.3. We show the Bravais-Pearson correlation coefficient (ρ), a histogram of the CATE, and correlation ellipses. It is reassuring that all methods are positively

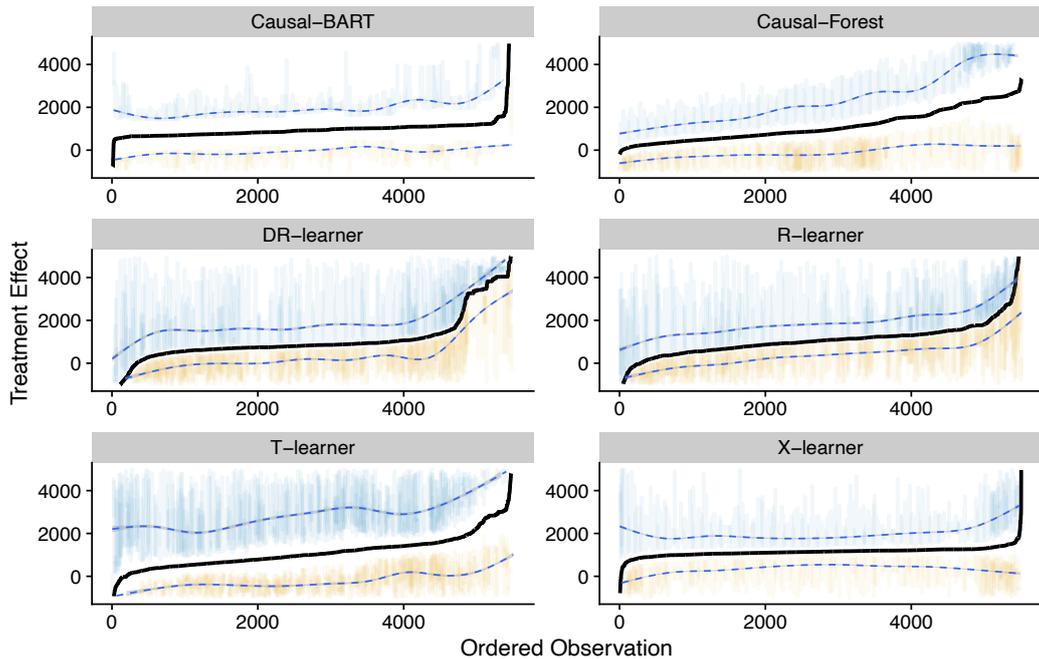


Figure 3.1: Microcredit: Observations sorted by level of treatment effect. \mathcal{O}_{sort}

correlated. The highest correlation is between the doubly-robust and the R-learner ($\rho = 0.57$) as well as between the T- and X-learner ($\rho = 0.5$). The smallest correlation appears between the causal BART and the R-learner with a correlation coefficient of 0.15.

If we believe that there is at least some difference in the effect between the least and most affected observations, then we can look at the average characteristics of these groups to understand what are potential drivers for the heterogeneity. Here we adopt a simple approach introduced by [Chernozhukov, Demirer, Duflo, and Fernandez-Val \(2018b\)](#), namely the Classification Analysis. The idea is to regress the least and most affected groups on some pre-chosen characteristics ($g(X)$) with G_q being the observations given a specific group of the treatment effect:

$$\delta_{least} = E[g(X)|G_{least}] \quad \text{and} \quad \delta_{most} = E[g(X)|G_{most}].$$

Table 3.4: Classification results for DR-learner and causal forest.

	DR-learner			Causal Forest		
	Most Affected	Least Affected	Difference	Most Affected	Least Affected	Difference
Head age	19.19	46.92	-27.73	10.25	46.80	-36.55
	(17.81,20.58)	(45.53,48.30)	(9.271,11.22)	(45.82,47.77)	(-37.93,-35.17)	(-55.00,-51.49)
	-	-	[0.000]	-	-	[0.000]
Non-agricultural	0.118	0.186	-0.068	0.073	0.136	-0.064
self-employed	(0.097,0.139)	(0.165,0.207)	(0.055,0.091)	(0.118,0.154)	(-0.089,-0.038)	(0.121,0.232)
	-	-	[0.000]	-	-	[0.000]
Borrowed from	0.138	0.338	-0.201	0.050	0.388	-0.338
any source	(0.113,0.162)	(0.314,0.363)	(0.028,0.072)	(0.366,0.411)	(-0.370,-0.307)	(-0.351,-0.219)
	-	-	[0.000]	-	-	[0.000]

Notes: 90% confidence interval in parenthesis and p-values in brackets.

Here we focus on the head age, the probability of being self-employed in a non-agricultural sector, and whether someone had an outstanding loan over the past 12 months (borrowed from any source). In Table 3.4 we estimate the average of the characteristics for the two groups as well as if there is a significant difference between the groups. We show results for two methods, the doubly-robust meta-learner (DR-learner) and the causal forest. Detailed CLAN results that include all methods can be seen in the Appendix (C.1). For both methods, we find a significant difference in head age, probability of being self-employed in a non-agricultural sector, and probability of having a loan. The most affected people seem to be younger. Low values in the head age can arise since many people did not respond to that question and got a value of zero. However, we believe that the non-respondents are missing at random. This allows us to interpret the difference between the two. Looking at employment, we find diverse effects. Interpreting results from the DR-, X-learner, and causal forest we find that people who benefit most from microcredits are those who do not work in the non-agricultural sector. Results from the R-, T-learner and the causal BART suggest the other way around. We note that the result from the T-learner is not statistically

significant. The absolute magnitude in the probability difference is rather small which is why we do not interpret this variable as a driver for treatment effect heterogeneity. We also find that people who already have a loan (with a higher probability) are less affected by microcredit availability. There are other possibilities to investigate which covariates might be drivers for effect heterogeneity. For example, if a tree-based method should be the best method for mapping the covariates on the treatment effects then we could use variable importance plots to see which variables (at a certain split in a tree) increase the variance between two leaves. If a variable is (randomly) chosen for a split and the mean values in the two resulting nodes are quite the same as before the split then this variable might not be very useful to explain the heterogeneity. We can also apply partial dependence plots to see how the treatment effect changes if we change one variable.

3.2 Effect of 401(k) eligibility on accumulated assets

While the microcredit data is based on a randomized controlled trial, the eligibility of a 401(k) pension plan is not. Only some firms offer access to a 401(k) and hence there is self-selection into treatment. It might be the case that more educated people chose firms that provide a 401(k) pension plan and that they have higher financial assets in the first place. [Poterba and Venti \(1994\)](#) argue that conditioning on observed characteristics, like the income, can restore the random assignment mechanism. The dataset we use is the same as in [Chernozhukov and Hansen \(2004\)](#) which is based on the 1991 Survey of Income and Program Participation. We are interested in the question if 401(k) eligibility, our treatment variable, has an impact on accumulated assets (here we use the net financial assets as the outcome variable). We control for income and other variables related to the job choice that may have an impact on treatment assignment and assets. In total, we have 9,915 observations and 13 covariates consisting of age, family size, income, years of education, and indicator variables for married, two-earner status, defined benefit pension status, homeownership, and IRA participation. We split

the dataset into 5 parts and proceed as described in the sample splitting section (2.3).

The dataset and R-code for the 401(k) analysis can be found here .

Table 3.5: Descriptive statistics of observations (mean)

	All	Treated	Control
<i>Outcome Variable</i>			
Net financial assets	18,052	30,347	10,788
<i>Baseline Covariates</i>			
Age	41.06	41.48	40.81
Income	37,201	46,862	31,494
Years of education	13.21	13.76	12.88
Proportion of being married	0.60	0.67	0.56
Proportion of two-earners	0.38	0.48	0.31
Proportion of home-ownership	0.63	0.74	0.57

Table 3.5 shows the mean values for the net financial assets and for some pre-treatment covariates. The amount of assets is higher in the treatment group than in the control group. Concerning the self-selection into treatment, we see that some characteristics are different between the treatment and control group. For example, the proportion of home-ownership, years of education, and income is higher for treated people. There are further reasons to believe that such characteristics are positively correlated with financial assets. In this case, we have to control for such variables to account for the self-selection into treatment.

Table 3.7 shows the estimated CATE for the 20% least affected and 80% most affected as well as the ATE. The ATE is positive and ranges from 7,120 to 9,055, depending on the method. Its variance between the methods is quite low, compared to the estimates for the least and most affected groups. While the T- and X-learner predict a negative effect from 401(k) eligibility on financial assets for the lowest group, all other methods predict a positive effect. The highest affected group has values from 9,806 (from the DR-learner) to 25,326 (from the T-learner). The causal forest predicts

Table 3.6: Weights of ML methods.

	$\hat{e}(X)$	$\hat{\mu}_0(X)$	$\hat{\mu}_1(X)$	$\hat{\mu}(X)$	DR	R
Boosting	0.36	0.08	0.05	0.08	0.00	0.00
Neural Network	0.14	0.09	0.06	0.06	0.49	0.33
Random Forest	0.50	0.82	0.89	0.85	0.51	0.67

Table 3.7: CATE results for different methods.

Category	Method	20% Least	ATE	20% Most
Meta-Learner	DR-learner	4,998	7,120	9,806
	R-learner	4,250	7,410	11,320
	T-learner	-4,171	7,579	25,326
	X-learner	-285	7,631	18,648
Modified ML	Causal BART	2,466	9,055	21,525
Methods	Causal Forest	5,210	8,228	12,360

values with the lowest heterogeneity. Except for the causal forest, all other learners predict extreme values in the tails of the CATE. If we would use a majority vote from all the methods to interpret the estimated effects, then it is reassuring that everyone has a positive effect from the 401(k) eligibility as can be seen in Figure 3.2. Given the wide confidence intervals, the evidence of treatment effect heterogeneity is not so clear.

Figure B.4 shows the correlation of the CATE between the different methods. We find that the methods are highly correlated with each other. The lowest correlation is between the DR- and T-learner with a correlation coefficient of $\rho = 0.64$ while the

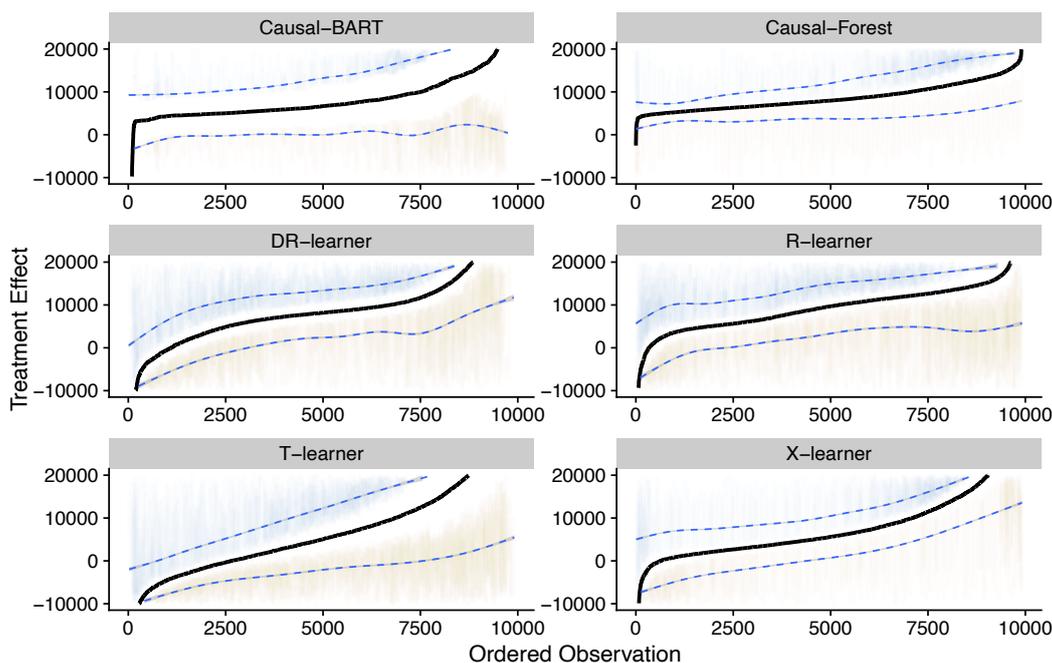


Figure 3.2: 401k: Observations sorted by level of treatment effect. [sort](#)

highest correlation is between the causal BART and causal forest ($\rho = 0.85$). The reason why the estimated CATE is more similar might be the large sample size of $N = 9915$.

Since the data does not come from a randomized controlled trial, we expect the distribution of the covariates to be different given treatment status. To see this, we plot the distribution of age, years of education, marital status, income, homeowner status, and two-earner status in Figure 3.3. As we already saw from Table 3.5, treated people have a higher income, slightly more years of education and, among others, are more often homeowners. To see if the estimated propensity score can catch the differences, we can look at a weighted histogram. What we do is weigh the counts in each variable by the inverse of the propensity score. If someone is in the treatment group, we weigh by $1/\hat{e}(x)$ and the control group observations by $1/(1 - \hat{e}(x))$. The result is shown in Figure 3.4. Indeed, we see that the distributions are quite similar after reweighing with the propensity score.

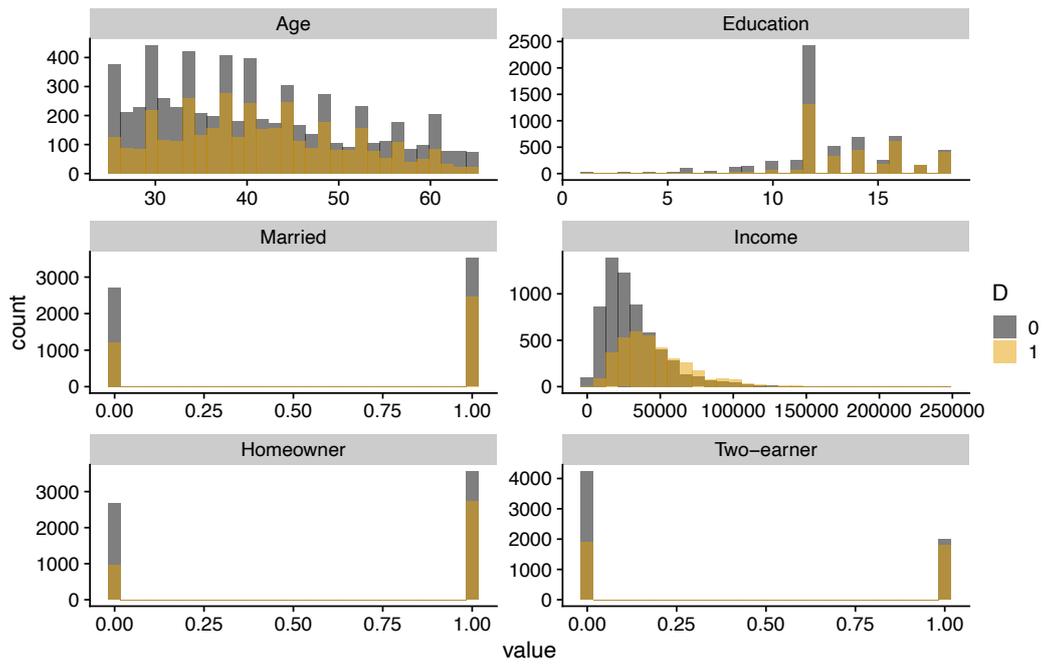


Figure 3.3: Unweighted distribution of variables given treatment status.  

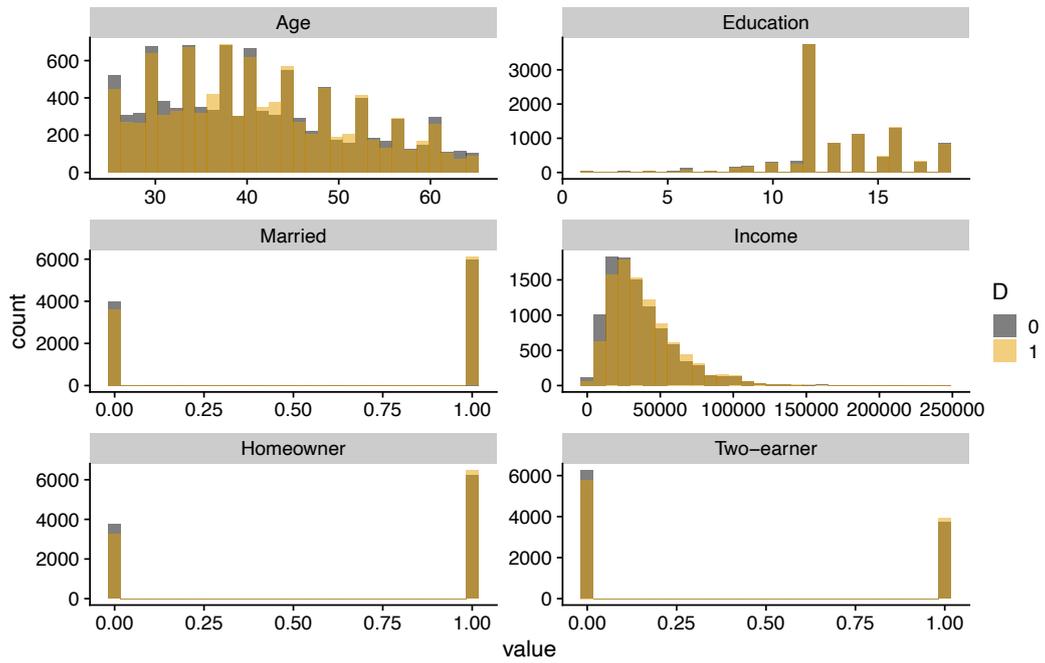


Figure 3.4: IPW weighted distribution of variables given treatment status.  

4 Simulated Data

Since the true treatment effect is never known beforehand, we provide a simulation to evaluate different approaches in terms of performance for parameter estimation. The data-generating process allows controlling the number of observations, dimensionality, and the distributions of the variables. The possibility to specify datasets for different simulations and scenarios helps to investigate the methods used in this tutorial. Note that simulated data often lack realistic data structures. An alternative is to rely on synthetic data where only the treatment effect is artificially added. Since a simulation study in the type of a Monte Carlo study is not the main focus of this tutorial, we will only use two simulated data generating processes. The purpose is to give an idea of how to simulate data and test different methods. Instead of relying on purely artificial data, [Wendling, Jung, Callahan, Schuler, Shah, and Gallego \(2018\)](#) creates synthetic data based on real covariates and a treatment assignment mechanism. Only the outcome is simulated based on non-parametric models of the real outcome.

4.1 Data Generating Process

The basic model used in this tutorial is a partially linear regression model based on [Robinson \(1988\)](#) with extensions:

$$Y = \tau(X_i)D + \mu_0(X_i) + U, \quad \mathbb{E}[U|X, D] = 0, \quad (4.13)$$

$$D = e(X_i) + V, \quad \mathbb{E}[V|X] = 0, \quad (4.14)$$

$$\tau(X_i) = t(X_i) + W, \quad \mathbb{E}[W|X] = 0. \quad (4.15)$$

Let Y be the outcome variable. $\tau(X_i)$ is the true treatment effect or population uplift, while D is the treatment status. The vector $X = (X_1, \dots, X_p)$ consists of p different features or covariates and U , V and W are unobserved covariates which follow a random normal distribution $= N(0, 1)$.

Equation 4.14 is the propensity score. In the case of completely random treatment assignment, the propensity score is constant for all units, and, if equally distributed,

then $e(X_i) = 0.5$. The covariates X are generated from a random multivariate normal distribution ($N(0, 1)$). Note that all values are continuous. In business applications, discrete values (categorical variables) are very common. For the data generation process as well as for the evaluation of most models, it would make no difference if such variables are present. This is because vanilla machine learning methods can handle categorical variables quite well. An exception is the causal forest where one has to use one-hot encoding, to transform the variable into dummies. Next, we describe the generation of the functions in detail.

Covariates (X)

1. Generate a random positive definite covariance matrix Σ based on a uniform distribution over the space $p \times p$ of the correlation matrix. Let $p = 20$.
2. Scale the covariance matrix. This equals the correlation matrix and can be seen as the covariance matrix of the standardized random variables $\Sigma = \frac{X}{\sigma(X_i)}$.
3. Generate random normal distributed variables $X_{N \times p}$ with mean = 0 and variance = Σ .

The function $\mu_0(X_i)$ is calculated using a linear function with interaction terms and contains the following covariates:

$$\mu_0(X) = X_1 \otimes X_2 + X_3 \otimes X_4 + X_5. \tag{4.16}$$

All covariates are normal distributed except X_5 which only takes four values, namely $\{-0.2, 0, 0.2, 0.6\}$. Next, we describe how to build the function $e(X_i)$ as well as how to create heterogeneous treatment effects. A varying treatment effect implies that its strength differs among the observations and is therefore conditioned on some covariates X . Regarding the treatment assignment, two settings are considered. Setting 1 assumes D to be completely randomly assigned among the observations. In this case, D is just a vector of random numbers with values 0 or 1. In setting 2, the treatment assignment is dependent on the covariates. The binary treatment assignment is generated

through a Bernoulli function. This implies per default a sort of uncertainty or random error. Even if the probability from the propensity score is 90% for $D = 1$, there is still a 10% chance that it is generated to be zero. The functions are generated as follows:

<u>Treatment Assignment (e_0)</u>	
<u>Setting 1: e_0</u>	
$D \stackrel{ind.}{\sim} \text{Bernoulli}(e_0),$	with $e_0 = 0.5$
<u>Setting 2: $e(X_i)$</u>	
<ol style="list-style-type: none"> 1. Dependence is non-linear (through interaction terms): $a(X_i) = X_1 \otimes X_2 + X_3 \otimes X_4$. 2. Calculate the probability distribution for the vector a from the normal distribution function: <div style="text-align: center; margin: 10px 0;"> $e(X_i) = \Phi\left(\frac{a - \mu(a)}{\sigma(a)}\right) = \frac{1}{2} \left[1 + \text{erf}\left(\frac{a - \mu(a)}{\sigma(a)\sqrt{2}}\right) \right]$ </div> <p style="text-align: center; margin: 0;">Here $\mu(a)$ denotes the mean of a and $\sigma(a)$ the standard deviation.</p> 3. Apply a random number generator from a Binomial function $B\{N, e(X_i)\}$ with probability for success = $e(X_i)$. This creates a vector $D \in \{0;1\}$ such that $D \stackrel{ind.}{\sim} \text{Bernoulli}\{e(X_i)\}$. 	

Regarding the treatment effect, we also consider two different settings. First, $\tau(X_i)$ depends linear on covariates X , and second, $\tau(X_i)$ has a non-linear, more complex form concerning the covariates. In both settings, we can examine heterogeneous treatment effects. The vector $b = \frac{1}{l}$ with $l \in \{1, 2, \dots, p\}$ represents weights for every covariate.

Treatment Effect ($\tau(X_i)$)

Setting 1: linear dependence

$$\tau(X_i) = 0.6X_1 + 0.6X_2 + 0.6X_3 + 0.6X_4 + X_5 + W \quad \text{with} \quad W \sim \mathcal{N}(0, 0.5).$$

Setting 2: non-linear dependence

$$\tau(X_i) = \sin(X_{1:3} \times b_{1:3}) + 1.5 \cos(X_4) + X_5.$$

The simulated data that include the true treatment effect can be found here:  [sim](#).

4.2 Results

To evaluate the different methods, we consider two data generating processes (DGP). Setting 1 is a randomized controlled trial with a constant propensity score of 0.5, while the treatment effects depend linear on covariates. In setting 2, we consider confounding variables, namely that the treatment probability now depends on covariates (through interactions of covariates) while the treatment effect depends non-linear on the covariates. In both settings, we set $N = 2000$ and $p = 20$. We use up to 5 variables to generate the different variables and the treatment effects while all other variables have no dependence on any function. They are spurious and the hope is that the ML methods find the important variables while excluding the others. Table 4.2 shows the mean squared error (MSE) for all considered methods and both settings. We list to different versions of the DR- and R-learner. The first is the in-sample estimator where the regression and estimation of the CATE based on the pseudo-outcome or R-loss is done on the same sample. Only the nuisance functions are regressed and estimated on different samples. This is in line with the sample splitting theory. In the last step, we just want to have a good approximation of the CATE which is why we can use the same sample for training and prediction. Note that the DR-learner already estimates the

CATE in the pseudo-outcome. Using the whole sample should increase the prediction power.

In practice, however, we find that it might be better to split the sample again and not use the same sample for training and prediction. The reason is the following: If the predictions of the nuisance functions are not perfect, the pseudo-outcome deviates from the true CATE. The deviation becomes clearer with a higher estimation error and also if there are extreme values in the propensity score. Using different samples in the last step aims to smooth the function and discard outliers. This approach adds sample splitting (the two-step sample splitting without cross-fitting). Here we apply this approach with cross-fitting. This means we not only want to have different samples for training and prediction but also want to average the prediction fold over different training samples. Therefore we split the sample into 6 folds (the proportions are 10% for the first 5 folds and 50% for the last one). We use fold 1 to 5 individually to train regression functions and predict on all fold 6. Then we average the 5 estimates for fold 6. Now we reverse the role, combining fold 1 to 5 and split fold 6 in 5 parts. We proceed as above. We call the sample split estimator simply cross-fit estimator. Table 4.2 shows the results for both versions. Using the cross-fit version we can decrease the MSE by at least 50%. In simulations, we find that even a 50:50 split where we use 50% for training and predict on the other half can decrease the MSE. The cross-fit version turns out to further decrease the MSE in simulations with different DGP's. For completeness, we show the procedure of the 50:50 split approach in Figure B.1 in the Appendix.

Table 4.1 shows that in setting 1 the tree-based methods (boosting and random forest) perform best in predicting the propensity score while the neural network does better in the regression tasks. The lasso only gets significant weight in the treatment effect regression. The lasso is excluded when applying the R-learner since in a linear setting the loss-function slightly differs from the more general non-parametric one. If the data generating process becomes more complex, the lasso method becomes less important shifting weights towards the neural network. In setting 2, the tree-based methods are most important in all tasks but for the treatment-effect regression based on

the R-learner. We also experimented with excluding the neural network and found that in the linear setting, more weight is based on the lasso, while in the non-linear setting, the tree-based methods are superior. Since all methods are important in at least one task, we include all methods but the lasso when creating bootstrapped confidence intervals.

Table 4.1: Weights of ML methods.

	$\hat{e}(X)$	$\hat{\mu}_0(X)$	$\hat{\mu}_1(X)$	$\hat{\mu}(X)$	DR	R
<i>Setting 1</i>						
Boosting	0.91	0.11	0.27	0.27	0.26	0.12
Lasso	0.00	0.00	0.01	0.00	0.34	—
Neural Network	0.00	0.83	0.70	0.68	0.00	0.75
Random Forest	0.09	0.06	0.02	0.04	0.39	0.12
<i>Setting 2</i>						
Boosting	0.40	0.62	0.72	0.72	0.14	0.13
Lasso	0.00	0.00	0.00	0.00	0.09	—
Neural Network	0.00	0.07	0.06	0.07	0.13	0.39
Random Forest	0.60	0.31	0.22	0.21	0.65	0.47

Figure 4.1 and 4.3 show the sorted treatment effect heterogeneity with 95% confidence intervals for setting 1 and 2, respectively. While the causal BART method produces the lowest MSE, it has higher credible intervals than the causal forest. Figure B.7 in the Appendix shows boxplots of all methods and their variation. The blue line indicates the true ATE, hence we can see how accurate all methods are to predict the ATE. We find that all methods are unbiased if the DGP is linear. The bias increases if the functions are more complex as shown in Figure B.8. Figure B.7 and B.8 also shows the decrease in outliers for the DR- and R-learner when we apply additional sample splitting and cross-fitting. We do not observe these outliers for the X-learner. In Figure 4.2 and 4.4, we plot scatterplots for the estimated vs. the true CATE. The blue line indicates a linear regression estimate with pointwise confidence intervals (around the

Table 4.2: MSE for different methods. \mathbf{Q}_{res}

Category	Method	MSE Setting 1	MSE Setting 2
Meta-Learner	DR-learner (in-sample)	2.68	3.42
	DR-learner (cross-fit)	1.11	1.17
	R-learner (in-sample)	2.36	2.58
	R-learner (cross-fit)	0.80	1.34
	T-learner	1.17	2.09
	X-learner	0.58	1.10
Modified ML	Causal BART	0.55	0.48
Methods	Causal Forest	0.86	1.34

Notes: The term oob denotes that the sample is split while in-sample denotes the use of the whole sample.

mean) for each method. As we have seen from the MSE, the causal BART method performs best over the whole interval and in both settings. One observation is that the meta-learners estimate the CATE with higher variance (and potentially producing more outliers that need to be controlled for) than the two modified ML methods. The T-learner has the highest variance in both settings while the DR- and R-learner show the second-highest variance in setting 1 and 2. Looking at the correlation of the methods, we find that the highest correlation is between the causal BART and causal forest ($\rho = 0.85, 0.81$ for the two settings). In general, the correlation is quite high in both settings (ranging from $\rho = 0.64$ to $\rho = 0.85$). However, we do not see any improvement in the correlation in functions that are easier to estimate.

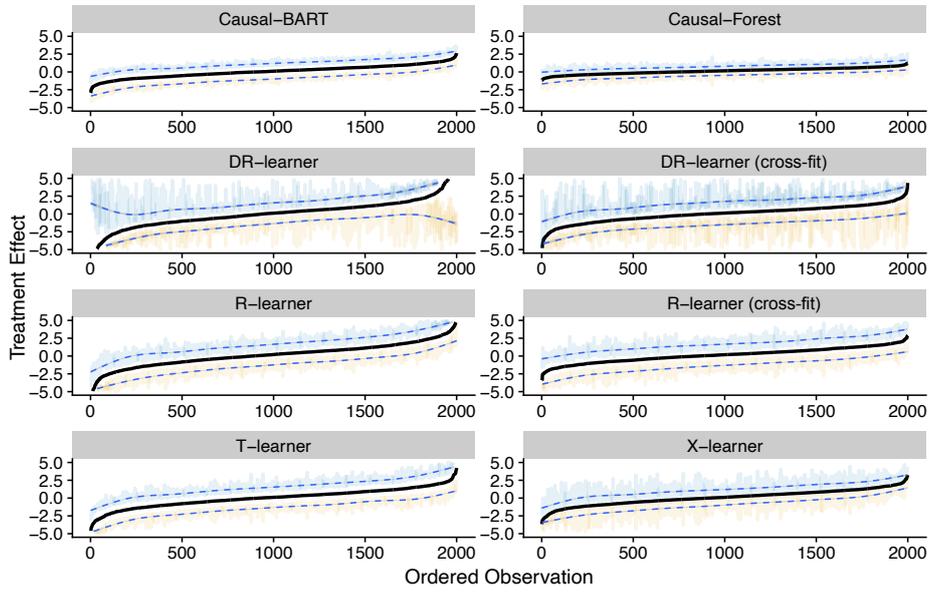


Figure 4.1: Setting 1: Observations sorted by treatment effect. \mathcal{Q}_{S1}

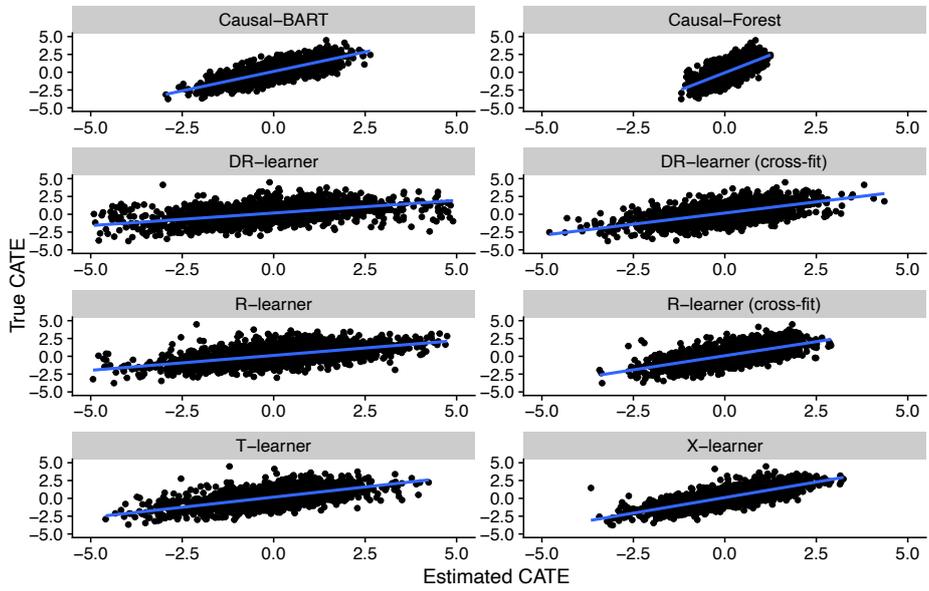


Figure 4.2: Setting 1: Scatterplot of estimated and true CATE. \mathcal{Q}_{T1}

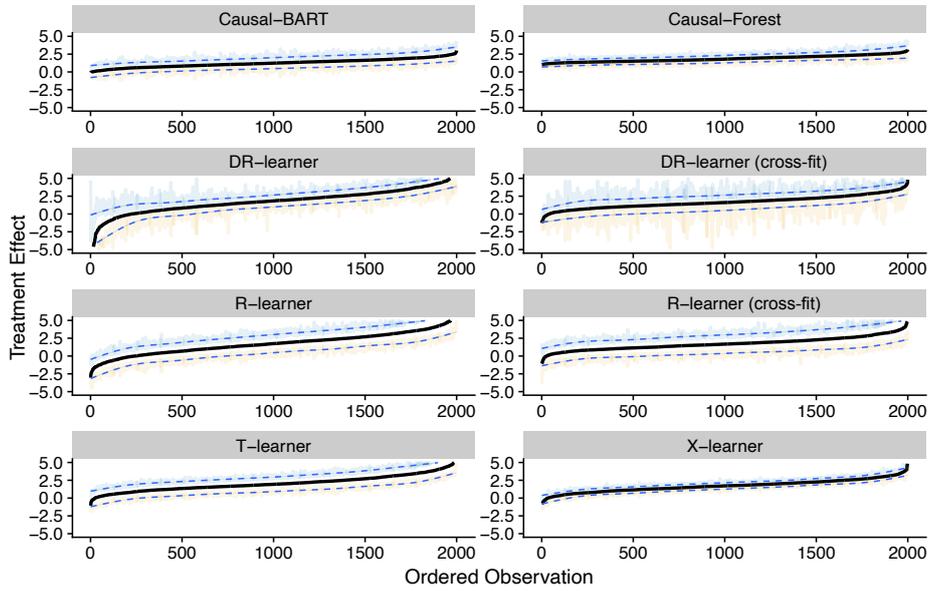


Figure 4.3: Setting 2: Observations sorted by treatment effect. \mathcal{Q}_{S_2}

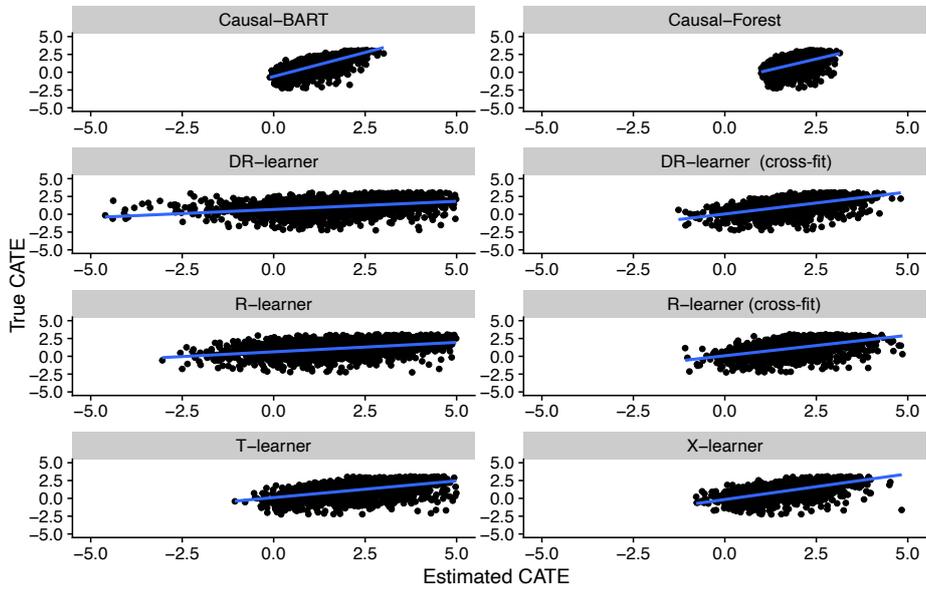


Figure 4.4: Setting 2: Scatterplot of estimated and true CATE. \mathcal{Q}_{T_2}

5 Conclusion

In this tutorial, we present novel methods to estimate the conditional average treatment effect using machine learning methods. We categorize the methods into two branches. First, so-called meta-learners, that make use of off-the-shelf machine learning methods by creating a transformed outcome to estimate the CATE. They are flexible in the choice of the machine learning method as long as they converge with a specific rate. For example, we can use classification and regression trees, random forest, boosting methods, and even neural networks. The second branch contains machine learning methods that are specific designed to estimate the CATE. These methods rely on trees or an ensemble of trees like the generalized random forest, causal boosting, and a Bayesian approach using additive regression trees. The use of meta-learners needs special care because they are quite flexible in the choice of the ML method and also concerning sample splitting. We, therefore, provide pseudo-code along with R-code for many of such meta-learners and show how they can be used to estimate the CATE on the whole dataset. We also demonstrate how to use the second branch of methods by integrating the packages in R-code that uses the same data structure as the meta-learners. When possible we apply cross-fitting as an averaging procedure of a subset of the data conditional on different training folds.

To demonstrate the strength and differences of all the methods that we consider, we present four examples. Two empirical examples, the first from a randomized control trial and the second from an observational study. The third and fourth examples contain simulated data where the true treatment effect can be observed and hence compared with the estimates from all the methods. In the empirical examples, we find strong evidence of positive treatment effects for each observation while significant heterogeneity in the effects is not that clear. This is mainly if we base the conclusion on calculated confidence intervals via the bootstrap or credible intervals. We do, however, find differences in the width of the confidence intervals and also in the CATE prediction among the methods. These differences also occur in the simulated data. We, therefore, recommend that practitioners not rely on only one method but rather use multiple

methods and compare the results. One should also carefully think about the different tuning parameters that can be set when using machine learning methods. Depending on the method there can be a variety of options to consider. We try to avoid the problem of manually selecting such parameters through cross-validation and the selection of different ML methods for each nuisance function. Sample splitting and cross-fitting is a further necessary step to get robust and accurate estimates among the methods. One observation from this simulation is clearly that the meta-learners can improve in terms of MSE with simpler functions. We note that the results heavily depend on the chosen ML methods. Through applying different ML methods in the Super Learner we find that the selection of the best ML method depends on the data generating process and varies across the functions. For example, if the data structure is quite complicated and non-linear, a model based on the lasso might not be the best choice. Including more ML methods could improve the prediction accuracy depending on the data generating process. Using two-step sample splitting with cross-fitting further improves the prediction.

References

- Susan Athey. The impact of machine learning on economics. In *The economics of artificial intelligence*, pages 507–552. University of Chicago Press, 2019.
- Susan Athey and Stefan Wager. Efficient policy learning. ArXiv preprint, <https://arxiv.org/abs/1702.02896>, 2017.
- Susan Athey, Stefan Wager, and Julie Tibshirani. Generalized random forests. *Annals of Statistics*, 47(2):1148–1178, 2019. doi: 10.1214/18-AOS1709.
- Peter J Bickel. On adaptive estimation. *Annals of Statistics*, 10(3):647–671, 1982. doi: 10.1214/aos/1176345863.
- Victor Chernozhukov and Christian Hansen. The effects of 401 (k) participation on the wealth distribution: an instrumental quantile regression analysis. *Review of Economics and statistics*, 86(3):735–751, 2004. doi: 10.1162/0034653041811734.
- Victor Chernozhukov, Juan Carlos Escanciano, Hidehiko Ichimura, Whitney K Newey, and James M Robins. Locally robust semiparametric estimation. ArXiv preprint, <https://arxiv.org/abs/1608.00033>, 2016.
- Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1):C1–C68, 2018a. doi: 10.1111/ectj.12097.
- Victor Chernozhukov, Mert Demirer, Esther Duflo, and Ivan Fernandez-Val. Generic machine learning inference on heterogenous treatment effects in randomized experiments. ArXiv preprint, <https://arxiv.org/abs/1712.04802>, 2018b.
- Hugh A Chipman, Edward I George, Robert E McCulloch, et al. Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010. doi: 10.1214/09-AOAS285.

- Bruno Crépon, Florencia Devoto, Esther Duflo, and William Parienté. Estimating the impact of microcredit on those who take it up: Evidence from a randomized experiment in morocco. *American Economic Journal: Applied Economics*, 7(1): 123–50, 2015. doi: 10.1257/app.20130535.
- Microsoft Research EconML. EconML: A Python Package for ML-Based Heterogeneous Treatment Effects Estimation. <https://github.com/microsoft/EconML>, 2019. Version 0.x.
- Qingliang Fan, Yu-Chin Hsu, Robert P Lieli, and Yichong Zhang. Estimation of conditional average treatment effects with high-dimensional data. ArXiv preprint, <https://arxiv.org/abs/1908.02399>, 2019.
- Max H Farrell, Tengyuan Liang, and Sanjog Misra. Deep neural networks for estimation and inference. *Econometrica*, 89(1):181–213, 2021.
- Jared C Foster, Jeremy MG Taylor, and Stephen J Ruberg. Subgroup identification from randomized clinical trial data. *Statistics in medicine*, 30(24):2867–2880, 2011.
- Rina Friedberg, Julie Tibshirani, Susan Athey, and Stefan Wager. Local linear forests. arxiv. ArXiv preprint, <https://arxiv.org/abs/1807.11408>, 2018.
- P. Richard Hahn, Jared S. Murray, and Carlos M. Carvalho. Bayesian regression tree models for causal inference: Regularization, confounding, and heterogeneous effects. *Bayesian Analysis*, 2020. doi: 10.1214/19-BA1195.
- Behram Hansotia and Brad Rukstales. Incremental value modeling. *Journal of Interactive Marketing*, 16(3):35, 2002. doi: 10.1002/dir.10035.
- Jennifer L Hill. Bayesian nonparametric modeling for causal inference. *Journal of Computational and Graphical Statistics*, 20(1):217–240, 2011.
- Daniel G Horvitz and Donovan J Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47(260):663–685, 1952.

- Daniel Jacob. Cross-fitting and averaging for machine learning estimation of heterogeneous treatment effects. ArXiv preprint, <https://arxiv.org/abs/2007.02852>, 2020.
- Edward H Kennedy. Optimal doubly robust estimation of heterogeneous causal effects. ArXiv preprint, <https://arxiv.org/abs/2004.14497>, 2020.
- Michael C Knaus, Michael Lechner, and Anthony Strittmatter. Machine Learning Estimation of Heterogeneous Causal Effects: Empirical Monte Carlo Evidence. *The Econometrics Journal*, 06 2020. doi: 10.1093/ectj/utaa014.
- Sören R Künzel, Jasjeet S Sekhon, Peter J Bickel, and Bin Yu. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the National Academy of Sciences*, 116(10):4156–4165, 2019. doi: 10.1073/pnas.1804597116.
- Jared K Lunceford and Marie Davidian. Stratification and weighting via the propensity score in estimation of causal treatment effects: A comparative study. *Statistics in Medicine*, 23(19):2937–2960, 2004. doi: 10.1002/sim.1903.
- Daniel F McCaffrey, Greg Ridgeway, and Andrew R Morral. Propensity score estimation with boosted regression for evaluating causal effects in observational studies. *Psychological methods*, 9(4):403, 2004.
- Sendhil Mullainathan and Jann Spiess. Machine learning: an applied econometric approach. *Journal of Economic Perspectives*, 31(2):87–106, 2017.
- Whitney K Newey and James R Robins. Cross-fitting and fast remainder rates for semiparametric estimation. ArXiv preprint, <https://arxiv.org/abs/1801.09138>, 2018.
- X Nie and S Wager. Quasi-oracle estimation of heterogeneous treatment effects. *Biometrika*, 09 2020. ISSN 0006-3444. doi: 10.1093/biomet/asaa076.

- Eric C Polley, Sherri Rose, and Mark J Van der Laan. Super learning. In *Targeted learning*, pages 43–66. Springer, 2011. doi: 10.1007/978-1-4419-9782-1_3.
- James M Poterba and Steven F Venti. 401 (k) plans and tax-deferred saving. In *Studies in the Economics of Aging*, pages 105–142. University of Chicago Press, 1994.
- Scott Powers, Junyang Qian, Kenneth Jung, Alejandro Schuler, Nigam H Shah, Trevor Hastie, and Robert Tibshirani. Some methods for heterogeneous treatment effect estimation in high dimensions. *Statistics in Medicine*, 37(11):1767–1787, 2018. doi: 10.1002/sim.7623.
- James M Robins, Peng Zhang, Rajeev Ayyagari, Roger Logan, Eric Tchetgen Tchetgen, Lingling Li, Thomas Lumley, and Aad van der Vaart. New statistical approaches to semiparametric regression with application to air pollution research. *Research report (Health Effects Institute)*, (175):3, 2013.
- Peter M Robinson. Root-n-consistent semiparametric regression. *Econometrica: Journal of the Econometric Society*, pages 931–954, 1988. doi: 10.2307/1912705.
- Andrea Rotnitzky, James Robins, and Lucia Babino. On the multiply robust estimation of the mean of the g-functional. ArXiv preprint, <https://arxiv.org/abs/1705.08582>, 2017.
- Donald B Rubin. Randomization analysis of experimental data: The fisher randomization test comment. *Journal of the American Statistical Association*, 75(371):591–593, 1980. doi: 10.2307/2287653.
- Anton Schick. On asymptotically efficient estimation in semiparametric models. *Annals of Statistics*, 14(3):1139–1151, 1986. doi: doi:10.1214/aos/1176350055.
- Amit Sharma, Emre Kiciman, et al. DoWhy: A Python package for causal inference. <https://github.com/microsoft/dowhy>, 2019.
- Mark J van der Laan. Targeted maximum likelihood based causal inference: Part i. *The international journal of biostatistics*, 6(2), 2010.

Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018. doi: 10.1080/01621459.2017.1319839.

T Wendling, K Jung, A Callahan, A Schuler, NH Shah, and B Gallego. Comparing methods for estimation of heterogeneous treatment effects using observational data from health care databases. *Statistics in medicine*, 37(23):3309–3324, 2018. doi: 10.1002/sim.7820.

Richard Wyss, Alan R Ellis, M Alan Brookhart, Cynthia J Girman, Michele Jonsson Funk, Robert LoCasale, and Til Stürmer. The role of prediction modeling in propensity score estimation: an evaluation of logistic regression, bcart, and the covariate-balancing propensity score. *American journal of epidemiology*, 180(6):645–655, 2014. doi: 10.1093/aje/kwu181.

A Additional Proofs

Proof of the doubly-robustness property for the DR-learner. If either the propensity score or the conditional mean is correctly specified, the doubly-robust estimator is an unbiased estimator. Let us look at $\hat{m}u_1(x)$, the same procedure holds analog or $\hat{\mu}_0$. That is, $\hat{\mu}_1(x)$ estimates the following:

$$\begin{aligned}
&= \mathbb{E} \left[\mu_1(x) + \frac{D \{Y - \mu_1(x)\}}{e(x)} \right] \\
&= \mathbb{E} \left[\frac{e(x)}{e(x)} \mu_1(x) + \frac{D \{Y - \mu_1(x)\}}{e(x)} \right] \\
&= \mathbb{E} \left[\frac{DY}{e(x)} - \frac{(D - e(x))\mu_1(x)}{e(x)} \right], \text{ add } \frac{e(x)}{e(x)}Y, -\frac{e(x)}{e(x)}Y \\
&= \mathbb{E} \left[\frac{(D - e(x))Y}{e(x)} - \frac{(D - e(x))\mu_1(x)}{e(x)} + \frac{e(x)}{e(x)}Y \right] \\
\hat{\mu}_1(x) &= \mathbb{E} \left[\frac{(D - e(x))(Y^1 - \mu_1(x))}{e(x)} + Y^1 \right] \\
\hat{\mu}_1(x) &= \mathbb{E}[Y^1] + \mathbb{E} \left[\frac{(D - e(x))(Y^1 - \mu_1(x))}{e(x)} \right] \tag{1.17}
\end{aligned}$$

B Additional Plots

Figure B.1 describes the two-step sample splitting. The first splitting is necessary for nuisance parameter estimation. The second split is done to get out-of-bag estimates of the CATE. In this approach no cross-fitting is applied. Figure B.2 now uses the samples that include the pseudo-outcomes and split it into different folds. Each fold is used to train a regression model. Prediction is done in the out-of-bag fold. Through the different folds we get as many predictions as folds used for training. These estimates are then averaged which applies the cross-fitting procedure.

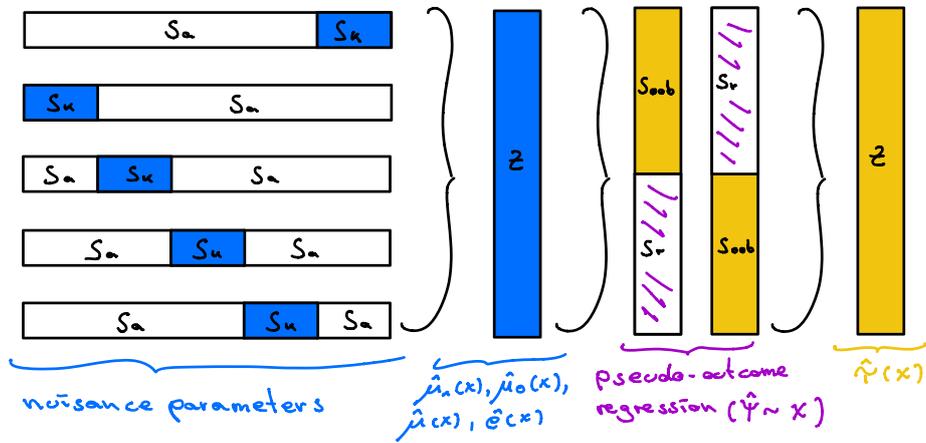


Figure B.1: Two-step sample splitting: 50:50, no cross-fitting.

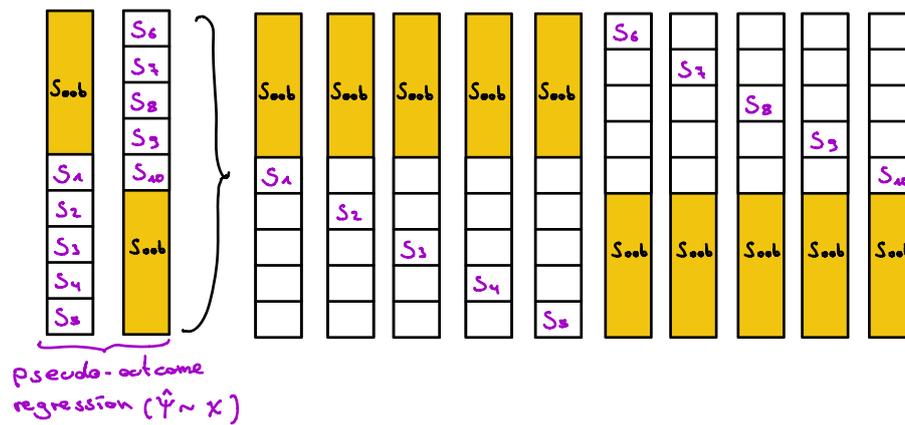


Figure B.2: Detailed cross-fitting procedure for 5 folds.

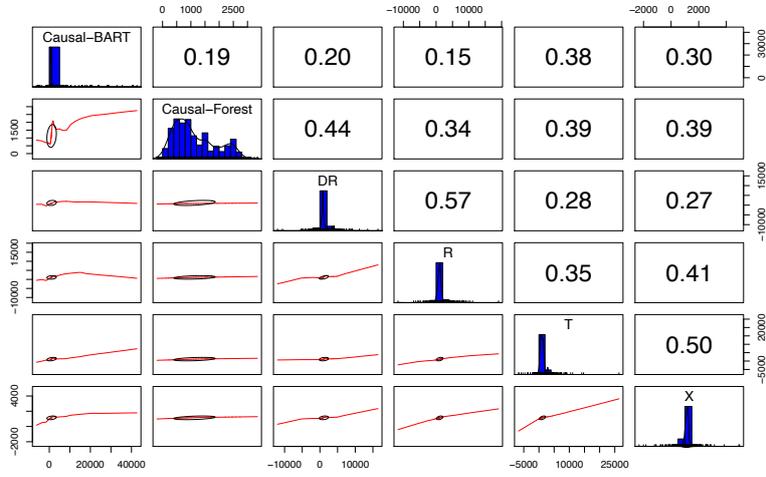


Figure B.3: Correlation of CATE between different methods from the microcredit data. \mathcal{O}_{C1}

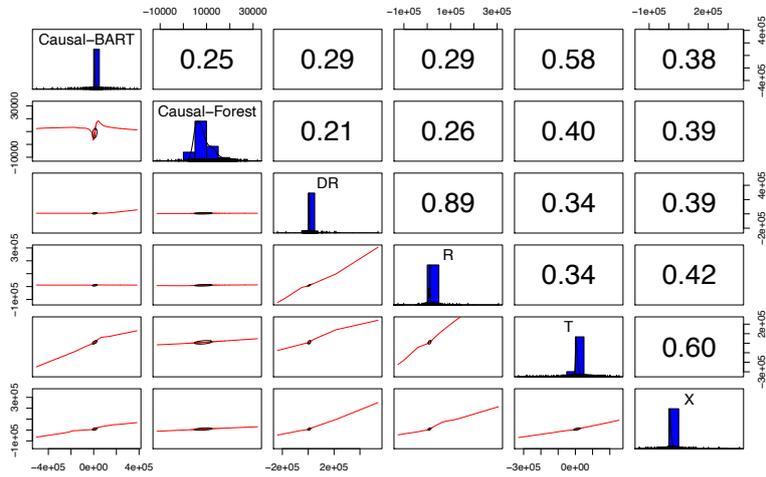


Figure B.4: Correlation of CATE between different methods from the 401k data. \mathcal{O}_{C2}

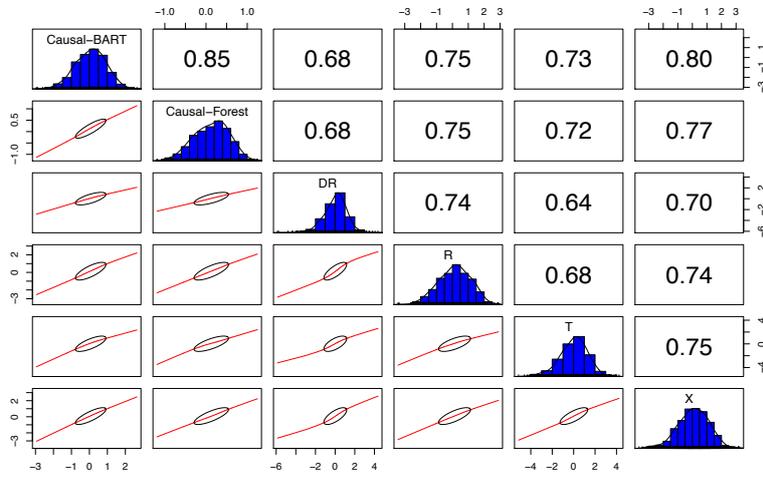


Figure B.5: Correlation of CATE between different methods for simulation setting 1. 

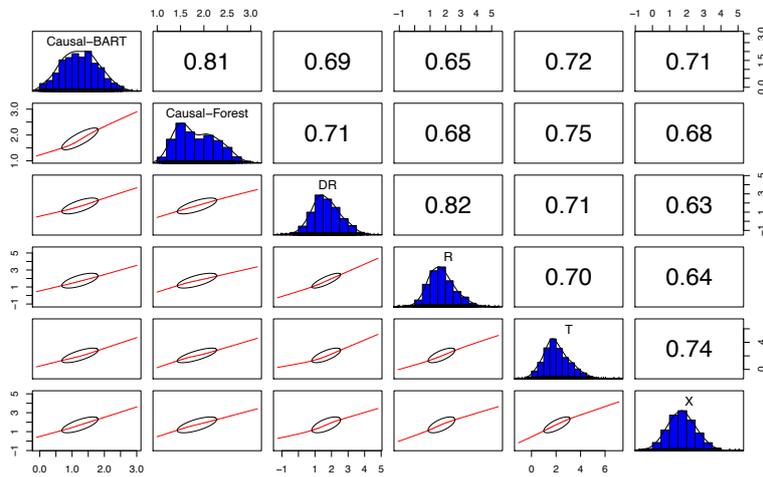


Figure B.6: Correlation of CATE between different methods for simulation setting 2. 

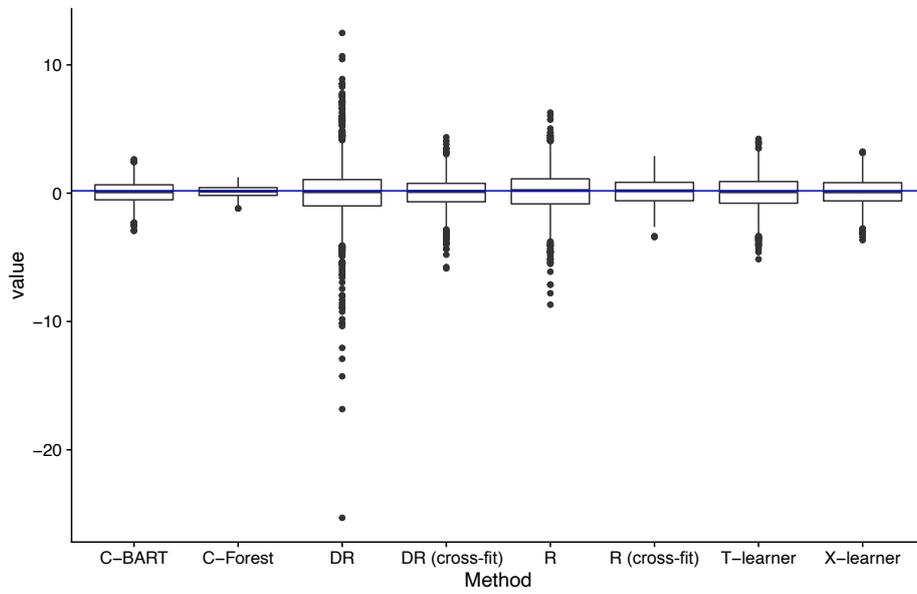


Figure B.7: Setting 1: Boxplots of different methods. Blue line shows true ATE. \mathcal{Q}_{B3}

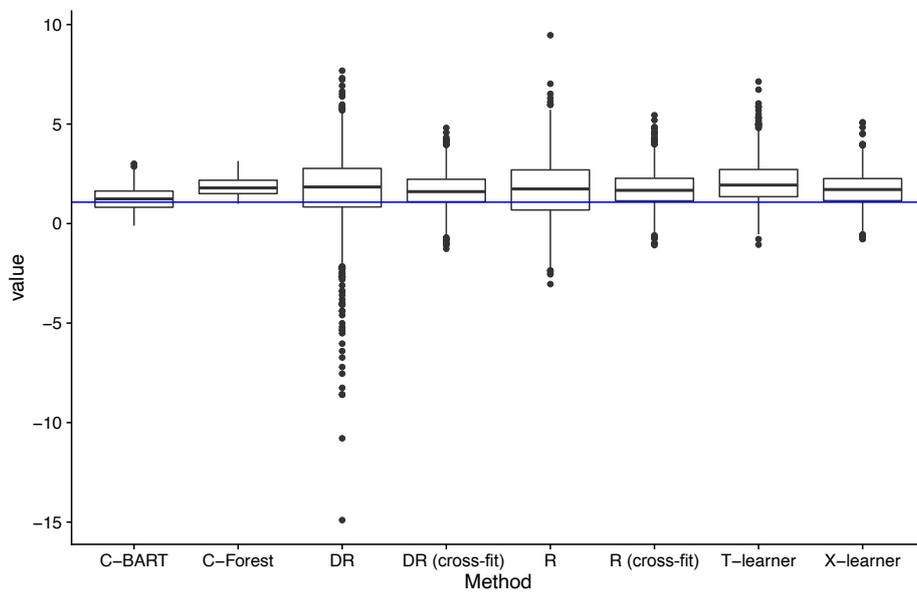


Figure B.8: Setting 2: Boxplots of different methods. Blue line shows true ATE. \mathcal{Q}_{B4}

C Tables

C.1 Classification results for the microcredit example

Table C.1: CLAN results for different methods.

	DR-learner			R-learner			T-learner		
	Most Affected	Least Affected	Difference	Most Affected	Least Affected	Difference	Most Affected	Least Affected	Difference
Head age	19.19 (17.81,20.58)	46.92 (45.53,48.30)	-27.73 (-29.68,-25.77)	29.99 (28.50,31.49)	42.52 (40.98,44.06)	-12.53 (-14.67,-10.38)	23.43 (21.93,24.94)	38.17 (36.67,39.68)	-14.74 (-16.87,-12.61)
	-	-	[0.000]	-	-	[0.000]	-	-	[0.000]
Non-agricultural	0.118 (0.097,0.139)	0.186 (0.165,0.207)	-0.068 (-0.098,-0.038)	0.178 (0.158,0.198)	0.097 (0.076,0.118)	0.081 (0.052,0.110)	0.149 (0.128,0.169)	0.126 (0.106,0.146)	0.023 (-0.006,0.051)
self-employed	-	-	[0.000]	-	-	[0.000]	-	-	[0.244]
Borrowed from	0.138 (0.113,0.162)	0.338 (0.314,0.363)	-0.201 (-0.235,-0.166)	0.181 (0.155,0.206)	0.320 (0.294,0.346)	-0.139 (-0.175,-0.103)	0.116 (0.093,0.139)	0.273 (0.250,0.296)	-0.157 (-0.189,-0.125)
any source	-	-	[0.000]	-	-	[0.350]	-	-	[0.000]
	X-learner			Causal BART			Causal Forest		
	Most Affected	Least Affected	Difference	Most Affected	Least Affected	Difference	Most Affected	Least Affected	Difference
Head age	19.66 (18.19,21.14)	38.94 (37.47,40.42)	-19.28 (-21.37,-17.19)	14.85 (13.68,16.02)	50.05 (48.88,51.22)	-35.21 (-36.86,-33.55)	10.25 (9.271,11.22)	46.80 (45.82,47.77)	-36.55 (-37.93,-35.17)
	-	-	[0.000]	-	-	[0.000]	-	-	[0.000]
Non-agricultural	0.138 (0.116,0.160)	0.181 (0.159,0.202)	-0.043 (-0.073,-0.012)	0.150 (-0.073,-0.012)	0.099 (0.080,0.119)	0.051 (0.130,0.169)	0.073 (0.118,0.154)	0.136 (0.055,0.091)	-0.064 (0.038,0.089)
self-employed	-	-	[0.013]	-	-	[0.000]	-	-	[0.000]
Borrowed from	0.091 (0.067,0.115)	0.417 (0.394,0.441)	-0.327 (-0.360,-0.293)	0.091 (0.068,0.113)	0.300 (0.278,0.323)	-0.210 (-0.242,-0.178)	0.050 (0.028,0.072)	0.388 (0.366,0.411)	-0.338 (-0.370,-0.307)
any source	-	-	[0.000]	-	-	[0.000]	-	-	[0.000]

Notes: Averages are taken from the mean of the CATE over 500 bootstrap iterations.

Additional Pseudocode

Algorithm 7: S-learner

Input: $Z_i = \{Y_i, D_i, X_i\}_{i \in N}$

```
1 Split sample  $Z$  into  $K$  random subsets
2 for  $k$  in  $\{1, \dots, K\}$  do
3   assign Sample  $S_a = Z \cup S_k$  and  $S_k$ 
4   regress  $Y_i = \hat{\mu}(X_i, D_i) + \hat{U}_i$ , with  $i \in S_a$ 
5     estimate  $\hat{Y}_i^0 = \hat{\mu}(X_i, D = 0)$ , with  $i \in S_k$ 
6     estimate  $\hat{Y}_i^1 = \hat{\mu}(X_i, D = 1)$ , with  $i \in S_k$ 
7   create  $\hat{\tau}_k(X_i) = \hat{\mu}(X_i, D = 1) - \hat{\mu}(X_i, D = 0)$ 
8 end
9 combine  $\hat{\tau}(X_i) = \{\hat{\tau}_1, \hat{\tau}_k, \dots, \hat{\tau}_K\}$ 
```

Algorithm 9 refers to the inverse probability weighting (IPW) estimator based on [Horvitz and Thompson \(1952\)](#). While [Künzel et al. \(2019\)](#) refer to this estimator as the F-learner, it is also known as the (simplest) transformed outcome estimator. This is because $\hat{\psi}_{IPW}$ is an unbiased estimate of the ATE. The only nuisance function that is needed to create this outcome is the propensity score. We then map the covariates onto the transformed outcome (or pseudo-outcome). The reason for this mapping is again to smooth the function since the IPW estimator can suffer from high variance if the propensity score estimates are near zero or one. Below we show that the IPW estimator is an unbiased estimator for the ATE.

Algorithm 8: Bootstrap Confidence Interval

Input: $Z_i = \{Y_i, D_i, X_i\}_{i \in N}$

```
1  $p$ : evaluation point (out-of-sample)
2  $S_0 = \{i : D_i = 0\}$ 
3  $S_1 = \{i : D_i = 1\}$ 
4  $n_0 = \#S_0$ 
5  $n_1 = \#S_1$ 
6 for  $b$  in  $\{1, \dots, B\}$  do
7    $S_b^* = c(\text{sample}(S_0, S_1))$ 
8    $y_b^* = y[S_b^*]$ 
9    $d_b^* = d[S_b^*]$ 
10   $x_b^* = x[S_b^*]$ 
11   $\hat{\tau}_b^*(p) = \text{learner}(y_b^*, d_b^*, x_b^*)(p)$ 
12 end
13  $\hat{\tau}(p) = \text{learner}(y, d, x)(p)$ 
14  $\hat{\sigma} = \text{sd}\left(\left\{\hat{\tau}_b^*(p)\right\}_{b=1}^B\right)$ 
15 return  $(\hat{\tau}(p) - q_{\alpha/2}\hat{\sigma}, \hat{\tau}(p) + q_{1-\alpha/2}\hat{\sigma})$ 
```

$$\begin{aligned} \mathbb{E}[\psi_{IPW}|X_i = x] &= \mathbb{E}\left[Y\left\{\frac{D}{e(x)} - \frac{1-D}{1-e(X_i)}\right\}|X_i = x\right] \\ &= \mathbb{E}\left[\left\{D\frac{Y}{e(x)} - (1-D)\frac{Y}{1-e(x)}\right\}|X_i = x\right] \\ &= \mathbb{P}(D = 1|X_i = x)\frac{1}{e(x)}\mathbb{E}[Y|D = 1, X_i = x] \\ &\quad - \mathbb{P}(D = 0|X_i = x)\frac{1}{1-e(x)}\mathbb{E}[Y|D = 0, X_i = x] \\ &= \mathbb{E}[Y|D = 1, X_i = x] - \mathbb{E}[Y|D = 0, X_i = x] \\ &= \mu_1(x) - \mu_0(x) = \tau(x) \end{aligned}$$

Algorithm 9: IPW-learner

Input : $Z_i = \{Y_i, D_i, X_i\}_{i \in N}$

```
1 Split sample  $Z$  into  $K$  random subsets
2 for  $k$  in  $\{1, \dots, K\}$  do
3   assign Sample  $S_a = Z \cup S_k$  and  $S_k$ 
4   regress  $D_i = \hat{e}(X_i) + \hat{V}_i$ , with  $i \in S_a$ 
5     estimate  $\hat{D}_i = \hat{e}(X_i)$ , with  $i \in S_k$ 
6   create  $\hat{\psi}_{IPW} = Y \left\{ \frac{D}{\hat{e}(x)} - \frac{1-D}{1-\hat{e}(X_i)} \right\}$ 
7   store  $\hat{\psi}_{IPW,k}$  for  $i \in S_k$ 
8 end
9 Cross-fitting:
10 for  $oob$  in (1:2) do
11   if  $oob = 1$ :  $S_{oob} = Z_i$  with  $i \in \{1, \dots, N/2\}$  and  $S_{train} = Z_i \cup S_{oob}$ 
12   if  $oob = 2$ :  $S_{train} = Z_i$  with  $i \in \{1, \dots, N/2\}$  and  $S_{oob} = Z_i \cup S_{in}$ 
13   for  $l$  in 1:5 do
14     split  $S_{train}$  in  $\{S_1, S_2, \dots, S_5\}$ 
15     regress  $\hat{\psi}_i = \hat{t}_{IPW}(X_i) + W_i$ , for  $i \in S_l$ 
16     estimate  $\tilde{\tau}_l(X_i) = \hat{t}_{IPW}(X_i)$ , with  $i \in S_{oob}$ 
17   end
18   average  $\hat{\tau}_{oob}(X_i) = \mathbb{E}[\tilde{\tau}(X_i)]$ 
19 end
20 row bind  $\hat{\tau}(X_i) = \{\hat{\tau}_1, \hat{\tau}_2\}$ 
```
