Model averaging and double machine learning*

Achim Ahrens† Christian B. Hansen‡ Mark E. Schaffer§
Thomas Wiemann‡

February 15, 2023

Abstract

This paper discusses pairing double/debiased machine learning (DDML) with *stacking*, a weighted average of several candidate machine learners. The combination of the two methods leads to improved performance and addresses some of the recent concerns raised about the use of causal machine learning in economic applications. Additionally, we introduce DDML with *short-stacking*, which exploits the cross-fitting step of DDML to substantially reduce the computational burden of stacking. The central motivation for pairing DDML and stacking approaches is that, in practice, it is rarely obvious which machine learner performs best for a specific application, and we show that stacking can successfully address this problem. Based on a diverse set of applications and calibrated simulation studies, we gauge the finite sample performance of DDML combined with stacking approaches on real data in various commonly encountered settings. We show that DDML with stacking is more robust than other approaches when the underlying data-generating process is unknown and develop practically-relevant recommendations for researchers.

Keywords: high-dimensional models, model averaging, machine learning, causal inference, Neyman-orthogonal
JEL: C21, C26, C52, C55, J01, J08

*Many thanks to Elliot Ash, Daniel Björkegren, Gabriel Okasa for helpful discussions and comments. This is an early draft. Please do not cite or circulate without permission.

†ETH Zürich, Switzerland, *Email:* achim.ahrens@gess.ethz.ch
‡University of Chicago, United States
§Heriot-Watt University, Edinburgh, United Kingdom and IZA Institute of Labor Economics

1 Introduction

Supervised machine learning promises superior prediction performance compared to conventional parametric approaches and more flexibility in approximating unknown functional forms. Although the precise distinction to traditional estimators is elusive, key features shared by many supervised machine learners are that they are data-adaptive in the sense that model selection and validation are partially inbuilt, that they are nonparametric since they do not impose that outcomes follow parametrized conditional distributions and that they rely on some form of regularization, e.g., shrinkage. While originally designed for prediction and classification tasks, a flourishing literature proposes strategies for leveraging machine learning for causal inference. One domain of causal machine learning uses supervised machine learning to control for observed confounders or the approximation of optimal instrumental variables (IVs). Within this field, the post-double-selection (PDS) lasso of Belloni, Chernozhukov, and Hansen (2014) selects control variables in a partially linear model through two auxiliary lasso regressions. Belloni et al. (2012) and Athey, Tibshirani, and Wager (2019) use lasso and random forests for IV regressions.[1]

More recently, however, concerns have been raised about the use of methods leveraging machine learning for causal inference. Angrist and Frandsen (2022) illustrate through a series of calibrated simulation studies that the selection of IVs via the lasso can perform poorly relative to more conventional many IV estimators (e.g., LIML). In experiments using artificial pure-noise IVs, they also show that IVs residualized by random forests may yield spurious results. Wüthrich and Zhu (2021) find that PDS lasso is prone to underselect in small samples, leading to a potentially substantial bias, and conclude that OLS with appropriate standard errors compares favorably also in high-dimensional settings. Giannone, Lenza, and Primiceri (2021) argue that the approximate sparsity assumption, which the lasso fundamentally relies on, is frequently not plausible in economic data sets. In an application to the evaluation of active labor market programs, Goller et al. (2020) find that random forests are not suitable for the estimation of propensity scores.

Against this background, we revisit the use of machine learning for causal inference. We highlight the benefits of pairing Double/Debiased Machine Learning (DDML; Chernozhukov et al., 2018a) with stacking approaches and introduce a novel, computationally more efficient variation of DDML with stacking, which we refer to as *short-stacking*. DDML relies on cross-fitting, a sample-splitting approach, which offers a path for utilizing a general class of machine learners for the estimation of causal parameters.[2] Stacking is a form of model averaging due originally to Wolpert (1992) and Breiman (1996). Stacking

[1] Another influential strand of the literature exploits machine learning for the estimation of conditional average treatment effects; see Wager and Athey (2018), Chernozhukov et al. (2018b), and Künzel et al. (2019). For a review and Monte Carlo evidence, see Knaus, Lechner, and Strittmatter (2021).

[2] Sample splitting approaches, such as cross-fitting, are commonly applied to leverage supervised machine learners for causal inference; e.g., Wager and Athey (2018) for the estimation of CATE using causal forests, or Athey and Wager (2021) in the context of policy learning.

combines several *base* or *candidate learners* into a final estimator or "super learner" and has been shown to perform at least as well as the best-performing candidate learner (Laan, Dudoit, and Vaart, 2006; Laan, Polley, and Hubbard, 2007).[3] Short-stacking is a variant of DDML with stacking that leverages the cross-fitting step of DDML to substantially reduce the computational burden of stacking.

The central motivation for pairing DDML and stacking approaches is that, in practice, it is rarely obvious which machine learner performs best for a specific application. In the context of causal inference, Laan, Rose, et al. (2011) advocate for stacking for Targeted Maximum Likelihood Estimations. Yet, despite its strong theoretical foundation and practical relevance, stacking is rarely used in causal inference in economics and social sciences. Based on a diverse set of applications and calibrated simulation studies, we gauge the finite sample performance of DDML combined with stacking and short-stacking on real data in various commonly encountered settings, including IV regressions, and show that DDML with stacking addresses some of the aforementioned concerns. We also find that short-stacking performs better in small samples.

We begin with a review of the motivation for causal machine learning. A reason often brought forward is that machine learning allows one to deal with high-dimensional data sets where the number of variables is "large" relative to the sample size. High-dimensional models appear in many contexts in economics: for example, when a rich set of job seeker characteristics is observed on an online job platform (Hangartner, Kopp, and Siegenthaler, 2021), when IVs constitute complex weather patterns (Gilchrist and Sands, 2016), or when confounds are represented as text data (Roberts, Stewart, and Nielsen, 2020b). However, the advantages of causal machine learning are not confined to settings where the number of observed confounds or IVs is large. For instance, if identification relies on an unconfoundedness assumption, traditional parametric approaches require knowledge of the nuisance functions. Even if only a few controls are observed, the empirical researcher faces the difficult challenge of correctly specifying which controls to include and how to transform them (e.g., interaction terms, dichotomization). If the model is incorrectly specified, parametric estimators generally do not give consistent estimates, even if unconfoundedness holds true. Data-adaptive nonparametric approaches offer, in comparison, more flexibility in approximating unknown data structures. This flexibility translates into enhanced robustness in estimating causal effects, as we show in multiple simulation studies.

Another motivation is related to the *weak causality* property introduced in Blandhol et al. (2022). A weakly causal estimand can be expressed as a weighted average of subgroup treatment effects where the weights are positive. In the context of IV estimation where instrument validity relies on observable confounders, Blandhol et al. (2022) em-

[3]See also Hansen and Racine (2012) for discussion of jackknife (leave-one-out) stacking. Hastie, Tibshirani, and Friedman (2009) includes a textbook treatment of stacking.

phasize that, in the absence of strong functional form assumptions, TSLS is generally not weakly causal. The implication is that parametric conditioning is generally insufficient for guaranteeing that the TSLS estimand can be interpreted as the local average treatment effect (LATE). As we discuss, the point is more general and does not only apply to IV models. We highlight that nonparametric conditioning through DDML allows sustaining a weakly causal interpretation of structural parameters also in partially linear models.

We then provide simulation evidence demonstrating that stacking safeguards against ill-chosen and poorly tuned machine learners provided a generous and diverse set of base learners is chosen. Stacking assigns large weights to linear learners when the data-generating process is linear, and large weights to non-linear learners when the data-generating process is non-linear. This behavior is reflected in a relatively low bias of causal estimates, independent of the underlying data-generating process. Stacking, thus, proves practically relevant in the ubiquitous scenario where there is uncertainty about the appropriate regularization assumption for the application of interest. We also consider the small-sample behavior of DDML with stacking. Following the simulation design of Wüthrich and Zhu (2021), we show that stacking performs comparably to linear approaches for even moderate sample sizes when the data-generating process is indeed linear. We argue that the poor sample performance of PDS lasso is partially driven by the choice of covariate transformations and illustrate how stacking can accommodate a richer set of specifications, including competing parametric models. However, while increasing the number of folds helps, DDML with stacking should be used with caution for very small samples. Motivated by these finite sample concerns, we suggest an alternative approach which we refer to as *short-stacking*. Short-stacking leverages the cross-fitted predicted values, is computationally cheaper, and performs better in small samples.

Lastly, we examine DDML for IV regressions. We test whether the concerns raised by Angrist and Frandsen (2022) concerning IV regressions relying on random forests for conditioning on covariates also apply to DDML with stacking. Using the application of Angrist and Evans (1998), we find no evidence for spurious or imprecise IV estimations when we use DDML paired with stacking and when constraints fundamental for identification are directly enforced. The specific example illustrates the risks associated with hidden identifications assumptions that are implicitly enforced by linear estimators, but that need to be explicitly enforced by flexible nonparametric learners. Furthermore, we show that the finite sample performance of the fully flexible DDML-IV estimator for approximating optimal IVs can be enhanced by enforcing the law of iterated expectations.

Taken together, our results support the use of machine learning for causal inference. We find that DDML paired with stacking approaches enhances robustness against misspecification compared to traditional parametric estimators as well as compared to causal machine learning estimators exclusively relying on a single regularization assumption.

We proceed as follows. The next section provides a general introduction to DDML. In

Section 3, we motivate the use of DDML paired with stacking, introduce short-stacking and provide simulation evidence in support of stacking approaches. In Section 4, we turn to IV estimations. We discuss two alternative DDML estimators for IV regression and assess their finite sample performance. Section 5 illustrates DDML using two applications. In the first application, we estimate the citation penalty for female authors while conditioning on the full article text. The second application estimates the effect of a cash transfer program in the presence of heterogeneous treatment effects. Section 6 concludes.

2 Double Debiased Machine Learning

On a general level, DDML considers the all-causes model

$$Y = f_0(D, X, U), \tag{1}$$

where f_0 is a structural function, Y is the outcome variable, D is the treatment or policy variable of interest, which may be binary or continuous, X are observed covariates, and U collects unobserved determinants. Throughout this article, we will discuss special cases of the all-causes model with different underlying assumptions.

DDML is concerned with estimating a parameter of interest θ_0 while flexibly accommodating high-dimensional nuisance functions η_0. The parameter of interest θ_0 will typically be a summary of the causal effect of D on the outcome. DDML approaches the estimation of causal parameters in two stages: the estimation of possibly nonparametric nuisance functions, and then the estimation of the parameter itself. Depending on the underlying model assumptions, these nuisance functions may govern the treatment assignment mechanism or, if identification relies on IVs, describe how endogenous treatments relate to excluded IVs. In leading cases, the nuisance functions represent conditional expectation functions that lend themselves to the application of supervised machine learning. DDML relies on cross-fitting, a sample-splitting approach that ensures that the same observation is not used for both the first- and second-stage estimation. Cross-fitting allows utilizing a general class of supervised machine learners for the estimation of causal effects, only relying on a relatively mild convergence rate requirement. Its general applicability is the main motivation for our focus on DDML.

DDML fundamentally relies on Neyman-orthogonality conditions of the form

$$\partial_\eta E\left[\psi\left(W; \theta_0, \eta_0\right)\right]\left[\eta - \eta_0\right] = 0, \qquad W = (Y, D, X). \tag{2}$$

where ψ is a score function. Neyman-orthogonality refers to the property that the moment condition is insensitive to local perturbations around the true nuisance parameter η_0. The implication is that we can utilize noisy estimates of η_0 to estimate θ_0.

One leading special case of the all-causes model is the Partially Linear Model, which assumes that treatment, unobservables, and observables are additively separable while allowing the control variables to enter the model through an unknown function, i.e.,

$$f_0(D, X, U) = \theta_0 D + g_0(X) + U. \tag{3}$$

In addition, we make an unconfoundedness assumption, stating that treatment assignment is as good as random conditional on observed covariates.

Assumption 1 (PL)**.** *We assume* (3) *and conditional orthogonality* $E[Cov(U, D|X)] = 0$.

Under Assumption 1, the Neyman-orthogonal moment condition in (2) holds for the Robinson (1988)-style score

$$\psi(W; \theta_0, \eta_0) = \Big(Y - \ell_0(X) - \theta_0\big(D - m_0(X)\big)\Big)\big(D - m_0(X)\big),$$

where $\eta_0 \equiv (\ell_0, m_0)$ are the nuisance functions, which constitute the condition expectations $m_0(X) \equiv E[D|X]$ and $\ell_0(X) \equiv E[Y|X]$. Rewriting yields

$$\theta_0 = \frac{E\left[\big(Y - \ell_0(X)\big)\big(D - m_0(X)\big)\right]}{E\left[(D - m_0(X))^2\right]}. \tag{4}$$

Equation (4) highlights that the estimand θ_0 depends on the CEFs $m_0(X)$ and $\ell_0(X)$. Estimating these CEFs is in itself a predictive problem, for which we can draw from the machine learning toolbox. In the classical setting of least squares with a known small-dimensional set of covariates X, the estimation of (4) involves partialling out the controls from Y and D, which by Frisch-Waugh-Lovell is equivalent to regressing Y against D and X. However, DDML allows for estimating the high-dimensional nuisance functions flexibly using a general class of supervised machine learners.

We note here that the population parameter θ_0 in (4) has the weakly causal property of Blandhol et al. (2022). Suppose that D is integer-valued, taking on the values $\{0, 1, \ldots, \overline{d}\}$, then θ_0 can be represented as a weighted average over the conditional average treatment effects (CATEs) of increasing the dose by one unit:

$$\theta_0 = E\left[\sum_{t=1}^{\overline{d}} \mathrm{CATE}_{t,t-1}(X)\omega(t, X)\right].$$

where $\mathrm{CATE}_{t,t-1}(X) \equiv E\left[f(t, X, U) - f(t-1, X, U)|X\right]$. Angrist and Krueger (1999) show that the weights $\omega(t, X)$ are weakly positive and integrate to one. Similarly, for continuous D with differentiable conditional expectation function, it holds that

$$\theta_0 = E\left[\int_{\mathrm{supp}\, D} \left(\frac{\partial}{\partial t} E[Y|D = t, X]\right)\omega(t, X)dt\right],$$

where again the weights are weakly positive and integrate to one. However, erroneously imposing that the CEFs are linear or employing ill-suited CEF estimators may result in negative weights. To avoid issues associated with negative weights, Blandhol et al. (2022) stress the need for flexible covariate specifications. In this sense, DDML guarantees the weakly causal interpretation of causal estimands by approximating the high-dimensional nuisance functions flexibly.

In principle, we could plug CEF estimates derived from supervised machine learners into (4). This, however, would generally induce an *own-observation bias*, or over-fitting bias, unless we rely on appropriate regularization assumptions and confine ourselves to a relatively narrow set of machine learners imposing, e.g., *Donsker* conditions. The own observation bias stems from reusing the same observations for both estimating nuisance terms and structural parameters. For example, in the partially linear model, the error from learning $\ell_0(X)$ may be correlated with $D - m_0(X)$. Under the additional assumption of approximate sparsity, we could employ PDS lasso with the plugin penalty of Belloni et al. (2012), which tightly controls the overfitting bias. To allow for a general class of data-adaptive non-parametric estimators and machine learners, while relaxing the linearity or weakening the sparsity assumptions, Chernozhukov et al. (2018a) propose cross-fitting—a form of sample splitting on swapped samples.

We introduce cross-fitting with an example. Suppose we observe data from a random sample $\{(Y_i, D_i, X_i)\}_{i \in I}$ with $I = \{1, \ldots, n\}$. The aim is to estimate the nuisance functions η_0 where $\eta_0 \equiv (\ell_0, m_0)$ in the partially linear model. In the simplest case, sample splitting involves randomly splitting the sample into two typically evenly-sized folds, denoted as I_1 and $I_2 = I \setminus I_1$. The auxiliary sample I_1 is used to estimate the nuisance terms η_0 using supervised machine learners and obtain the out-of-sample estimates $\hat{\eta}_{I_1}(X_i)$ for $i \in I_2$, where the I_1 sub-script indicates that the learner was trained only using the sample I_1. The main sample I_2 is then used to estimate the structural parameter. While this simple sample splitting approach ensures that the bias in learning η_0 does not spill over to the estimation of θ_0, it is costly as only half of the data is used for the final estimation. The trick of cross-fitting is to also fit the machine learner to the main sample I_2 to obtain $\hat{\eta}_{I_2}(X_i)$ for $i \in I_1$, which yields a out-of-sample estimates of the conditional expectations for each observation in the sample.[4]

The cross-fitting algorithm generalizes straightforwardly to more than two folds. We split the sample in K folds of approximately equal size, denoted as $I_1, \ldots, I_K$, and $k(i)$ is the fold that observation i is assigned to. The complement, $T_k \equiv I \setminus I_k$, constitutes the training sample. The cross-fitted predicted values are then defined as $\tilde{\eta}_i := \hat{\eta}_{T_{k(i)}}(X_i)$ for

[4]The approach described above corresponds to algorithm 'DML2' described in (Chernozhukov et al., 2018a). An alternative approach, referred to as 'DML1', is to fit the final estimation by fold to obtain one parameter estimate per fold $\hat{\alpha}_k$. The overall DDML estimate is then calculated as $\hat{\alpha}_{\text{DML1}} = 1/K \sum_k \hat{\alpha}_k$. Chernozhukov et al. (2018a, Remark 3.1) recommend 'DML2' over 'DML1'. We thus focus exclusively on 'DML2' in this exposition.

$i = 1, \ldots, n$. Thus, the cross-fitted predicted value for observation i is constructed using all folds except fold $k_{(i)}$, the fold that observation i falls into.

The DDML estimate is calculated by plugging the cross-fitted values into the score function, and setting

$$\frac{1}{n} \sum_{i \in I} \psi \left(W; \hat{\theta}, \tilde{\eta}_i \right) = 0$$

and solving for $\hat{\theta}_0$.

In the partially linear model, where $\tilde{\eta}_i \equiv (\tilde{\ell}_i, \tilde{m}_i)$, this yields

$$\hat{\theta}_n^{\text{PLM}} = \frac{\frac{1}{n} \sum_{i=1}^{n} \left(Y_i - \tilde{\ell}_i \right) \left(D_i - \tilde{m}_i \right)}{\frac{1}{n} \sum_{i=i}^{n} \left(D_i - \tilde{m}_i \right)^2}. \tag{5}$$

Hence, in the final estimation stage of the DDML estimation for the Partially Linear Model, we regress the residualized outcome, $(Y_i - \tilde{\ell}_i)$ against the residualized treatment $(D_i - \tilde{m}_i)$ where we use cross-fitted predicted values for residualization. Classical, heteroskedasticity-consistent or cluster-robust standard errors can be applied.

Since the DDML point estimate relies on the randomly drawn cross-fitting split, we follow the recommendation of Chernozhukov et al., 2018a by repeating the cross-fitting procedure using different random folds and aggregating the estimates to reduce reliance on a specific fold split. For example, suppose $\hat{\theta}_n^{(r)}$ is the DDML estimate from the rth cross-fit repetition and $\hat{s}_n^{(r)}$ is the associated standard error estimate with $r = 1, \ldots, R$. One option is to calculate the median point estimate and associated standard error are defined as

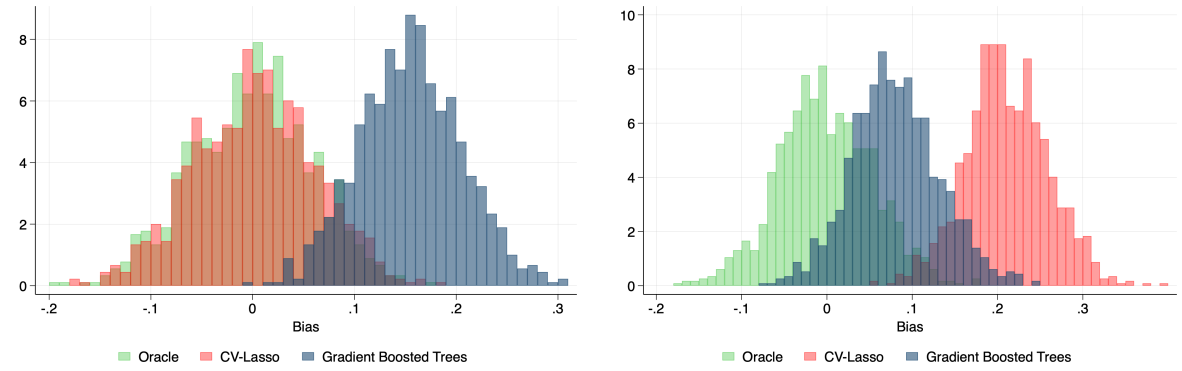
$$\breve{\hat{\theta}}_n = \text{median}\left(\left(\hat{\theta}_n^{(r)} \right)_{r=1}^{R} \right) \quad \text{and} \quad \breve{\hat{s}}_n = \sqrt{\text{median}\left(\left((\hat{s}_n^{(r)})^2 + (\hat{\theta}_n^{(r)} - \breve{\hat{\theta}}_n)^2 \right)_{r=1}^{R} \right)}.$$

While mean aggregation is also legitimate, we favor median aggregation since it is less sensitive to outlier estimates.

3 The choice of machine learner

By virtue of cross-fitting, DDML estimators are well-behaved for a general class of CEF estimators achieving a relatively mild convergence rate requirement. DDML estimators are asymptotically normal and achieve the root-N convergence rate, even if estimators of the nuisance terms converge at a slower rate.[5] Naturally, we can employ fast-converging

[5]The exact convergence rate requirement for CEF estimators depends on the problem. Chernozhukov et al. (2018a) name the crude rate requirement of $o(n^{-1/4})$, but provide examples where the rate requirement is considerably weaker. Recent contributions show that these requirements are satisfied by specific instances of machine learners; see, e.g., results for lasso (Bickel, Ritov, and Tsybakov, 2009; Belloni et al., 2012), random forests (Wager and Walther, 2016; Wager and Athey, 2018; Athey, Tibshirani, and Wa-



(a) Linear DGP (b) Non-linear DGP

Notes: Figures (a) and (b) compare the bias of the oracle estimator (which knows the true data-generating process), cross-validated lasso and gradient-boosted trees under two alternative data-generating processes. Specifically, we generate 1'000 samples of size $n = 1000$ using the partially linear model $Y_i = \theta_0 D_i + g(X_i) + \varepsilon_i$, $D_i = g(X_i) + u_i$ where the nuisance function is either $g(X_i) = \sum_j 0.9^j X_{ij}$ (linear) or $g(X_i) = \mathbb{1}\{X_{i1} > 0.3\}\mathbb{1}\{X_{i2} > 0\}\mathbb{1}\{X_{i3} > -1\}$ (non-linear DGP). Gradient boosting uses 1000 trees, a learning rate of 0.01 and early stopping with 20% validation sample. See Ahrens et al. (2023, Section 4.2) for details.

Figure 1: Bias in estimating θ_0 in the partially linear model under linear and non-linear data-generating process

parametric estimators, such as ordinary least squares, which assume knowledge of the nuisance functions. However, the strength of DDML comes into play when we use flexible adaptive nonparametric estimators that are able to accommodate certain non-linear and unknown data structures.

Not every machine learner is suitable for a given task. To the practitioner, it is rarely obvious which CEF estimator will perform best, especially when domain knowledge is scarce. As we illustrate in the simulation exercise in Figure 1, the bias from relying on a poorly chosen or misspecified machine learner can be severe. In the left figure, we use a linear, approximately sparse data-generating process (DGP). Cross-validated lasso perfectly matches the distribution of the infeasible oracle estimator, which assumes knowledge of the unobserved nuisance functions, whereas gradient-boosted trees fail to approximate the nuisance term sufficiently well, leading to a substantial bias. In the right figure, we use non-linear DGP involving interaction effects. Here, gradient-boosted trees perform much better compared to the cross-validated lasso, but still exhibit a significant bias.

Given the apparent risk of misspecification and misleading inference, the choice of machine learner needs to be thoroughly validated by comparing the performance of a diverse set of candidate learners. A common approach is to cross-validate candidate learners and select the learner with the lowest cross-validated loss. This approach is sometimes referred to as *cross-validation selector* or *single-best selector*.

ger, 2019), neural networks (Schmidt-Hieber, 2020; Farrell, Liang, and Misra, 2021), and boosting (Luo, Spindler, and Kück, 2022). The exact asymptotic properties of many machine learners, however, remains an active research area.

But instead of relying on a single learning algorithm, we can enhance the flexibility and robustness of CEF estimators using stacking. Stacking allows combining several individually trained base learners into a meta learner. Stacking is particularly well-suited to DDML, given the main objective is to estimate the causal parameter of interest rather than the CEFs per se. We also view stacking as a framework that encourages practitioners to report not only the tuning parameters of the final learner, but the specification of all candidate learners considered in the tuning process, thereby facilitating transparency and reproducibility.

In Section 3.1, we outline stacking for DDML estimation. We also propose an alternative, computationally cheaper approach to stacking for DDML estimation, which we refer to as *short-stacking*. Section 3.2 demonstrates the advantages of stacking using a calibrated simulation. Section 3.3 examines the behavior of stacking and short-stacking in small samples.

3.1 Stacking and short-stacking

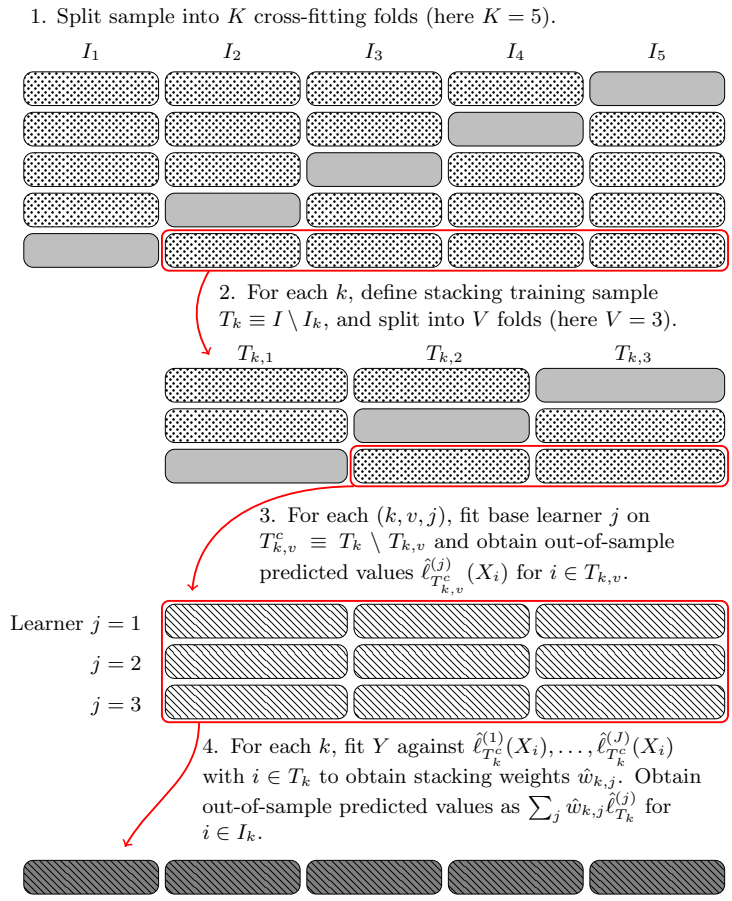
Throughout this section, we assume the researcher observes a sample $\{(Y_i, X_i)\}_{i \in I}$ which we split into K cross-fitting folds, denoted $I_1, \ldots, I_K$. Furthermore, we assume that we have a set of J base or candidate learners at our disposal. The set of learners could include distinct algorithms—e.g., tree-based methods and regularized regression—but also the same algorithm with varying (hyper-)tuning parameters or different sets of predictors. We use the estimation of $\ell_0(X) = E[Y|X]$ as an example, but note that the estimation of other CEFs proceeds in the same way. Figures 2 and 3 illustrate the stacking and short-stacking algorithm.

Stacking. In the context of DDML, stacking involves two layers of re-sampling. The *cross-fitting layer* divides the sample into K cross-fitting folds. In each cross-fitting step $k \in \{1, \ldots, K\}$, the stacking learner is trained on the training sample $T_k \equiv I \setminus I_k$. Fitting the stacking learner, in turn, requires to sub-divide the training sample T_k further into V cross-validation folds, which constitutes the *cross-validation layer*. We denote the cross-validation folds by $T_{k,1}, \ldots, T_{k,V}$. Each candidate learner $j \in \{1, \ldots, J\}$ is cross-validated on these folds, which yields cross-validated predicted values.

The final learner fits the outcome Y_i against the cross-validated predicted values of each candidate learner. The final learner could be a parametric or non-parametric estimator, although the most common choice is constrained least squares where we impose that the coefficients are non-negative and sum to one. Specifically, for each k, we use the objective

$$\min_{w_{k,1},\ldots,w_{k,J}} \sum_{i \in T_k} \left(Y_i - \sum_{j=1}^{J} w_{k,j} \hat{\ell}^{(j)}_{T^c_{k,v(i)}}(X_i) \right)^2 \qquad \text{s.t. } w_{k,j} \geq 0,\ \sum_{j=1}^{J} |w_{k,j}| = 1.$$

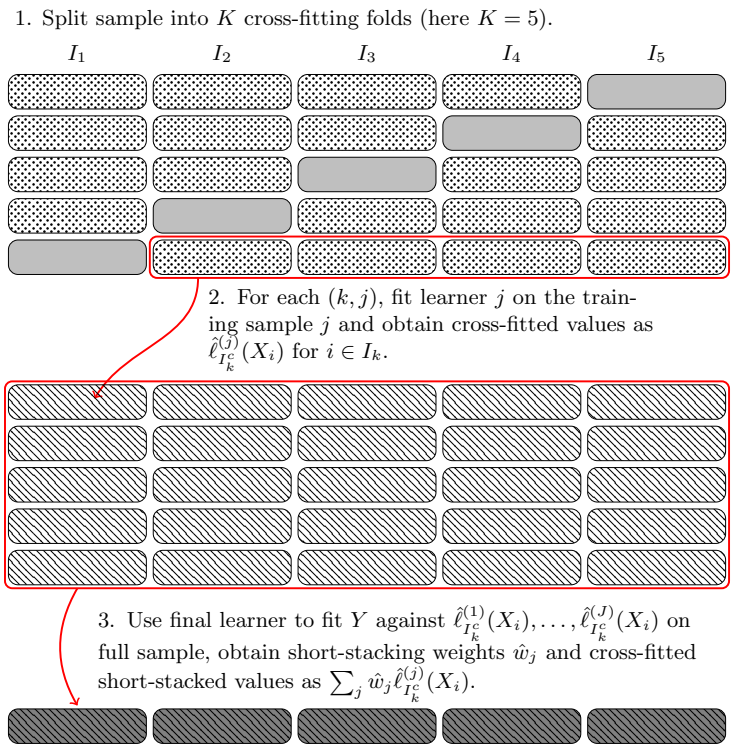
Figure 2: Cross-fitting and stacking. Illustration using the estimation of $\ell_0 = E[Y|X]$ as an example.



Notes: The diagram illustrates cross-fitting with $K = 5$ cross-fitting folds combined with stacking using $V = 3$ cross-validation folds and three base learners. The illustration uses the estimation of $\ell_0 = E[Y|X]$ as an example.

Here, $\hat{\ell}^{(j)}_{T^c_{k,v(i)}}(X_i)$ denotes the out-of-sample predicted value for observation i, which is calculated from training candidate learner j on $T^c_{k,v(i)} \equiv T_k \setminus T_{k,v(i)}$, i.e., all step-$k$ cross-validation folds but fold $(k, v(i))$ which is the fold that observation i falls into. We call $\hat{w}_{k,j}$ the stacking weights. The stacking predictions are obtained as $\sum_j \hat{w}_{k,j}\hat{\ell}^{(j)}_{T_k}(X_i)$ where each learner j is fit on T_k. Constrained least squares facilitates the interpretation of stacking as a weighted average of base learners (Hastie, Tibshirani, and Friedman, 2009). Due to this constraint, CLS tends to set some stacking weights to exactly zero. The constraint also implies a form of regularization that reduces the risk of over-fitting, which is especially relevant given that candidate learners are likely highly correlated. If we instead impose the constraint that $w_{k,j} \in \{0, 1\}$ and $\sum_j w_{k,j} = 1$, implying that only one stacking weight can be one, the approach is equivalent to the cross-validation selector, which selects the candidate learner with lowest cross-validated loss. Other final learners are possible. Laan, Dudoit, and Vaart (2006, Theorem 1) show for a parametric final learner where the number of candidate learners grows at most at polynomial rate that the stacking learner performs asymptotically at least as well as the best candidate learner.[6]

Figure 3: Cross-fitting and *short*-stacking. Illustration using the estimation of $\ell_0 = E[Y|X]$ as an example.



Notes: The diagram illustrates cross-fitting combined with short-stacking with $K = 5$ cross-fitting folds. The illustration uses the estimation of $\ell_0 = E[Y|X]$ as an example.

[6]The *scikit-learn* (Buitinck et al., 2013) routines `StackingRegressor` and `StackingClassifier` implement stacking for Python. In Stata, stacking regression and classification are available via `pystacked`, a Stata front-end for these Python routines (Ahrens, Hansen, and Schaffer, 2022).

Short-stacking. A drawback of DDML with stacking is its computational complexity. Considering the estimation of a single base learner as the unit of complexity, DDML with stacking heuristically has a computational cost proportional to $K \times V \times J$. For example, when considering DDML with $K = 5$ cross-fitting folds and $J = 10$ base learners that are combined based on $V = 5$ fold cross-validation, more than 250 base learners need to be individually estimated. Although DDML with stacking is "embarrassingly parallel" and can thus be expected to decrease in computational time nearly linearly in the number of available computing processes, the increased complexity limits its application to moderately complex applications. Another potential concern (which we investigate in Section 3.3) is that DDML with stacking might not perform well in small samples, given that base learners are effectively trained on approximately $\frac{(K-1)(V-1)}{KV}\%$ of the full sample (see Figure 2). These two concerns motivate *short-stacking*.

In the context of DDML where we rely on cross-fitting, we can take a short-cut to stacking: Instead of fitting the final learner on the cross-validated fitted values in each step k of the cross-fitting process, we can directly train the final learner on the cross-fitted values using the full sample, see Figure 3. The objective function becomes:

$$\min_{w_1,\ldots,w_J} \sum_{i=1}^{n} \left(Y_i - \sum_{j=1}^{J} w_j \hat{\ell}^{(j)}_{I^c_{k(i)}}(X_i) \right)^2 \qquad \text{s.t. } w_j \geq 0,\ \sum_j |w_j| = 1$$

where w_j are the short-stacking weights. Cross-fitting thus serves a double purpose: First, it avoids the own-observation bias by avoiding overlap between the samples used for estimating high-dimensional nuisance functions and the samples used for estimating structural parameters. Second, it yields out-of-sample predicted values, which we can leverage for training the final stacking learner. As a consequence, the computational cost of DDML with short stacking is heuristically only proportional to $K \times J$ in units of estimated base learners. In the example of the previous paragraph, short-stacking thus requires about 200 fewer estimated base learners.

3.2 The benefits of pairing DDML and stacking

To test the performance of stacking and short-stacking in a realistic setting, we consider the analysis of 401(k) eligibility and total financial wealth in Poterba, Venti, and Wise (1995) as the basis for an empirically calibrated Monte Carlo simulation. The application has recently been revisited by Belloni et al. (2017), Chernozhukov et al. (2018a), and Wüthrich and Zhu (2021) to approximate high-dimensional confounding factors using machine learning. We focus on the partially linear model where the outcome is measured as net financial wealth, and the treatment variable is a dummy for eligibility to the 401(k) pension scheme. The set of controls includes age, income, education in years, family size, as well as indicators for two-earner status, home ownership, and participation in two

alternative pension schemes.

In the calibration step of the simulation, we fit two generative models to the $n = 9{,}915$ households from the 1991 SIPP. Specifically, we use linear regression and gradient-boosted trees to extract and magnify the linear or non-linear structures in the empirical conditional distributions. The generative step then simulates two samples from the fully linear and the partially linear model, respectively, where we set the effect of 401(k) eligibility on total financial wealth to $\theta_0 = 6000$. Finally, in the estimation stage, we fit OLS, PDS lasso and different DDML estimators on bootstrapped samples to estimate the effect of 401(k) eligibility, and compare their bias and coverage rates. The simulation set-up thus allows for assessing the performance of estimators across favorable or unfavorable data-generating processes.

Since we will make use of similar simulation designs repeatedly in this article, we outline the steps used for constructing the two generative models in detail below:

1. Let $\{(y_i, d_i, x_i)\}_{i=1,\dots,n}$ denote the observed sample, where i is a household in the 1991 SIPP. y_i, d_i, and x_i denote the net wealth, an indicator for 401(k) eligibility, and the vector of control variables, respectively.
2. Using the full sample, calculate the slope coefficient $\hat{\theta}_{OLS}$ from linear regression of d_i against d_i, and x_i in the original data.[7] Construct the partial residuals $y_i^{(r)} = y_i - \hat{\theta}_{OLS} d_i$, $\forall i$.
3. Fit a supervised learning estimator (either linear regression or gradient boosting) to predict $y_i^{(r)}$ with the controls x_i. Denote the fitted estimator by $\tilde{g}$. Similarly, fit a supervised learning estimator to predict d_i with x_i and denote the fitted estimator by $\tilde{h}$.
4. Sample from the empirical distribution of x_i by bootstrapping n_b observations from the original data. Denote the bootstrapped sample by $\mathcal{D}_b$.
5. Draw $\nu_i \overset{iid}{\sim} \mathcal{N}(0, \kappa_1)$ and $\varepsilon_i \overset{iid}{\sim} \mathcal{N}(0, \kappa_2)$, where κ_1 and κ_2 are simulation hyperparameters. Define

$$\begin{aligned} \tilde{d}_i^{(b)} &= \mathbb{1}\{\tilde{h}(x_i) + \nu_i \geq 0.5\} \\ \tilde{y}_i^{(b)} &= \theta_0 \tilde{d}_i^{(b)} + \tilde{g}(x_i) + \varepsilon_i \qquad \forall i \in \mathcal{D}_b \end{aligned}$$

   where we set $\theta_0 = 6000$ to roughly resemble the magnitude of the regression coefficient of 401(k) eligibility in the full data. We set the hyper-parameter κ_1 and κ_2 to approximately match variance of 401(k) eligibility and log-wealth in the data.[8]

The simulation exercise repeats steps 4.-5. to generate simulated samples of size n_b.

[7] In the full sample, this results in $\hat{\theta}_{OLS} \approx 5896$.

[8] The values of the simulation hyperparameters (κ_1, κ_2) differ slightly depending on the supervised learning estimator used to fit the reduced form equations in the data. We take $\kappa_1 = 0.35$ in both scenarios but take $\kappa_2 = 55500$ when using linear regression and $\kappa_2 = 54000$ when using gradient boosting. Differences arise because gradient boosting reduces residual variance in the true data.

For each sample, we calculate estimates of the effect of 401(k) eligibility on wealth, which allows us to compare the performance of linear regression, PDS lasso, as well as various DDML estimators with different supervised learners for estimating the CEFs $E[Y|X]$ and $E[D|X]$. We consider the following learners: CV lasso and CV ridge with interactions and second-order polynomials, CV lasso and CV ridge with 10th-order polynomials and no interactions, two versions of random forests, two versions of gradient-boosted trees and feed-forward neural nets with three hidden layers of size five (see Table 1 notes for details). In addition to calculating DDML estimators based on a single estimator, we consider two stacking and two short-stacking approaches, each using either CLS or the single-best learner (i.e., the learner that minimized the cross-validated MSPE) as the final learner.

Table 1 presents the bias and coverage rates of a 95% confidence interval associated with estimates of the effect of 401(k) eligibility on total financial assets for all considered estimators and bootstrap sample sizes of $n_b = \{9915, 99150\}$. The top and bottom panels correspond to results based on data simulated from the linear and non-linear generative models, respectively. The individual learners combined in each ensemble are given in Table 2 along with their respective stacking weights averaged over folds and over bootstrap iterations. Table A.1 gives the corresponding mean-squared prediction errors.

Given the construction of the generative models, we would expect that OLS performs best in the fully linear setting and that DDML with gradient boosting performs best in the nonlinear setting where the nuisance function is generated by gradient boosting. The simulation results confirm this intuition, showing that the two procedures achieve among the lowest bias and median absolute bias (MAB) in the data-generating processes that are based on them. Researchers are rarely certain of the functional structure in economic applications, however, so that it is more interesting to consider their respective performance in the non-favorable setting. In the non-linear data-generating process, linear regression is among the estimators with the worst performance across all three measures. Similarly, gradient boosting-based DDML is non-optimal in the linear data-generating process, outperformed by linear regression and CV lasso in terms of MAB, both of which enforce a linear functional form on the control variables.

The simulation results are consequences of the "no free lunch" theorem in machine learning (Wolpert, 1996). Informally, the theorem states that there exists no estimator that performs best across all empirical settings. Researchers must, therefore, carefully match estimators to their application. However, with limited knowledge about underlying data-generating processes and few functional form restrictions implied by economic theory, the number of plausibly well-suitable estimators is typically large.

Stacking reduces the burden of choice the researcher faces when selecting appropriate estimators by allowing for the simultaneous consideration of multiple estimators. The four approaches we consider adaptively weight the included learners, which increases the

Table 1: Bias and Coverage Rates in the Linear and Non-Linear DGP

	n_b = 9,915			n_b = 99,150		
Panel (A): Linear DGP	Bias	MAB	Rate	Bias	MAB	Rate
Full sample:						
OLS	25.2	827.7	0.95	−1.8	263.1	0.95
PDS-Lasso	26.2	822.8	0.96	0.7	264.1	0.95
DDML methods:						
Base learners						
OLS	27.2	807.0	0.95	−2.1	263.7	0.95
Lasso with CV (2nd order poly)	26.1	804.7	0.96	−1.0	263.5	0.95
Ridge with CV (2nd order poly)	24.6	814.7	0.95	−1.6	261.5	0.95
Lasso with CV (10th order poly)	−4.7	995.5	0.95	61.1	270.6	0.95
Ridge with CV (10th order poly)	583.3	1269.9	0.94	35.6	266.2	0.95
Random forest (low regularization)	−129.9	1028.0	0.90	−14.8	337.0	0.87
Random forest (high regularization)	30.5	852.7	0.95	−14.2	277.9	0.94
Gradient boosting (low regularization)	−21.1	841.0	0.95	−18.7	255.1	0.95
Gradient boosting (high regularization)	85.4	836.2	0.96	71.9	270.8	0.95
Neural net	−3756.6	5537.2	0.16	−2269.4	3144.1	0.16
Meta learners						
Stacking: CLS	10.7	826.9	0.96	−2.2	263.9	0.95
Stacking: Single best	16.7	830.0	0.95	−4.1	262.3	0.95
Short-stacking	21.5	801.9	0.96	−2.5	264.5	0.95
Single best	25.9	800.9	0.95	−3.7	262.9	0.95
Panel (B): Non-Linear DGP	Bias	MAB	Rate	Bias	MAB	Rate
Full sample:						
OLS	−2733.3	2730.6	0.55	−2641.0	2630.4	0.
PDS-Lasso	−2744.8	2739.8	0.55	−2640.1	2631.5	0.
DDML methods:						
Base learners						
OLS	−2759.5	2729.6	0.54	−2643.9	2634.4	0.
Lasso with CV (2nd order poly)	599.7	1014.7	0.92	705.8	700.4	0.63
Ridge with CV (2nd order poly)	662.0	1037.7	0.92	716.5	710.9	0.62
Lasso with CV (10th order poly)	−5866.9	1944.0	0.91	−24.6	275.1	0.94
Ridge with CV (10th order poly)	−3036.6	2180.1	0.89	−12.7	273.3	0.94
Random forest (low regularization)	−197.9	1024.5	0.90	−14.7	332.4	0.88
Random forest (high regularization)	−292.8	941.1	0.93	−0.1	275.0	0.94
Gradient boosting (low regularization)	−73.2	864.9	0.94	26.5	250.8	0.96
Gradient boosting (high regularization)	72.4	868.4	0.94	191.0	296.8	0.93
Neural net	−4584.3	5728.9	0.20	−2968.9	3882.3	0.16
Meta learners						
Stacking: CLS	−309.5	1008.8	0.95	21.3	247.7	0.96
Stacking: Single best	−21.6	969.4	0.94	27.8	247.8	0.95
Short-stacking	58.0	889.0	0.95	33.0	254.3	0.95
Single best	−16.8	859.3	0.95	26.5	250.8	0.96

Notes: The table reports mean bias, median absolute bias (MAB) and coverage rate of a 95% confidence interval for the listed estimators. We consider DDML with the following individual learners: OLS with elementary covariates, CV lasso and CV ridge with second-order polynomials and interactions, CV lasso and CV ridge with 10th-order polynomials but no interactions, random forest with low regularization (8 predictors considered at each leaf split, no limit on the number of observations per node, bootstrap sample size of 70%), highly regularized random forest (5 predictors considered at each leaf split, at least 10 observation per node, bootstrap sample size of 70%), gradient-boosted trees with low regularization (500 trees and a learnings rate of 0.01), gradient-boosted trees with high regularization: 250 trees and a learnings rate of 0.01, feed-forward neural nets with three hidden layers of size five. For reference, we report two estimators using the full sample: OLS and PDS lasso. We report results for four meta learners: Stacking with CLS, short-stacking with CLS, single best overall and single best by fold. Results are based on 1,000 replications.

Table 2: Average stacking weights

	Stacking		Single-Best	
Panel (A): Linear DGP	$E[Y\|X]$	$E[D\|X]$	$E[Y\|X]$	$E[D\|X]$
OLS	0.673	0.507	0.828	0.654
Lasso with CV (2nd order poly)	0.115	0.156	0.146	0.267
Ridge with CV (2nd order poly)	0.065	0.058	0.017	0.020
Lasso with CV (10th order poly)	0.032	0.084	0.003	0.039
Ridge with CV (10th order poly)	0.030	0.044	0.005	0.012
Random forest (low regularization)	0.012	0.012	0.	0.
Random forest (high regularization)	0.018	0.027	0.	0.
Gradient boosting (low regularization)	0.028	0.045	0.	0.005
Gradient boosting (high regularization)	0.025	0.063	0.	0.002
Neural net	0.013	0.003	0.	0.
Panel (B): Non-Linear DGP	$E[Y\|X]$	$E[D\|X]$	$E[Y\|X]$	$E[D\|X]$
OLS	0.033	0.020	0.	0.
Lasso with CV (2nd order poly)	0.038	0.067	0.095	0.157
Ridge with CV (2nd order poly)	0.182	0.238	0.118	0.130
Lasso with CV (10th order poly)	0.060	0.085	0.079	0.046
Ridge with CV (10th order poly)	0.084	0.066	0.024	0.057
Random forest (low regularization)	0.044	0.011	0.	0.
Random forest (high regularization)	0.029	0.079	0.007	0.001
Gradient boosting (low regularization)	0.556	0.222	0.665	0.371
Gradient boosting (high regularization)	0.016	0.209	0.011	0.239
Neural net	0.009	0.002	0.	0.

Notes: The table shows the average stacking weights of each base learner (left) and the relative frequency at which each base learner is selected by the single-best meta learner (right). The bootstrap sample size is 9,915. Results are based on 1,000 replications.

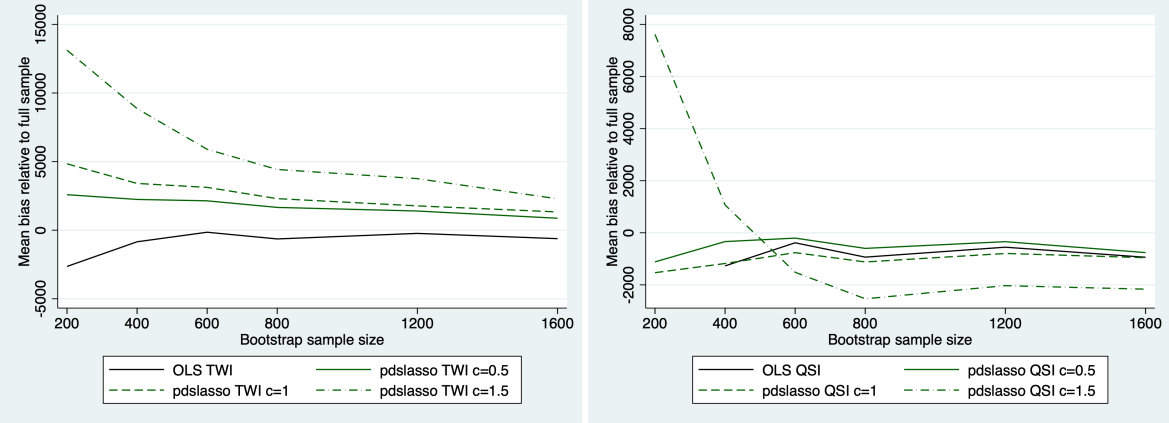
relative impact of well-suited estimators. Our simulation results show that, both in the linear and the non-linear settings, the DDML estimators based on the four meta-learners perform near-optimally. The stacking weights in Table 2 indicate that the procedures successfully select well-suited estimators among the ten included base learners. For example, the stacking approaches applied to the non-linear data-generating process assign the largest weights to the gradient-boosting estimators and the lowest weights to estimators that impose a linear functional form on the control variables. Stacking, therefore, appears to have attractive robustness properties across a variety of data-generating processes.

3.3 DDML and stacking in very small samples

A possible concern for estimators relying on machine learning is that they might not perform well for very small samples given that they are designed for and typically applied to large data sets. Wüthrich and Zhu (2021, henceforth WZ) use two simulations to demonstrate that PDS lasso tends to underselect controls, which may result in a substantial small-sample bias. They also show that the bias heavily depends on the exact lasso penalty chosen (i.e., whether the penalty of Belloni, Chernozhukov, and Hansen (2014) is scaled by 0.5 or 1.5). We revisit the 401(k) simulation in WZ to assess the performance of DDML paired with stacking relative to PDS lasso and OLS.

Following WZ, we consider two sets of controls: two-way interactions (TWI), and

quadratic splines with interactions (QSI) (as in Belloni et al., 2017). The number of predictors are 167 and 272, respectively. WZ runs their simulations on bootstrap samples of the data ($n_b = \{200, 400, 800, 1600\}$) and approximates the bias as the mean difference to the full-sample estimates (n = 9,915).[9] Figure 4 replicates the main results of WZ (Figure 8). Figure (a) and (b) show the bias relative to the full sample estimate for the TWI and QSI specification. It is noteworthy that the speed at which the bootstrapped estimates converge to the full-sample estimate depends on the set of controls for the PDS lasso, but less so for OLS. While PDS lasso with $c = \{0.5, 1\}$ and OLS perform similarly if QSI controls are used, PDS lasso converges much slower to the full-sample estimate with TWI controls.



(a) Bias (TWI) (b) Bias (QSI)

Notes: The figures report the mean bias calculated as the mean difference to the full sample estimates. Full sample estimates reported in Table B.1. Following WZ, we draw 600 bootstrap samples of size $n_b = \{200, 400, 600, 800, 1200, 1600\}$. 'TWI' indicates that the predictors have been expanded by two-way interactions. 'QSI' refers to the quadratic spline & interactions specification of Belloni et al. (2017).

Figure 4: Replication of Figure 8 in Wüthrich and Zhu (2021).

Figure 5 compares the performance of OLS and PDS lasso (with scaling factor of one) to various DDML estimators, including DDML with stacking and short-stacking. We report DDML results using $K = 10$ cross-fitting folds, but also show results for $K = 2$ in the Appendix (see Table B.2 for full results.) We begin with Figure 5a, where we focus on the lasso. One advantage of DDML over PDS lasso is that cross-fitting allows us to employ lasso with cross-validated penalization for a fully data-driven penalization approach. When using TWI controls, we find that DDML with CV lasso performs slightly better than PDS lasso, but both lasso approaches converge only slowly to the full-sample estimates, suggesting that the TWI specification is very sensitive to omitted variable biases. Turning to the QSI controls, DDML with CV lasso outperforms PDS lasso and even OLS. DDML with CV ridge (shown in Figure 5b) converges faster than DDML with CV lasso, for both TWI and QSI controls, suggesting that approaches fully relying on the lasso are sub-optimal in this application. Results for DDML combined with two

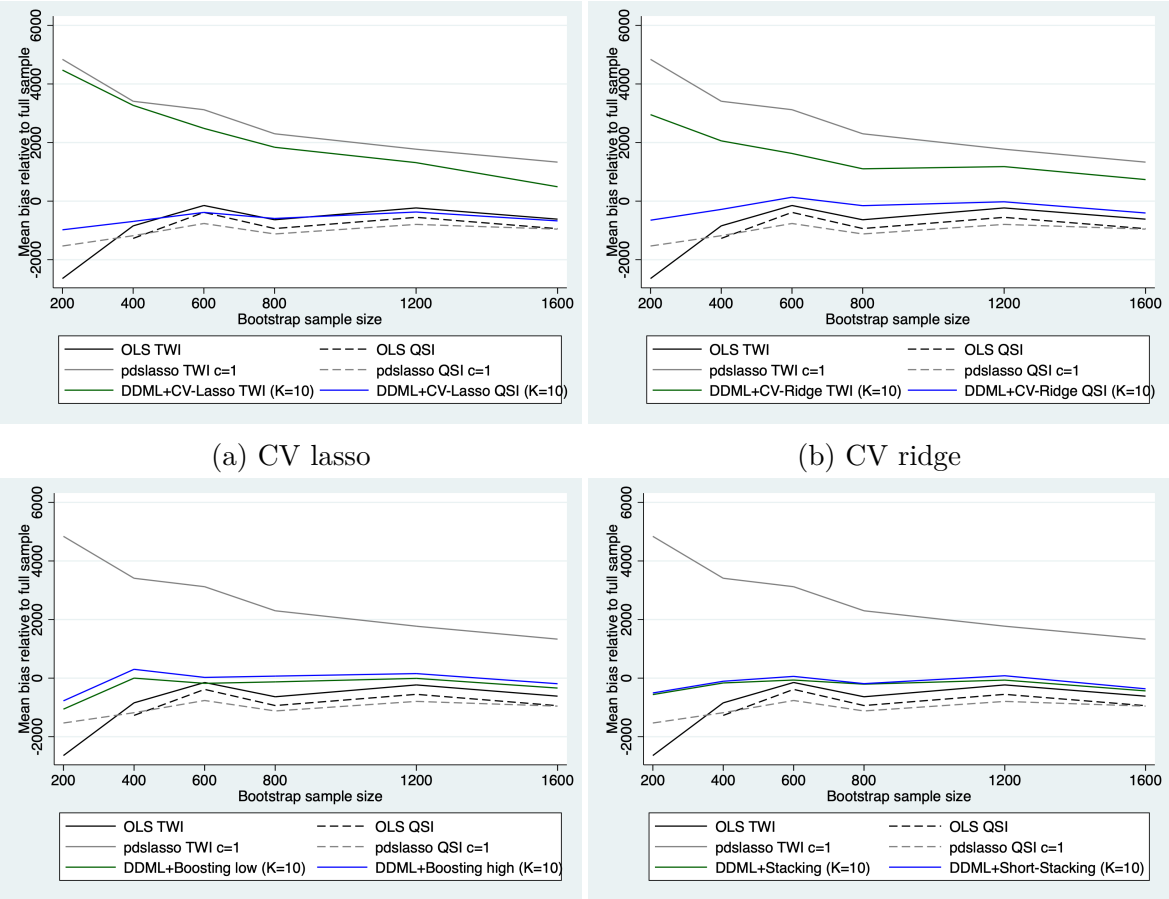
[9]The full-sample estimates are reported in Table B.1.

versions of gradient-boosted trees are in Figure 5c. Both gradient boosting specifications outperform OLS estimators across all sample sizes, even for $n_b = 200$. When comparing random forests and gradient boosting, neither method seems to clearly outperform the other.

The results highlight again that there are risks in relying on poorly chosen specifications. In practice, the researcher does not know whether TWI or QSI controls perform better and whether to use lasso or gradient boosting. We thus turn to Figure 5d, where we show the stacking and short-stacking results. The base learners are CV lasso and CV ridge with TWI and QSI controls, two types of random forests, two types of gradient boosting as well as OLS. Crucially, by considering both TWI and QSI controls and by including OLS, we are agnostic about the optimal set of covariates and about whether a fully parametric approach might be more appropriate in this application. The results are encouraging: DDML paired with stacking or short-stacking outperforms OLS and PDS lasso across all sample sizes. Even at $n_b = 200$, stacking approaches produce estimates which are on average closer to the full-sample estimate. The stacking weights reported in B.3 reveal that stacking approaches also adapt to the sample size. For example, for smaller sample sizes, a larger weight is put on OLS in the estimation of $E[Y|X]$. Furthermore, the stacking weights differ substantially for the estimation of the two nuisance functions, questioning the common approach to rely on the same estimator for all CEFs.

A drawback of measuring the bias as the difference to the full-sample estimate is that we do not gain insights about convergence to the true parameter. We thus revisit the calibrated simulation study from Section 3.2, which allows us to measure the bias as the difference to the true parameter. When the DGP is linear, see Figure 6a, DDML with stacking or short-stacking perform overall similarly to OLS, but only when the number of folds is increased to $K = 10$. With only two folds, DDML with stacking appears, in this application, unsuitable for sample sizes below 800. If the true DGP is non-linear, see Figure 6b, OLS and PDS-Lasso are unable to recover the true effect. DDML with stacking performs much better in comparison but exhibits a substantial bias for any sample size smaller than the full sample. DDML with short-stacking is the only method able to produce a reasonably close approximation of the true parameter, even for small sample sizes.

The results highlight again the risks of relying on inappropriate functional form assumptions. Stacking or short-stacking, when combined with a rich and diverse set of base learners, impose much weaker conditions on the underlying data-generating process. We emphasize that regular stacking might exhibit a large bias for (very) small sample sizes, which can be partially attenuated by increasing the number of folds. Short-stacking is less costly. In the linear DGP it behaves almost identically to OLS. In the non-linear DGP, it is the best-performing method.



(a) CV lasso

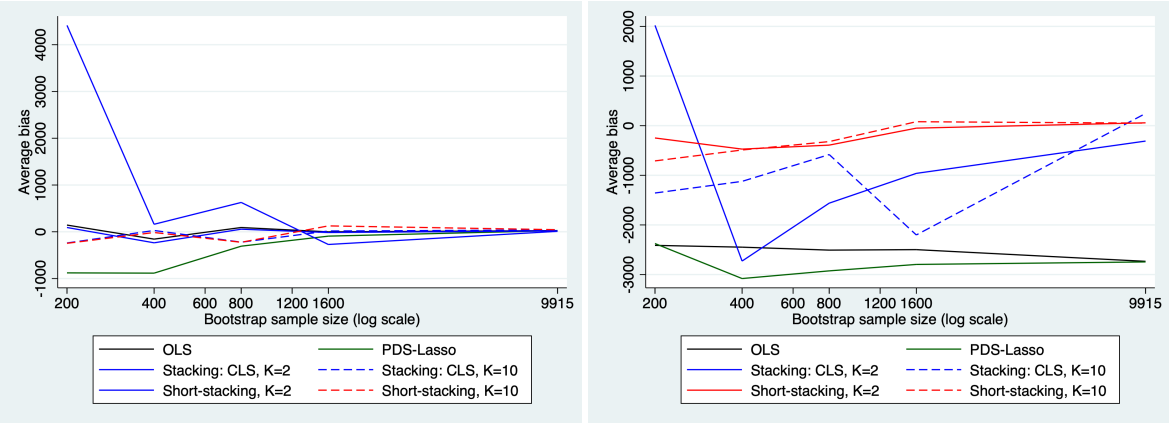
(b) CV ridge

(c) Gradient-boosted trees

(d) Stacking

Notes: The figures report the mean bias calculated as the mean difference to the full sample estimates (see Table B.1). Following WZ, we draw 600 bootstrap samples of size $n_b = \{200, 400, 600, 800, 1200, 1600\}$. 'TWI' indicates that the predictors have been expanded by two-way interactions. 'QSI' refers to the quadratic spline & interactions specification of Belloni et al. (2017). For comparison, we show results for the full-sample estimators OLS and PDS lasso with TWI and QSI controls in all figures. Figures (a) and (b) consider DDML with CV lasso and CV ridge. In Figure (c), we also show results for DDML paired with gradient-boosted tree with either low (500 trees, learnings rate of 0.01) or high (250 trees, learning rate of 0.01) regularization where we only use the elementary controls. Furthermore, in Figure (d), we consider stacking and short-stacking with CLS, relying on the following individual learners: OLS, CV lasso and ridge with TWI or QSI controls, random forest with low (8 predictors considered at each leaf split, no limit on the number of observations per node, bootstrap sample size of 70%) or high regularization (5 splitting predictors, at least 10 observation per node, bootstrap sample size of 70%), gradient-boosted trees with low or high regularization (as defined above). All DDML estimators use $K = 10$. We report full results (including results for $K = 2$) in Table B.2.

Figure 5: Mean bias relative to full sample



(a) Linear DGP (b) Non-linear DGP

Notes: The figure shows results from the calibrated simulation in Table 1, but with smaller bootstrap sample sizes.

Figure 6: Mean bias for very small sample sizes

4 DDML, stacking and IV

This section examines DDML with stacking for IV regressions. Section 4.1 introduces two practically relevant and distinct DDML approaches to IV regression, along with their underlying assumptions. Section 4.2 uses the framework of Angrist and Frandsen (2022) to test DDML with stacking for IV regressions with flexibile controls. Section 4.3 discusses the fully flexible DDML-IV estimator which allows to approximate optimal instruments. We suggests a LIE-compliant algorithm for improving the finite-sample performance of this estimator.

4.1 Partially Linear Model with IVs

We continue focusing on partially linear models of the form

$$f_0(D, X, U) = \theta_0 D + g_0(X) + U,$$

but now leverage IVs Z to identify θ_0. The first of the two approaches takes the identity of IVs Z as given but flexibly conditions on confounders X. A typical application involves one or a few IVs for which the exclusion restrictions only hold conditional on observables. The second approach is more agnostic about how we exploit the IVs for estimation. We approximate optimal IVs (as in Belloni et al. 2012 and Chernozhukov, Hansen, and Spindler 2015) based on the set of observed IVs Z, while again allowing to flexibly control for covariates X. If we consider the lasso as a CEFs estimator, this approach involves selecting both controls and instruments. The benefits of the second approach are especially evident when dealing with many observed IVs or when we expect that the treatment and IVs are connected through an unknown, possibly nonlinear function. Another motivation

for the second approach is that it yields optimal IVs under homoskedasticity.[10]

The two approaches rely on two distinct sets of assumptions:

Assumption 2 (Partially Linear IV Model)**.** *We assume Linear Conditional Orthogonality:* $Cov(U, Z|X) = 0$*, and Linear IV Relevance:* $Cov(D, Z|X) \neq 0$*.*

Assumption 3 (Flexible Partially Linear IV Model)**.** *We assume Conditional IV Mean Independence:* $E[U|Z, X] = 0$*, and Conditional IV Relevance:* $Var(E[D|Z, X]|X) \neq 0$*.*

Both Assumption 2 and 3 form beliefs about the dependence structures of IVs with respect to unobserved errors and with respect to the treatment variable. By confining itself to linear dependence, Assumption 2 uses a weaker exclusion restriction but a stronger relevance requirement in comparison to Assumption 3.

Under Assumption 2, Neyman-orthogonality holds with

$$\psi(W; \theta_0, \eta_0) = \Big(Y - \ell_0(X) - \alpha(D - m_0(X))\Big)\Big(Z - r_0(X)\Big). \tag{6}$$

where $\eta_0 \equiv (\ell_0, m_0, r_0)$ are the nuisance functions, which constitute the condition expectations $m_0(X) \equiv E[D|X]$, $\ell_0(X) \equiv E[Y|X]$ and $r_0(X) \equiv E[Z|X]$. The score in (6) implies that we can obtain the DDML-IV estimate of θ_0 under Assumption 2 using

$$\hat{\theta}_n^{\text{PLIV}} = \frac{\frac{1}{n}\sum_{i=1}^n (Y_i - \tilde{\ell}_i)(Z_i - \tilde{r}_i)}{\frac{1}{n}\sum_{i=1}^n (D_i - \tilde{m}_i)(Z_i - \tilde{r}_i)},$$

where $\tilde{\ell}_i$, $\tilde{m}_i$ and $\tilde{r}_i$ are cross-fitted estimates of the CEFs. The estimation stage amounts to an IV regression of the residualized outcome against the residualized treatment using the residualized IV $(Z_i - \tilde{r}_i)$.

Under Assumption 3, we can also define the score as

$$\psi(W; \theta_0, \eta_0) = \Big(Y - \ell_0(X) - \alpha(D - m_0(X))\Big)\Big(p_0(Z, X) - m_0(X)\Big). \tag{7}$$

with $\eta_0 \equiv (\ell_0, m_0, p_0)$ and $p_0(X) \equiv E[D|Z, X]$. This suggests the alternative DDML-IV estimator

$$\hat{\theta}_n = \frac{\frac{1}{n}\sum_{i=1}^n (Y_i - \tilde{\ell}_i)(\tilde{p}_i - \tilde{m}_i)}{\frac{1}{n}\sum_{i=1}^n (D_i - \tilde{m}_i)(\tilde{p}_i - \tilde{m}_i)}, \tag{8}$$

where $\tilde{p}_i$ are K-fold cross-fitted predicted values from fitting D_i against (Z_i, X_i) and $\tilde{m}_i$ are the cross-fitted predicted values from fitting D_i against X_i. The IV $(\tilde{p}_i - \tilde{m}_i)$ summarizes the information shared by D_i and Z_i conditional on X_i.

A practical concern with the estimator in (8) is that the IV may not be mean independent of X_i, i.e., we cannot guarantee that $\hat{E}[(\tilde{p}_i - \tilde{m}_i)|X_i] \approx 0$. We thus suggest

[10] Under heteroskedasticity or dependent errors, optimality would require estimating conditional variance functions, which we do not consider here.

estimating m_0 by fitting (in-sample) estimates of p_0 against X_i to estimate m_0. We show in simulations in Section 4.3 that the resulting estimator exhibits superior finite sample performance.

4.2 Flexible controls in IV regression

To examine the performance of DDML paired with stacking for IV regressions with flexible controls, we follow Angrist and Frandsen (2022, henceforth AF) in revisiting Angrist and Evans (1998). The original study estimates the effect of giving birth to a third child on the mother's employment status using a binary same-sex indicator of the first two children as the IV. The rationale for the IV choice is that women are more likely to have a third child if the first two children are of the same sex, while the sex assignment of each child is as good as random. Since the mother's employment decision could also be affected by the first and second child's sex individually, as well as by other demographic characteristics, Angrist and Evans (1998) include the first and second child's sex as controls, as well as mother's age, mother's age at first child, indicators for black, hispanic and other race, as well as mother's years of education.

The model can be written as

$$\begin{aligned} Y_i &= \theta\, morekids_i + g_Y(boy1st_i, boy2nd_i, X_i) + \varepsilon_i, \\ morekids_i &= \pi\, samesex_i + g_D(boy1st_i, boy2nd_i, X_i) + \nu_i, \qquad \text{for } i = 1, \ldots, n, \end{aligned}$$

where Y_i is the mother's labor market outcome (either measured as a binary employment indicator or in weeks of employment), $morekids_i$ is a dummy for more than 2 kids, $samesex_i$ is the same-sex IV described above, $boy1st_i$ and $boy2nd_i$ are indicators that equal 1 if the first and second child is male, respectively. The vector X_i collects the additional controls listed above.

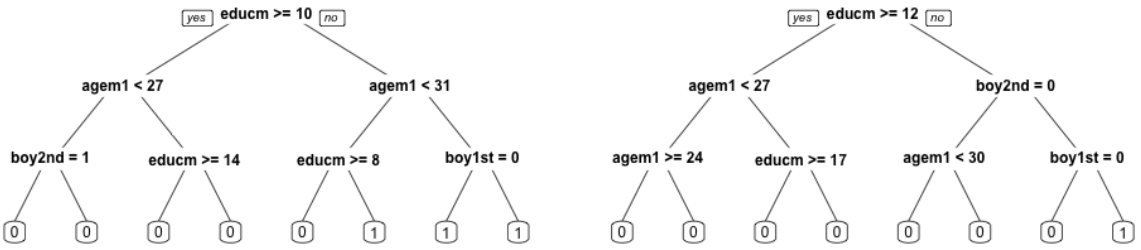
Angrist and Evans (1998) impose additivity by setting

$$\begin{aligned} g_Y(boy1st_{1i}, boy2nd_{2i}, X_i) &= \theta_0 + \theta_1 boy1st_i + \theta_2 boy2nd_i + X_i'\theta_X, \\ g_D(boy1st_{1i}, boy2nd_{2i}, X_i) &= \pi_0 + \pi_1 boy1st_i + \pi_2 boy2nd_i + X_i'\pi_X \end{aligned}$$

and thus allow for direct effects of first and second child's sex on labor outcomes, while ruling out direct effects of $boy1st_i \times boy2nd_i$ on the mother's labor market outcomes. Indeed, direct effects of $boy1st_i \times boy2nd_i$ would imply that the model is not identified since intercept, $boy1st_i$, $boy2nd_i$, $boy1st_i \times boy2nd_i$ and $samesex_i$ are linearly dependent.

AF estimate the effect as

$$\hat{\theta} = \frac{\sum_i \left[Y_i - \hat{g}_Y(boy1st_{1i}, boy2nd_{2i}, X_i)\right]\left[Z_i - \hat{g}_Z(boy1st_{1i}, boy2nd_{2i}, X_i)\right]}{\sum_i \left[D_i - \hat{g}_D(boy1st_{1i}, boy2nd_{2i}, X_i)\right]\left[Z_i - \hat{g}_Z(boy1st_{1i}, boy2nd_{2i}, X_i)\right]}$$



(a) Legal tree (b) Illegal tree

Notes: The two figures show two decision trees grown on bootstrapped samples of size 2000 using the data of Angrist and Evans (1998). The outcome is a binary indicator (labeled as $morekids_i$) set to one if a woman gives birth to more than two kids and zero otherwise. The predictors are mothers' years of education ($educm_i$), mothers' age ($agem_i$), and indicators for whether the first and second child is a boy (denoted $boy1st_i$ and $boy2nd_i$). The left tree is legal since $boy1st_i$ and $boy2nd_i$ do not interact, i.e., are not associated with the same leaf. The right tree violates the additivity assumption. The predicted outcome is shown in rounded boxes at the bottom.

Figure 7: Two exemplary decision trees

where $\hat{g}_Y$, $\hat{g}_Z$ and $\hat{g}_D$ are estimates using various random forests specifications. They consider random forests without sample splitting and with leave-one-out cross-fitting (as suggested in Athey, Tibshirani, and Wager, 2019). AF document that the random-forest-based estimates are imprecise compared to 2SLS. As the authors discuss, random forests do not enforce the additivity constraint that is required for identification.

To understand why, it helps to recall that random forests are composed of many trees, which are individually grown by estimating optimal splitting rules. Each rule involves one predictor. Figure 7 illustrates two decision trees where $morekids_i$ is used as the outcome, and $agem1_i$, $educm_i$, $boy1st_i$ and $boy2nd_i$ as predictors. The rounded box at the bottom of each tree indicates the predicted outcome. The left tree does not violate the additivity constraint, even though both $boy1st_i$ and $boy2nd_i$ feature in the same tree, since no leaf (also referred to as terminal node) is associated with both variables. The right tree does, however, violate the additivity constraint since the decision rule corresponding to the right-most leaf predicts that mothers whose first and second child is female and who have at least 12 years of education will have a third child. The illustration reveals that, since we can easily identify and discard illegal trees, we could, in principle, fit a constrained random forest only using legal trees. While there is, to our knowledge, no readily available random forest implementation that allows for interaction constraints, such interaction constraints are supported by `XGBoost` (Chen and Guestrin, 2016), a widely popular implementation for gradient-boosted trees.

We follow AF but consider various DDML estimators which comply with the additivity constraint. Namely, we use OLS with the base controls, CV lasso and ridge with third-order polynomials which are separately formed for $boy1st_i$ and $boy2nd_i$, and `XGBoost` which only uses additivity-compliant trees. We also consider DDML paired with stacking combining these learners. For comparison, we report TSLS estimates and DDML with unconstrained `XGBoost`. Furthermore, to learn more about the distribution of each estimator, we conduct a simulation exercise involving 500 bootstrap samples of $n_b = 20000$. The

results are shown in Table 3, Panel A, where we list the average coefficient estimates as well as standard deviations over bootstrap iterations in parentheses. Our results indicate little differences among DDML estimators as well as in comparison to TSLS. However, we find that unconstrained `XGBoost` behaves markedly differently from the constrained `XGBoost`, with larger average coefficient estimates and large standard deviation.

AF separately show that the random forest may yield spurious results when the same-sex IV is replaced by a signal-free IV that is generated as $agem_i + educm_i + u_i$ where u_i is drawn from the standard uniform distribution. They also consider an IV with some signal defined as $agem_i + educm_i + u_i \times samesex_i$. With IVs that are pure noise, AF find that the random forest yields implausibly large effects (in absolute value), which are in most cases significant, whereas TSLS standard errors are many times larger compared to TSLS standard errors relying on the original same-sex IV. They identify as one reason that the random forest may fail to purge the effect of the covariates $agem_i$ and $educm_i$ from the IV, reflected in a non-zero correlation between the random-forest-residualized IV and $agem_i + educm_i$. Based on these finding, AF suggest that random forest yields misleading estimates when used to condition on controls in IV regressions.

Table 3: Simulation based on Angrist and Evans (1998)

	TSLS	DDML					
Equation		OLS	lasso	ridge	XGBoost(c)	stacking	XGBoost
Panel A. Original IV:							
First stage	.0693	.0693	.0693	.0693	.0692	.0693	4.122
	(.0066)	(.0066)	(.0066)	(.0066)	(.0068)	(.0066)	(2.6612)
Index	-	0	-.0004	-.0003	0	-.0002	0
	(-)	(0)	(.0005)	(.0004)	(.0003)	(.0003)	(.0001)
Employment	-.1286	-.1284	-.1302	-.1311	-.1294	-.1297	-.1849
	(.0987)	(.0989)	(.0988)	(.099)	(.102)	(.0982)	(.7653)
Weeks worked	–5.6163	–5.6012	–5.6427	–5.7021	–5.6943	–5.6571	–7.9362
	(4.2916)	(4.3211)	(4.3184)	(4.3223)	(4.3898)	(4.279)	(30.4164)
Panel B. Artificial IV:							
First stage	.0004	.0004	.0005	.0035	-.0014	.0004	-.0014
	(.011)	(.011)	(.0109)	(.0127)	(.0055)	(.011)	(.0055)
Index	-	0	.0012	.0127	.0092	.0002	.0092
	(-)	(0)	(0)	(.0246)	(.0003)	(.0005)	(.0003)
Employment	–2.6761	1.1148	2.481	1.1053	-.8519	6.8049	-.655
	(68.4474)	(35.1599)	(30.8941)	(19.6774)	(21.7516)	(140.802)	(19.196)
Weeks worked	–119.6848	33.8069	80.3055	36.4932	–83.1191	292.5573	–61.3623
	(2864.082)	(3084.742)	(1277.043)	(514.9237)	(754.2967)	(6046.517)	(1154.677)
Panel C. IV with some signal:							
First stage	.0828	.0828	.0828	.0723	.0435	.0827	.0235
	(.0104)	(.0104)	(.0104)	(.0208)	(.0059)	(.0105)	(.0057)
Index	-	0	.0012	.0134	.0093	.0003	.0093
	(-)	(0)	(0)	(.0248)	(.0003)	(.0009)	(.0003)
Employment	-.1119	-.1116	-.1039	.0235	-.1036	-.1119	-.1338
	(.1277)	(.1285)	(.1284)	(.3045)	(.1368)	(.1272)	(.2435)
Weeks worked	–5.2283	–5.2204	–4.7685	2.3651	-.7736	–5.1799	1.7544
	(5.6945)	(5.7035)	(5.711)	(16.5527)	(6.3336)	(5.691)	(11.6248)

Notes: 500 bootstrap replications of size $n_b = 20,000$, s.e. in parentheses are calculated as standard deviation over coefficient estimates. OLS, lasso and ridge use 3rd order polynomials and interaction separately for first-child sex dummy × covariates and 2nd-child sex dummy × covariates. XGBoost(c) is impose a constraint on interactions between first and second child sex. Stacking uses OLS, lasso, Ridge and XGBoost(c) as base learners. Row 'Index' reports slope coefficient from regressing residualized IVs against *agem1+educm*. Row 'First stage' reports slope coefficient of regression residualized treatment (i.e., *morekids*) against residualized IV. Rows 'employment' and 'weeks worked' report effect of more than 2 kids on outcome.

In Panel B and C of Table 3, we again put DDML paired with constrained machine

learners to the test. Panel B uses the signal-free IV, Panel C refers to the IV with some signal. The row labeled Index reports the slope coefficient from regressing the residualized IV against $h_i = agem_i + educm_i$. In Panel B, we find that all DDML estimators yield noisy estimates. This includes DDML with stacking for which the bootstrapped standard errors more than twice as large as for TSLS. Turning to the mixed signal IV, Panel C shows that DDML with ridge and `XGBoost` yield estimates considerably different from TSLS, whereas DDML with OLS and DDML with stacking produce estimates and standard errors almost identical to TSLS. We indeed find that the residualized IV exhibits some degree of correlation with h_i in the case of ridge and `XGBoost`, but this is not the case for DDML with stacking where the slope coefficient is almost zero and insignificant.

For further insights, we again consider a calibrated simulation study (similar to Section 3.2). Based on the original data, we generate two artificial samples which match the original data in observed characteristics, but where the empirical reduced forms are either generated with linear regression or gradient-boosted trees to emulate a fully linear or non-linear setting (see Appendix C.2 for details). The advantage is that we know the true target parameter and can flexibly vary the IV strength. Results are shown in Table 4. We report median absolute bias and coverage rates of TSLS and DDML for varying IV strength, which is determined by π. Furthermore, we report rejection rates of the Kleibergen-Paap under-identification test using a 5% nominal level. We find that TSLS and DDML paired with stacking behave very similarly. The results suggest that even when we magnify the non-linear structures in the data, the bias of TSLS is hardly affected, suggesting that linear estimators perform well in this application.

Table 4: Calibrated Monte Carlo simulation based on Angrist and Evans (1998)

	Rejection rates					
	Kleibergen-Paap LM		*Median absolute bias*		*Coverage*	
π	TSLS	DDML	TSLS	DDML	TSLS	DDML
Panel A. Linear DGP						
0	0.048	0.048	1.159	1.154	0.993	0.993
.01	0.174	0.166	0.685	0.693	0.990	0.988
.025	0.776	0.768	0.281	0.287	0.966	0.964
.05	1.	1.	0.135	0.135	0.962	0.960
.075	1.	1.	0.088	0.088	0.956	0.954
.1	1.	1.	0.067	0.065	0.956	0.952
Panel B. Non-linear DGP						
0	0.038	0.038	1.076	1.089	0.990	0.993
.01	0.213	0.215	0.692	0.667	0.980	0.983
.025	0.805	0.795	0.282	0.279	0.961	0.963
.05	1.	1.	0.130	0.136	0.958	0.961
.075	1.	1.	0.083	0.085	0.946	0.946
.1	1.	1.	0.065	0.066	0.956	0.951

Notes: The table shows median absolute bias, coverage rate as well as rejection rates of the Kleibergen-Paap LM under-identification test for TSLS and DDML with stacking for varying IV strength. π determines the strength of the IV. Standard errors are clustered at the observation-ID level. Bootstrap samples of size $n_b = 50{,}000$ with 400 replications. The DDML estimator in this table uses stacking regression with two folds and the following base learners: OLS, CV lasso, CV ridge and constrained `XGBoost`. OLS, CV lasso and CV ridge use third-order polynomials formed separately for $(boy1st_i, x_i)$ and $(boy2nd_i, x_i)$. `XGBoost` uses the untransformed controls, but imposes the constraint of no interactions between $boy1st_i$ and $boy2nd_i$.

We take two lessons from this application. First, machine learning does not protect us from needing to impose constraints necessary for identification. In TSLS estimation, we guarantee identification by *not* including the interaction term $boy1st_i \times boy2nd_i$ as an excluded IV. There is a direct counterpart in tree-based methods where we can, and should, impose a ban on decision rules involving both $boy1st_i$ and $boy2nd_i$. The risk in using flexible methods stems from hidden identification assumptions—in this example, additivity—which, when ignored and not enforced, may lead to spurious results.

Second, DDML with stacking appears robust against the concerns raised by AF for the use of random forests in IV regression with controls. We do not find evidence for imprecise second-stage results when using the original IV or for spuriously precise results when using signal-free IVs (provided that we impose the required constraints). At first glance, the almost identical results between TSLS and DDML with stacking might suggest that there is little value in considering flexible approaches to nuisance function estimation. However, the conclusion would ignore the additional robustness of stacking approaches toward unexpected data structures. Indeed, the downside risks to DDML appear minimal in scenarios where parametric methods work well as long as we include OLS as one of the base learners.

4.3 Importance of Enforcing the Law of Iterated Expectations

The simulations in this section are based on the analysis of Angrist and Krueger (1991), who use quarter of birth (QOB) indicators as IVs to estimate the returns to schooling in a sample of 329,509 American men born between 1930 and 1939. A high-dimensional IV setting arises when interactions between QOB and indicators for year of birth (YOB) and place of birth (POB) are formed. These interactions are motivated by policies differing across cohorts and states that affect how schooling varies with QOB. A fully interacted specification results in 1530 excluded IVs and 510 control variables. The setting of Angrist and Krueger (1991) is thus an interesting application for assessing the performance of DDML with high-dimensional IVs.

We adopt the calibration of a generative model of Angrist and Frandsen (2022) but extend the set of considered estimators. In particular, we consider DDML estimators based on CV lasso and gradient boosting both with and without enforcing the law of iterated expectations in the estimation of the reduced-form equations.

The data generation process of Angrist and Frandsen (2022) is calibrated such that the true effect of schooling on log wages, controlling for year of birth (YOB) and place of birth (POB), is 0.1. A detailed construction of simulated samples is reviewed below:

1. Draw a bootstrap sample, $\mathcal{D}_b$, of size n_b from the empirical distribution of the data.
2. Simulate years of schooling via $\tilde{s}_i \sim \text{Poisson}(\mu_i), \forall i \in \mathcal{D}_b$, where

$$\mu_i = \max\left\{1,\, \overline{s}(QOB_i, YOB_i, POB_i) + \kappa_1 \nu_i\right\},$$

is the sample-average years of education among all individuals with the same combination of QOB, YOB, and POB. Further, $\nu_i \sim \mathcal{N}(0,1)$ and $\kappa_1 = 1.7$.[11]

3. The simulated log wage is then constructed as

$$\tilde{y}_i = \hat{y}(YOB_i, POB_i) + 0.1\tilde{s}_i + \omega(QOB_i, YOB_i, POB_i)(\nu_i + \kappa_2\epsilon_i),$$

$\forall i \in \mathcal{D}_b$, where $\hat{y}(YOB_i, POB_i)$ are second stage fitted values from a LIML estimator on the full sample, $\omega(QOB_i, YOB_i, POB_i)$ is a weight parameter chosen proportionally to the variance of the TSLS residuals in the original data to mimic the heteroskedasticity. Further $\epsilon_i \sim \mathcal{N}(0,1)$ and $\kappa_2 = 0.1$.

A single iteration of the simulation generates a sample of size n_b from this data-generating process, estimates the IV slope coefficient using various estimators, and computes the corresponding bias and t-statistic (where the true parameter is 0.1).[12]

Table 5: Bias and coverage with and without LIE enforcement

Estimator	Bias	Coverage	Bias	Coverage	Bias	Coverage
	n_b = 32,951		n_b = 164,755		b_n = 329,509	
Full sample:						
TSLS	0.092	0.	0.058	0.015	0.040	0.076
TSLS (all IVs)	0.099	0.	0.078	0.	0.061	0.
LIML	0.087	0.952	0.021	0.970	0.013	0.972
LIML (all IVs)	0.077	0.964	0.014	0.970	0.007	0.966
IV-Lasso	0.089	0.046	0.034	0.663	0.023	0.893
IV-Lasso (all IVs)	0.089	0.226	0.076	0.129	0.066	0.254
DDML with LIE enforcement:						
Lasso with CV	0.086	0.969	0.026	0.946	0.016	0.937
Lasso with CV (all IVs)	0.114	0.970	0.026	0.959	0.013	0.943
No LIE enforcement:						
Lasso with CV	0.048	0.515	0.015	0.885	0.012	0.894
Lasso with CV (all IVs)	1.303	0.785	0.271	0.017	0.078	0.087

Notes: The table reports average bias and coverage rates of 95% heteroskedasticity-robust confidence intervals. The controls are quarter-of-birth fixed effects interacted with year of birth. The base set of IVs is quarter of birth, quarter of birth interacted separately with year of birth and place of birth. The extended IV set interacts quarter of birth with year of birth and place of birth. We consider TSLS, the IV-Lasso of Belloni et al. (2012) and DDML with cross-validated lasso with and without LIE enforcement. n_b indicates the bootstrap sample size.

Table 5 gives the median absolute bias (MAB) and coverage rates of conventional two-stage least squares, post-double selection lasso, as well as two sets of DDML estimators based on CV lasso and gradient boosting where the law of iterated was enforced and not enforced, respectively. Importantly, DDML estimates that are allowed to violate the law of iterated expectations exhibit higher MAB and worse coverage rates compared to the other DDML estimates. While the gap in performance decreases with sample size, it persists even in the sample with n_b = 164,755 observations.

[11] Angrist and Frandsen (2022) set κ_1 such that the the first stage R^2 and partial F statistic match those of a 2SLS procedure with IVs $QOB \times POB$ and $QOB \times YOB$.

[12] Note that the data-generating process as described here is dense, not sparse. In the first stage, for example, there are 2040 possible combinations of QOB, YOB, and POB (QOB takes four values, YOB takes 10 values, and POB takes 51 values), each of which corresponds to a unique conditional mean. The simulated data may thus be, at best, approximately sparse.

5 Applications

5.1 Gender gap in citations

Our first applications illustrates DDML by estimating the gender gap in citations. The application is related to the wider literature on gender gaps in various domains, e.g., entry to STEM programs (Card and Payne, 2021), ICT literacy (Siddiq and Scherer, 2019) or wages (Strittmatter and Wunsch, 2021; Bonaccolto-Töpfer and Briel, 2022). The application is also an example of the increasingly frequent appearance of text data in economics and social sciences (e.g., Chen and Ornaghi, 2023; Eberhardt, Facchini, and Rueda, 2022).

We revisit Maliniak, Powers, and Walter (2013, henceforth MPW) who study the gender gap in citations in the International Relations literature. The original study estimates the effect of all-female authorship on citation counts using more than 2,500 articles published between 1980 and 2006. Since female authors might choose to write on different subjects and employ different methodologies than male authors, MPW control for a range of hand-coded variables, including 18 topics (e.g., US foreign policy, human rights), 6 paradigms (e.g., liberal, Marxist), 8 methodologies (e.g., quantitative, formal theory), and three indicators describing the theoretical approach. The authors also control for tenure and journal. Roberts, Stewart, and Nielsen (2020a) extend the analysis by incorporating the full article text into the analysis. They propose a text-matching approach based on structural topic modeling. Grimmer, Roberts, and Stewart (2022) employ a full-sample CV lasso to fit outcome against an unpenalized all-female dummy, MPW controls and word counts, yet without considering sample splitting or double-selection.

Similar to Roberts, Stewart, and Nielsen (2020a) and Grimmer, Roberts, and Stewart (2022), we match article texts retrieved from JSTOR Constellate with the original MPW data.[13] We consider five sets of control variables: no controls, the hand-coded controls of MPW (44 controls), the article text in the form of word counts (92,374), both article text and hand-coded controls (92,408). Splitting text and hand-coded controls allows us to compare a fully data-driven approach to manual coding guided by domain knowledge. In terms of estimation methods, we consider OLS, the full-sample one-step CV lasso as employed by Grimmer, Roberts, and Stewart (2022)), PDS lasso and DDML with CV lasso, CV ridge and CV elastic net (using a mixing parameter of 0.5), and six versions of `XGBoost`. In addition, we use DDML with stacking, which combines these candidate

[13]Due to restrictions on data sharing, we do not have access to exactly the same data set but constructed a similar data set. Since the MPW data does not include the JSTOR identifier, we query the article titles on JSTOR via the search engine DuckDuckGo to retrieve the JSTOR identifier. For the remaining articles, we use fuzzy string matches and verify that either the publication year or journal match. In the processing of the text data, we remove common stop words and non-alphabetical symbols. The final data set includes 2,563 articles, which is similar to the sample size in MPW. We thank Grimmer, Roberts, and Stewart (2022) who kindly shared their code.

learners.

Table 6: Citation penalty for all-female authors. Estimates based on OLS, naïve CV lasso and various DDML estimators.

	Controls variables			
Estimator	*None*	*Hand-coded*	*All*	*Only text*
OLS	-6.44	-7.527		
	(1.981)***	(2.326)***		
Naive Lasso		-7.475	-6.44	-5.976
		(2.298)***	(1.981)***	(1.926)***
PDS-Lasso		-7.65	-5.025	-7.94
		(2.242)***	(2.256)**	(2.117)***
DDML Stacking		-8.009	-8.032	-7.279
		(2.331)***	(2.437)***	(2.096)***
DDML Ridge		-7.772	-7.898	-7.331
		(2.316)***	(2.39)***	(2.052)***
DDML Lasso		-7.684	-7.549	-6.922
		(2.341)***	(2.322)***	(2.013)***
DDML Elastic net		-7.522	-7.971	-6.955
		(2.316)***	(2.41)***	(2.05)***
DDML XGB 1		-8.205	-7.877	-7.076
		(2.261)***	(2.649)***	(2.275)***
DDML XGB 2		-8.172	-6.173	-5.239
		(2.314)***	(3.915)	(2.68)*
DDML XGB 3		-7.836	-3.039	-4.113
		(2.302)***	(3.42)	(3.085)
DDML XGB 4		-8.43	-8.035	-7.551
		(2.269)***	(2.379)***	(2.257)***
DDML XGB 5		-8.076	-6.539	-5.615
		(2.282)***	(3.017)**	(2.466)**
DDML XGB 6		-8.093	-4.789	-3.772
		(2.241)***	(3.587)	(2.893)

Notes: The table shows estimates of the citation penalty for all-female authorship and heteroskedasticity-robust standard errors in parentheses). We consider the following estimators: OLS, naïve CV lasso which fits the outcome against unpenalized treatment and controls, PDS lasso as well as DDMl with the following base learners: CV ridge, CV lasso, CV elastic net (with mixing parameter of 0.5), and six versions of `XGBoost`. All `XGBoost` specifications rely on 1000 trees and enable early stopping after 5 rounds without improvement in predictive performance. The six specifications use the following parameters for the maximum tree depths and the learning rate: $(2, 0.01)$, $(2, 0.05)$, $(2, 0.2)$, $(5, 0.01)$, $(5, 0.05)$, $(5, 0.2)$. Finally, we also consider DDML with stacking using the above base learners. We use 5 re-sampling iterations, 5 cross-fitting folds and 10 cross-validation folds regularized regression. We consider five different sets of control variables (shown in columns): no controls, all controls, only text controls, only hand-coded controls of Maliniak, Powers, and Walter (2013). The sample size is 2,563.

Table 6 shows the citation penalties for all-female authorship. The unconditional citation penalty is -6.44 ($s.e. = 1.98$). If we condition on the hand-coded controls of MPW using OLS, the citation gap widens to -7.53 (2.33). When only considering hand-coded controls, the differences across estimators are relatively small, with point estimates ranging between -7.47 and 8.20. If we add word counts from the original articles into the set of controls, the number of controls increases from 44 to 92,418, and we find a substantial variation across estimators. The full-sample CV Lasso selects no controls and thus yields the same estimate as OLS without controls. This approach, however, is likely biased since it ignores controls that correlate weakly with citations, but strongly with all-female authorship. Indeed, we find that a number of controls are selected when we regress all-female authorship against text controls.

There are also large differences across DDML estimators, especially among the DDML estimators relying on XGBoost. For example, the penalty estimates of DDML with XG-

Boost 3 and XGBoost 4 are −3.04 (3.04) and −8.04 (2.38), highlighting that even the same machine learning algorithm can yield vastly different treatment effects. These stark differences emphasize the need to carefully choose and tune CEF estimators. Without thoroughly validating each base learner, it is impossible to judge which results are more credible.

Stacking provides a framework for tuning learners and discarding ill-suited CEF estimators. Appendix Table D.1 shows that stacking assigns on average relatively small weights to learners which exhibit relatively large cross-fitted mean-squared prediction errors. For example, the stacking weights associated with XGBoost 2 to 6 are close to zero for the estimation of $E[D|X]$. It is also noteworthy that the stacking weights for the estimation of $E[Y|X]$ and $E[D|X]$ differ, which shows that there is no reason to assume that the same learner is best suited for both CEFs. Interestingly, the point estimate of DDML with stacking is close if we exclude hand-coded controls, suggesting that a fully data-driven approach yields almost the same results as hand-coded variables.

5.2 The effect of cash-transfers on longevity

In the second application, we re-examine the analysis of Aizer et al. (2016) and estimate the effect of a cash transfer paid out to mothers in the US over 1911–1935 on children's long-term education, health, and economic outcomes. We focus on children's longevity for this demonstration. The sample includes 7860 children. For identification, the authors use families who applied for the transfer but were rejected as the control group and estimate the effect of longevity using linear regression with a battery of controls, including mother and children characteristics, application year, cohort and county fixed effects (between 34 and 61 controls in total). Słoczyński (2022) re-examines the same analysis and argues that the OLS estimate is a poor proxy for the average treatment effect (ATE) since the OLS estimate is biased towards the average treatment effect on the *un*treated (ATEU). As a solution, Słoczyński (2022) propose to decompose the OLS estimate into the average treatment effect on the treated (ATET) and the ATEU. The decomposition, however, assumes that the CEFs are linear.

For a more flexible approach that allows treatment effects to vary with observables, we relax the assumption that treatment D and controls X are separable. In addition, we assume overlap and that treatment assignment is random conditional on observables. We summarize the required assumptions:

Assumption 4 (Interactive Model)**.** *We assume* $Y = g_0(D, X) + U$*, conditional mean independence* $E[U|D, X] = 0$ *and overlap condition,* $\Pr(D = 1|X) \in (0, 1)$.

Under Assumption 4, we can use the Neyman-orthogonal score

$$\psi(W; \theta_0^{\text{ATE}}, \eta_0) = \frac{D(Y - g_0(1, X))}{m_0(X)} - \frac{(1 - D)(Y - g_0(0, X))}{1 - m_0(X)} + g_0(1, X) - g_0(0, X) - \theta_0^{\text{ATE}}$$

Table 7: The effect of cash transfers to mothers on their children's longevity.

	(1)	(2)	(3)	(4)	(5)
	OLS	OLS	OLS	DDML	
				OLS/Logit	Extended
Panel A.					
Accepted	0.0157*	0.0158*	0.0182*	0.0154*	0.0154*
	(0.00646)	(0.00655)	(0.00694)	(0.00677)	(0.00670)
Panel B.					
ATET	0.0129*	0.0149*	0.00967	0.0126	0.0173**
	(0.00606)	(0.00694)	(0.00790)	(0.0112)	(0.00632)
ATEU	0.0162**	0.0160*	0.0206**	0.0148*	0.0145*
	(0.00566)	(0.00631)	(0.00690)	(0.00677)	(0.00647)
ATE	0.0133*	0.0150*	0.0110	0.0124	0.0167**
	(0.00594)	(0.00666)	(0.00747)	(0.00965)	(0.00607)
State fixed effects	Yes			Yes	Yes
County fixed effects			Yes	Yes	Yes
Cohort fixed effects	Yes	Yes	Yes	Yes	Yes
State characteristics		Yes	Yes	Yes	Yes
County characteristics		Yes		Yes	Yes
Individual characteristics		Yes	Yes	Yes	Yes
Observations	7860	7859	7859	7859	7859

Notes: The table shows the effect of a cash transfer to mothers on children's longevity. Panel A, column 1-3, replicates Aizer et al. (2016, Table 4). Panel B, column 1-3, replicates Słoczyński (2022, Table 2) using the Stata package `hettreatreg` (Sloczynski, 2019). Panel A, column 4-5 uses DDML with stacking using the estimator (5). Panel B, column 4-5 also uses DDML with stacking but relies on the Interactive Model in Assumption 4.
The DDML estimator in column 4 uses OLS for the outcome equations and logit for estimating propensity scores, both with the three sets of controls corresponding to Columns 1-3. Column 5 adds CV lasso and ridge with all controls, random forests (RF) with low regularization (all controls, maximum tree depth of 10, 500 trees and approximately $\sqrt{p}$ features considered at each split), RF with medium regularization (same as previous but maximum tree depth of 6), RF with low regularization (maximum tree depth of 2), gradient-boosted trees (GB) with low regularization (all controls, 1000 trees, learning rate of 0.3, early stopping which 20% validation), GB with medium regularization (learning rate of 0.1), GB with high regularization (learning rate of 0.01).
Significance codes: * p<0.05, ** p<0.01, *** p<0.001

to identify the ATE θ_0. As in the partially linear models, the score depends on conditional expectation functions, which are collected in the nuisance term $\eta_0 := (g_0(D, X), m_0(X))$. The score for the ATET and ATEU can be defined similarly.

We report results in Table 7. The first three columns in Panel A replicate Aizer et al. (2016) and use three different sets of control variables as indicated at the bottom of the table. Columns 1-3 in Panel B replicate Słoczyński (2022) who decomposes the OLS estimates into ATE, ATET and ATEU under the linearity assumption. Note that while the results of Aizer et al. (2016) do not change much depending on the chosen set of controls, the ATET and ATE estimates of Słoczyński (2022) are smaller in comparison and insignificant when county fixed effects are included.

The final two columns show results for DDML combined with several stacking approaches. Column 4 uses OLS for the outcome equations and logit for estimating propensity scores, both with the three sets of controls corresponding to Columns 1-3. Column 5 adds regularized regression, random forests and gradient-boosted trees as base learners (see table notes). We use the DDML-PLM estimator in (5) in Panel A, and the interactive DDML estimators of average treatment effects in Panel B.

The DDML results based on the Partially Linear Model in Panel A are similar to the estimates of Aizer et al. (2016). Stacking puts the largest weights on the specification in Columns 1. The ATE and ATET estimates of the fully parametric DDML estimates in Column 4 are similar to the OLS-based estimates of Słoczyński (2022), suggesting that the OLS estimates in Panel A over-estimate the effect of the cash transfer. However, the DDML estimates in Panel B relying on an extended set of nonparametric estimators yield ATE and ATET that are larger and more precisely estimated. The ATET estimate from the extended DDML estimator is almost twice as large compared to the lowest OLS-based ATET estimate (0.0173 compared to 0.009), revealing noticeable differences when accounting for unknown nonlinear structures. Indeed, the stacking estimators assigns significant weights to tree-based estimators in each CEF estimation.

6 Conclusion

We indeed find there are pitfalls to leveraging machine learning for causal inference. The main risk arises when using supervised machine learners in applications where the regularization assumption does not match the underlying data-generating process. We highlight that DDML paired with stacking provides a solution and assess the performance of DDML with stacking in realistic settings using applications and simulation studies calibrated to real economic data. In addition, we introduce DDML with short-stacking, a variant of DDML with stacking, that is computationally less expensive and shows better small-sample performance.

Our findings overall support the use of machine learning for causal inference, provided

that a diverse set of candidate learners is considered. While machine-learning-based causal methods may yield fundamentally different results from linear regression only in specific examples, we show that DDML with stacking accommodates traditional parametric approaches and can help to decide between alternative models. There, thus, are good reasons for shifting from purely parametric to more flexible semi-parametric approaches that are robust to unexpected structures in the data.

References

Ahrens, Achim, Christian B. Hansen, and Mark E. Schaffer (2022). *pystacked: Stacking generalization and machine learning in Stata*. URL: https://arxiv.org/abs/2208.10896.

Ahrens, Achim, Christian B. Hansen, Mark E. Schaffer, and Thomas Wiemann (2023). *ddml: Double/debiased machine learning in Stata*. URL: https://arxiv.org/abs/2301.09397.

Aizer, Anna, Shari Eli, Joseph Ferrie, and Adriana Lleras-Muney (2016). "The Long-Run Impact of Cash Transfers to Poor Families". *American Economic Review* 106.4, pp. 935–971.

Angrist, Joshua D and William N Evans (1998). "Children and Their Parents' Labor Supply: Evidence from Exogenous Variation in Family Size". *American Economic Review*, pp. 450–477.

Angrist, Joshua D and Brigham Frandsen (2022). "Machine labor". *Journal of Labor Economics* 40.S1, S97–S140.

Angrist, Joshua D and Alan B Krueger (1991). "Does compulsory school attendance affect schooling and earnings?" *Quarterly Journal of Economics* 106.4, pp. 979–1014.

— (1999). "Empirical strategies in labor economics". In: *Handbook of Labor Economics*. Vol. 3. Elsevier, pp. 1277–1366.

Athey, Susan, Julie Tibshirani, and Stefan Wager (2019). "Generalized random forests". *Annals of Statistics* 47.2, pp. 1148–1178.

Athey, Susan and Stefan Wager (2021). "Policy Learning With Observational Data". *Econometrica* 89.1, pp. 133–161.

Belloni, A, V Chernozhukov, I Fernández-Val, and C Hansen (2017). "Program Evaluation and Causal Inference With High-Dimensional Data". *Econometrica* 85.1, pp. 233–298.

Belloni, Alexandre, D Chen, Victor Chernozhukov, and Christian Hansen (2012). "Sparse Models and Methods for Optimal Instruments With an Application to Eminent Domain". *Econometrica* 80.6, pp. 2369–2429.

Belloni, Alexandre, Victor Chernozhukov, and Christian Hansen (2014). "Inference on treatment effects after selection among high-dimensional controls". *Review of Economic Studies* 81, pp. 608–650.

Bickel, Peter J, Ya'acov Ritov, and Alexandre B Tsybakov (2009). "Simultaneous analysis of Lasso and Dantzig selector". *Annals of statistics* 37.4, pp. 1705–1732.

Blandhol, Christine, John Bonney, Magne Mogstad, and Alexander Torgovitsky (2022). "When is TSLS Actually LATE?" *BFI Working Paper* 2022-16.

Bonaccolto-Töpfer, Marina and Stephanie Briel (2022). "The gender pay gap revisited: Does machine learning offer new insights?" *Labour Economics* 78, p. 102223.

Breiman, Leo (1996). "Stacked regressions". *Machine Learning* 24.1, pp. 49–64.

Buitinck, Lars et al. (2013). "API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122.

Card, David and A. Abigail Payne (2021). "High School Choices and the Gender Gap in Stem". *Economic Inquiry* 59.1, pp. 9–28.

Chen, Daniel L and Arianna Ornaghi (2023). "Gender Attitudes in the Judiciary: Evidence from US Circuit Courts". *American Economic Journal: Applied Economics*.

Chen, Tianqi and Carlos Guestrin (2016). "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794.

Chernozhukov, Victor, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins (2018a). "Double/debiased machine learning for treatment and structural parameters". *The Econometrics Journal* 21.1, pp. C1–C68.

Chernozhukov, Victor, Mert Demirer, Esther Duflo, and Ivan Fernandez-Val (2018b). *Generic machine learning inference on heterogeneous treatment effects in randomized experiments, with an application to immunization in India*. Tech. rep. National Bureau of Economic Research.

Chernozhukov, Victor, Christian Hansen, and Martin Spindler (2015). "Post-Selection and Post-Regularization Inference in Linear Models with Many Controls and Instruments". *American Economic Review* 105.5, pp. 486–490.

Eberhardt, Markus, Giovanni Facchini, and Valeria Rueda (2022). "Gender Differences in Reference Letters: Evidence from the Economics Job Market". *SSRN Electronic Journal*.

Farrell, Max H, Tengyuan Liang, and Sanjog Misra (2021). "Deep neural networks for estimation and inference". *Econometrica* 89.1, pp. 181–213.

Giannone, Domenico, Michele Lenza, and Giorgio E Primiceri (2021). "Economic predictions with big data: The illusion of sparsity". *Econometrica* 89.5, pp. 2409–2437.

Gilchrist, Duncan Sheppard and Emily Glassberg Sands (2016). "Something to talk about: Social spillovers in movie consumption". *Journal of Political Economy* 124.5, pp. 1339–1382.

Goller, Daniel, Michael Lechner, Andreas Moczall, and Joachim Wolff (2020). "Does the estimation of the propensity score by machine learning improve matching estimation? The case of Germany's programmes for long term unemployed". *Labour Economics* 65, p. 101855.

Grimmer, Justin, Margaret E Roberts, and Brandon M Stewart (2022). *Text as data: A new framework for machine learning and the social sciences*. Princeton University Press.

Hangartner, Dominik, Daniel Kopp, and Michael Siegenthaler (2021). "Monitoring hiring discrimination through online recruitment platforms". *Nature* 589.7843, pp. 572–576.

Hansen, Bruce E. and Jeffrey S. Racine (2012). "Jackknife model averaging". *Journal of Econometrics* 167.1, pp. 38–46.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The Elements of Statistical Learning*. 2nd ed. New York: Springer-Verlag.

Knaus, Michael C, Michael Lechner, and Anthony Strittmatter (2021). "Machine learning estimation of heterogeneous causal effects: Empirical Monte Carlo evidence". *The Econometrics Journal* 24.1, pp. 134–161.

Künzel, Sören R., Jasjeet S. Sekhon, Peter J. Bickel, and Bin Yu (2019). "Metalearners for estimating heterogeneous treatment effects using machine learning". *Proceedings of the National Academy of Sciences* 116.10, pp. 4156–4165.

Laan, Mark J. van der, Sandrine Dudoit, and Aad W. van der Vaart (2006). "The cross-validated adaptive epsilon-net estimator". *Statistics & Decisions* 24.3, pp. 373–395.

Laan, Mark J Van der, Sherri Rose, et al. (2011). *Targeted learning: causal inference for observational and experimental data*. Vol. 4. Springer.

Laan, Mark J. van der, Eric C Polley, and Alan E. Hubbard (2007). "Super Learner". *Statistical Applications in Genetics and Molecular Biology* 6.1.

Luo, Ye, Martin Spindler, and Jannis Kück (2022). "High-Dimensional L_2 Boosting: Rate of Convergence". *arXiv preprint arXiv:1602.08927*.

Maliniak, Daniel, Ryan Powers, and Barbara F. Walter (2013). "The Gender Citation Gap in International Relations". *International Organization* 67.4, pp. 889–922.

Poterba, James M, Steven F Venti, and David A Wise (1995). "Do 401 (k) contributions crowd out other personal saving?" *Journal of Public Economics* 58.1, pp. 1–32.

Roberts, Margaret E., Brandon M. Stewart, and Richard A. Nielsen (2020a). "Adjusting for Confounding with Text Matching". *American Journal of Political Science* 64.4, pp. 887–903.

— (2020b). "Adjusting for confounding with text matching". *American Journal of Political Science* 64.4, pp. 887–903.

Robinson, P. M. (1988). "Root-N-Consistent Semiparametric Regression". *Econometrica* 56.4, p. 931.

Schmidt-Hieber, Johannes (2020). "Nonparametric regression using deep neural networks with ReLU activation function". *Annals of Statistics* 48.4, pp. 1875–1897.

Siddiq, Fazilat and Ronny Scherer (2019). "Is there a gender gap? A meta-analysis of the gender differences in students' ICT literacy". *Educational Research Review* 27, pp. 205–217.

Sloczynski, Tymon (2019). *HETTREATREG: Stata module to compute diagnostics for linear regression when treatment effects are heterogeneous*. Statistical Software Components, Boston College Department of Economics. URL: `https://ideas.repec.org/c/boc/bocode/s458715.html`.

Strittmatter, Anthony and Conny Wunsch (2021). *The Gender Pay Gap Revisited with Big Data: Do Methodological Choices Matter?* URL: https://arxiv.org/abs/2102.09207.

Słoczyński, Tymon (2022). "Interpreting OLS Estimands When Treatment Effects Are Heterogeneous: Smaller Groups Get Larger Weights". *The Review of Economics and Statistics* 104.3, pp. 501–509.

Wager, Stefan and Susan Athey (2018). "Estimation and Inference of Heterogeneous Treatment Effects using Random Forests". *Journal of the American Statistical Association* 113.523, pp. 1228–1242.

Wager, Stefan and Guenther Walther (2016). "Adaptive concentration of regression trees, with application to random forests". *arXiv preprint arXiv:1503.06388*.

Wolpert, David H. (1992). "Stacked generalization". *Neural Networks* 5.2, pp. 241–259.

Wolpert, David H (1996). "The lack of a priori distinctions between learning algorithms". *Neural computation* 8.7, pp. 1341–1390.

Wüthrich, Kaspar and Ying Zhu (2021). "Omitted variable bias of Lasso-based inference methods: A finite sample analysis". *Review of Economics and Statistics* 0.(0), pp. 1–47.

Supplementary material

A Additional tables 'The benefits of pairing DDML and stacking'

Table A.1: Mean-squared prediction error

	$n_b = 9{,}915$		$n_b = 99{,}150$	
Panel (A): Linear DGP	$Y\|X$ (1)	$D\|X$ (2)	$Y\|X$ (3)	$D\|X$ (4)
Base learners				
OLS	3.093	0.200	3.088	0.200
Lasso with CV (2nd order poly)	3.095	0.200	3.089	0.200
Ridge with CV (2nd order poly)	3.100	0.200	3.089	0.200
Lasso with CV (10th order poly)	3.218	0.202	3.095	0.200
Ridge with CV (10th order poly)	3.340	0.205	3.093	0.200
Random forest (low regularization)	3.613	0.233	3.698	0.239
Random forest (high regularization)	3.182	0.205	3.197	0.207
Gradient boosting (low regularization)	3.130	0.201	3.102	0.200
Gradient boosting (high regularization)	3.151	0.201	3.138	0.200
Neural net	3.657	18.260	3.408	9.864
Base learners				
Stacking: CLS	3.097	0.200	3.089	0.200
Stacking: Single-best	3.097	0.200	3.088	0.200
Panel (B): Non-Linear DGP	$Y\|X$ (5)	$D\|X$ (6)	$Y\|X$ (7)	$D\|X$ (8)
Base learners				
OLS	3.683	0.203	3.668	0.203
Lasso with CV (2nd order poly)	3.478	0.201	3.446	0.200
Ridge with CV (2nd order poly)	3.476	0.201	3.446	0.200
Lasso with CV (10th order poly)	5.286	0.228	3.419	0.200
Ridge with CV (10th order poly)	6.340	0.236	3.421	0.200
Random forest (low regularization)	3.793	0.231	3.514	0.236
Random forest (high regularization)	3.589	0.204	3.250	0.205
Gradient boosting (low regularization)	3.351	0.200	3.094	0.198
Gradient boosting (high regularization)	3.403	0.201	3.215	0.199
Neural net	4.214	20.126	4.064	8.365
Meta learners				
Stacking: CLS	3.640	0.203	3.062	0.198
Stacking: Single-best	3.981	0.206	3.095	0.198

Notes: The table shows the mean-squared prediction error of each base learner. The bootstrap sample size is 9,915 (left) and 99,150 (right).
Results are based on 1,000 replications. See Table 1 for more information.

B Additional results 'DDML and stacking in very small samples'

Estimator	*Estimate*	*Std. error*
Panel A. No sample splitting		
OLS QSI	5988.413	2033.021
OLS TWI	6751.907	2067.859
Post double Lasso QSI c=0.5	5648.14	1985.495
Post double Lasso QSI c=1	4646.575	2012.597
Post double Lasso QSI c=1.5	4472.324	2025.874
Post double Lasso TWI c=0.5	6562.923	2062.475
Post double Lasso TWI c=1	6630.751	2064.408
Post double Lasso TWI c=1.5	7474.508	2049.787
Panel B. DDML with $K = 2$		
Stacking	5689.147	2003.847
OLS	6472.339	2116.396
Lasso with CV (TWI)	6897.007	2066.365
Ridge with CV (TWI)	6967.574	2062.864
Lasso with CV (QSI)	5460.024	2026.607
Ridge with CV (QSI)	5946.86	2005.534
Random forest (low regularization)	7090.91	2150.247
Random forest (high regularization)	22164.99	2309.582
Gradient boosting (low regularization)	6958.092	2088.807
Gradient boosting (high regularization)	7985.865	2119.756
Minimum MSE	5440.646	1997.503
Short-stacking	5623.86	1996.515
Panel C. DDML with $K = 10$		
Stacking	5780.801	1997.006
OLS	6457.161	2116.448
Lasso with CV (TWI)	6777.813	2066.669
Ridge with CV (TWI)	6747.91	2051.715
Lasso with CV (QSI)	5730.387	2012.293
Ridge with CV (QSI)	6007.533	1998.671
Random forest (low regularization)	6969.11	2151.209
Random forest (high regularization)	22189.08	2307.768
Gradient boosting (low regularization)	6995.855	2081.898
Gradient boosting (high regularization)	7991.744	2111.687
Minimum MSE	5708.67	2006.347
Short-stacking	5834.879	1994.693

Notes: In the case of DDML estimators, the average estimates and standard errors are based on 300 replications. Panel A is reproduced from Table 1 in WZ.

Table B.1: Estimates and std. errors based on the full sample ($N = 9{,}915$).

	200	400	600	800	1200	1600
Panel A. No sample splitting						
OLS QSI		-1271.8	-383.9	-935.3	-553	-939.9
OLS TWI	-2642.9	-843.4	-147.7	-636.5	-230.7	-615.4
Post double Lasso QSI c=0.5	-1117.1	-337.9	-204	-599.2	-340.3	-760.9
Post double Lasso QSI c=1	-1531.6	-1179.8	-763	-1120.5	-794	-955.7
Post double Lasso QSI c=1.5	7619.6	1067.8	-1516.7	-2534.6	-2030.4	-2165.1
Post double Lasso TWI c=0.5	2589.5	2243.7	2142.1	1663.3	1399.3	872.3
Post double Lasso TWI c=1	4841.4	3409.9	3121.6	2300.2	1772.6	1331.6
Post double Lasso TWI c=1.5	13124.8	8868.1	5899.3	4426.6	3758.3	2289
Panel B. DDML with $K = 2$						
Stacking	362.6	540.4	118.1	-306.5	-148.5	-360.7
OLS	-408.8	-138	-57.5	-248.9	72.8	-450.3
Lasso with CV (TWI)	5858.6	4726.8	3890.4	3079.6	2244.4	1085.4
Ridge with CV (TWI)	3783.4	3249.5	2447.5	1758.2	1321.3	516.5
Lasso with CV (QSI)	266.7	-98.5	82.2	-146.3	-138.5	-462.8
Ridge with CV (QSI)	431.9	25.6	477.1	-123.7	95.4	-411.7
Random forest (low regularization)	680.3	682.2	734.3	514.9	645.2	71.7
Random forest (high regularization)	-895.2	-869.8	-566.2	-534	-458.4	-432.1
Gradient boosting (low regularization)	-694.8	-158.2	-357.8	-17.9	170.4	-361.4
Gradient boosting (high regularization)	-538.4	54.1	-19.7	179.5	259	-125.3
Minimum MSE	835.4	608.3	316.6	134.1	310.8	-191.1
Short-stacking	1001	553.8	427.7	83.7	265.7	-185.1
Panel C. DDML with $K = 10$						
Stacking	-561	-164.4	-65	-205.5	-66.1	-435.7
OLS	-912.5	-135	59.5	-202.7	2.1	-342.6
Lasso with CV (TWI)	4474.5	3270.1	2484	1840.1	1314.3	491
Ridge with CV (TWI)	2953.4	2057.9	1627.9	1103.8	1179.6	736.1
Lasso with CV (QSI)	-978.8	-692.7	-384.6	-591.7	-370.8	-674
Ridge with CV (QSI)	-650.3	-281.9	133	-154.1	-21.3	-404.4
Random forest (low regularization)	489.7	727.2	1003.9	703.4	565.9	193
Random forest (high regularization)	-967.5	-804.4	-449.4	-424	-331.7	-304.3
Gradient boosting (low regularization)	-1058.9	.6	-177.9	-127.5	-8.1	-338.8
Gradient boosting (high regularization)	-774.8	301.8	25	69.6	155.9	-190.7
Minimum MSE	-560.1	-133.7	51.8	-175.5	34.1	-431.8
Short-stacking	-506.1	-106.6	60.4	-187.9	82.5	-364.5

Notes: This table shows the extended results of the bootstrapping exercise in Figure 5. See notes below Figure 5 for more information.

Table B.2: Mean bias relative to full sample

Estimator	*Observations*						
	200	400	600	800	1200	1600	9915
Panel A. $E[Y\|X]$, $K = 10$							
OLS	.159	.159	.127	.096	.056	.027	0
Lasso with CV (TWI)	.046	.047	.033	.034	.057	.066	.121
Ridge with CV (TWI)	.062	.06	.061	.056	.041	.027	.084
Lasso with CV (QSI)	.268	.287	.291	.296	.309	.297	.567
Ridge with CV (QSI)	.208	.279	.324	.368	.39	.435	.252
Random forest (low regularization)	.165	.123	.117	.105	.086	.082	.002
Random forest (high regularization)	.044	.013	.004	.003	.001	0	0
Gradient boosting (low regularization)	.048	.05	.065	.073	.091	.1	0
Gradient boosting (high regularization)	.037	.022	.016	.009	.007	.003	0
Panel B. $E[D\|X]$, $K = 10$							
OLS	.141	.22	.233	.258	.242	.212	.173
Lasso with CV (TWI)	.046	.037	.031	.021	.017	.018	.14
Ridge with CV (TWI)	.054	.023	.017	.023	.03	.022	0
Lasso with CV (QSI)	.164	.215	.248	.254	.23	.204	.388
Ridge with CV (QSI)	.201	.167	.099	.073	.08	.097	0
Random forest (low regularization)	.135	.154	.214	.219	.253	.268	.125
Random forest (high regularization)	.159	.067	.028	.019	.005	.002	0
Gradient boosting (low regularization)	.035	.05	.064	.073	.104	.154	.172
Gradient boosting (high regularization)	.063	.068	.067	.06	.039	.023	.002

Notes: The table reports the stacking weights corresponding to the DDML stacking estimator in Figure 5. The stacking weights are averages over folds, based on 10-fold cross-fitting and shows for the estimation of $E[Y|X]$ and $E[D|X]$ in Panel A and B, respectively. See notes below Figure 5 for more information.

Table B.3: Stacking weights

C Additional results 'Flexible controls in IV regression'

C.1 Additional tables

Table C.1: Stacking weights

Estimator	Employment	Weeks worked	> 2 children	IVs		
				Same sex	Noise	Semi-noise
OLS	.401	.321	.449	.223	.832	.842
CV-Lasso	.365	.374	.26	.281	.144	.128
CV-Ridge	.014	.027	.018	.415	.022	.029
XGBoost-c	.22	.278	.273	.082	.001	.002

Notes: The table shows the average stacking weights of the DDML stacking estimator by outcome variable and by base learner. See Table 3 for more information.

C.2 Calibrated simulation

1. We generate the partial residuals $d_i^{(r)} = \texttt{morekids}_i - \hat{\pi}_{OLS}$ and $y_i^{(r)} = \texttt{workedm}_i - \hat{\theta}_{2SLS}$, where $\hat{\pi}_{OLS}$ and $\hat{\pi}_{OLS}$ are the full sample estimates $\hat{\theta}_{2SLS}$ using controls $(boy1st_i, boy2nd_i, X_i)$.

2. Then, fit a supervised learning estimator that aims to predict $y_i^{(r)}$ with the controls x_i. Denote the fitted estimator by $\tilde{g}_y$. Similarly, fit a supervised learning estimator that aims to predict $d_i^{(r)}$ and $\texttt{samesex}_i$ with x_i and denote the fitted estimator by $\tilde{g}_d$ and $\tilde{g}_z$. The two estimators considered in this exercise are linear regression and gradient boosting.

3. Sample from the empirical distribution of x_i by bootstrapping n_b observations from the original data. Denote the bootstrapped sample by $\mathcal{D}_b$.

4. To generate the simulated sample:

$$\begin{aligned}
\tilde{y}_i^{(b)} &= \mathbb{1}\{\theta_0 \tilde{d}_i^{(b)} + \tilde{g}_y(x_i) + \varepsilon_i > 0.5\}, \varepsilon_i \overset{iid}{\sim} \mathcal{N}(0, \kappa_y) \\
\tilde{d}_i^{(b)} &= \mathbb{1}\{\pi_0 \tilde{z}_i^{(b)} + \tilde{g}_d(x_i) + u_i \geq 0.486\}, u_i \overset{iid}{\sim} \mathcal{N}(0, \kappa_d) \\
\tilde{z}_i^{(b)} &= \mathbb{1}\{\tilde{g}_z(x_i) + \nu_i \geq 0.5\}, \nu_i \overset{iid}{\sim} \mathcal{N}(0, \kappa_z) \qquad \forall i \in \mathcal{D}_b,
\end{aligned}$$

   where $\kappa_y = \kappa_d = \kappa_z = 0.35$. We set $\theta_0 = -0.1$.

D Text as confounders

Table D.1: Stacking weights and mean-squared prediction error.

Learner	Hand coded	All	Text only	Hand coded	All	Text only
	Panel A. Stacking weights					
	$E[Y\|X]$			$E[D\|X]$		
Ridge	0.233	0.200	0.219	0.265	0.031	0.222
Elastic net	0.233	0.225	0.270	0.251	0.018	0.238
Lasso	0.210	0.159	0.217	0.267	0.015	0.215
XGB 1	0.148	0.076	0.065	0.079	0.720	0.255
XGB 2	0.088	0.052	0.038	0.042	0.095	0.028
XGB 3	0.062	0.099	0.076	0.031	0.023	0.012
XGB 4	0.000	0.048	0.018	0.018	0.095	0.025
XGB 5	0.007	0.048	0.024	0.004	0.000	0.002
XGB 6	0.019	0.091	0.074	0.043	0.003	0.003
	Panel B. Mean-squared prediction error					
	$E[Y\|X]$			$E[D\|X]$		
Ridge	2244.866	2660.745	2666.535	0.076	0.083	0.099
Elastic net	2244.735	2661.533	2704.363	0.076	0.082	0.099
Lasso	2244.861	2651.400	2745.828	0.076	0.082	0.099
XGB 1	2271.782	2672.759	2795.575	0.077	0.079	0.100
XGB 2	2260.094	2862.695	3005.694	0.077	0.083	0.109
XGB 3	2265.449	3058.253	3180.970	0.077	0.099	0.131
XGB 4	2280.125	2668.943	2796.651	0.077	0.079	0.101
XGB 5	2273.701	2865.233	3011.869	0.077	0.085	0.112
XGB 6	2278.368	3018.737	3187.590	0.078	0.104	0.137

Notes: The table shows stacking weights in Panel A and cross-fitted mean-squared prediction error (MSPE) in Panel B. The stacking weights are averaged over folds and over cross-fitting repetitions. The MSPEs are averaged over cross-fitting repetitions. See Table 6 for more information.