

A Algorithms

□ Algorithm A.1. DDML for the Interactive Model.

Split the sample $\{(Y_i, D_i, \mathbf{X}_i)\}_{i \in I}$ with $I = \{1, \dots, n\}$ in K folds of approximately equal size. Denote I_k the set of observations included in fold k and $I_k^c = I \setminus I_k$ its complement.

1. Estimate conditional expectations. For each k :
 - a. Fit the CEF estimator to observations in the sub-sample I_k^c for which $D_i = 1$ using Y_i as the outcome and \mathbf{X}_i as predictors to estimate the conditional expectation $g_0(1, \mathbf{X}) = E[Y|\mathbf{X}, D = 1]$. Obtain the out-of-sample predicted values $\hat{g}_{I_k^c}(1, \mathbf{X}_i)$ for $i \in I_k$. Proceed in the same way to obtain $\hat{g}(0, \mathbf{X})$.
 - b. For each k , fit the CEF estimator to the sub-sample I_k^c using D_i as outcome and \mathbf{X}_i as predictors to estimate the conditional expectation $m(\mathbf{X}) = E[D|\mathbf{X}]$. Obtain the out-of-sample predicted values $\hat{m}_{I_k^c}(\mathbf{X}_i)$ for $i \in I_k$.
2. Compute the ATE and ATET using (7) and (8).

□

□ Algorithm A.2. DDML for the Partially Linear IV Model.

Split the sample $\{(Y_i, D_i, \mathbf{X}_i)\}_{i \in I}$ with $I = \{1, \dots, n\}$ in K folds of approximately equal size. Denote I_k the set of observations included in fold k and $I_k^c = I \setminus I_k$ its complement.

1. Estimate conditional expectations. For each k :
 - a. Fit the CEF estimator to the sub-sample I_k^c using Y_i as outcome and \mathbf{X}_i as predictors to estimate the conditional expectation $\ell_0(\mathbf{X}) = E[Y|\mathbf{X}]$. Obtain the out-of-sample predicted values $\hat{\ell}_{I_k^c}(\mathbf{X}_i)$ for $i \in I_k$.
 - b. Fit the CEF estimator to the sub-sample I_k^c using D_i as outcome and \mathbf{X}_i as predictors to estimate the conditional expectation $m_0(\mathbf{X}) = E[D|\mathbf{X}]$. Obtain the out-of-sample predicted values $\hat{m}_{I_k^c}(\mathbf{X}_i)$ for $i \in I_k$.
 - c. Fit the CEF estimator to the sub-sample I_k^c using Z_i as outcome and \mathbf{X}_i as predictors to estimate the conditional expectation $r_0(\mathbf{X}) = E[Z|\mathbf{X}]$. Obtain the out-of-sample predicted values $\hat{r}_{I_k^c}(\mathbf{X}_i)$ for $i \in I_k$.
2. Compute (10).

□

□ Algorithm A.3. DDML for the Interactive IV Model.

Split the sample $\{(Y_i, D_i, \mathbf{X}_i)\}_{i \in I}$ with $I = \{1, \dots, n\}$ in K folds of approximately equal size. Denote I_k the set of observations included in fold k and $I_k^c = I \setminus I_k$ its complement.

1. Estimate conditional expectations. For each k :

- a. Fit the CEF estimator to observations in the sub-sample I_k^c for which $Z_i = 1$ using Y_i as outcome and \mathbf{X}_i as predictors to estimate the conditional expectation $\ell_0(1, \mathbf{X}) = E[Y|\mathbf{X}, Z = 1]$. Obtain the out-of-sample predicted values $\hat{\ell}_{I_k^c}(1, \mathbf{X}_i)$ for $i \in I_k$. Proceed in the same way for the estimation of $\ell_0(0, \mathbf{X})$.
 - b. Fit the CEF estimator to observations in the sub-sample I_k^c for which $Z_i = 1$ using D_i as outcome and \mathbf{X}_i as predictors to estimate the conditional expectation $p_0(1, \mathbf{X}) = \Pr(D = 1|\mathbf{X}, Z = 1)$. Obtain the out-of-sample predicted values $\hat{p}_{I_k^c}(1, \mathbf{X}_i)$ for $i \in I_k$. Proceed in the same way for the estimation of $p_0(0, \mathbf{X})$.
 - c. Fit the CEF estimator to the sub-sample I_k^c using D_i as outcome and \mathbf{X}_i as predictors to estimate the conditional expectation $r(\mathbf{X}) = E[Z|\mathbf{X}]$. Obtain the out-of-sample predicted values $\hat{r}_{I_k^c}(\mathbf{X}_i)$ for $i \in I_k$.
2. Compute (17).

□

□ **Algorithm A.4. DDML with short-stacking for the Partially Linear Model.**

Split the sample $\{(Y_i, D_i, \mathbf{X}_i)\}_{i \in I}$ with $I = \{1, \dots, n\}$ in K folds of approximately equal size. Denote I_k the set of observations included in fold k and $I_k^c = I \setminus I_k$ its complement. Select a set of J base learners with $J \geq 2$.

1. Estimate conditional expectations. For each k and base learner j :
 - a. Fit a CEF estimator j to the sub-sample I_k^c using Y_i as the outcome and \mathbf{X}_i as predictors. Obtain the out-of-sample predicted values $\hat{\ell}_{I_k^c}^{(j)}(\mathbf{X}_i)$ for $i \in I_k$.
 - b. Fit a CEF estimator j to the sub-sample I_k^c using D_i as the outcome and \mathbf{X}_i as predictors. Obtain the out-of-sample predicted values $\hat{m}_{I_k^c}^{(j)}(\mathbf{X}_i)$ for $i \in I_k$.
2. Short-stacking:
 - a. Apply constrained regression of Y_i against $\hat{\ell}_{I_k^c}^{(1)}(\mathbf{X}_i), \dots, \hat{\ell}_{I_k^c}^{(J)}(\mathbf{X}_i)$ using the full sample I , which yields the short-stacked predicted values $\hat{\ell}^*(\mathbf{X}_i)$.
 - b. Apply constrained regression of D_i against $\hat{m}_{I_k^c}^{(1)}(\mathbf{X}_i), \dots, \hat{m}_{I_k^c}^{(J)}(\mathbf{X}_i)$ using the full sample I , which yields the short-stacked predicted values $\hat{m}^*(\mathbf{X}_i)$.
3. Compute the short-stacked DDML estimator using

$$\hat{\theta}_n = \frac{\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{\ell}^*(\mathbf{X}_i))(D_i - \hat{m}^*(\mathbf{X}_i))}{\frac{1}{n} \sum_{i=1}^n (D_i - \hat{m}^*(\mathbf{X}_i))^2}.$$

□

□ **Algorithm A.5. DDML with short-stacking for the Flexible IV Model.**

Split the sample $\{(Y_i, D_i, \mathbf{X}_i)\}_{i \in I}$ with $I = \{1, \dots, n\}$ in K folds of approximately equal size. Denote I_k the set of observations included in fold k and $I_k^c = I \setminus I_k$ its complement. Select a set of J base learners with $J \geq 2$.

1. Estimating conditional expectations $\ell_0(\mathbf{X}) = E[Y|\mathbf{X}]$:
 - a. For each k and base learner j , fit the CEF estimator to the sub-sample I_k^c using Y_i as outcome and \mathbf{X}_i as predictors. Obtain the out-of-sample predicted values $\hat{\ell}_{I_k^c}^{(j)}(\mathbf{X}_i)$ for $i \in I_k$.
 - b. Fit a constrained regression of Y_i against $\hat{\ell}_{I_k^c}^{(j)}(\mathbf{X}_i), \dots, \hat{\ell}_{I_k^c}^{(J)}(\mathbf{X}_i)$ over the full sample I . The fitted values are the short-stacking estimates $\hat{\ell}^*(\mathbf{X}_i)$.
2. Estimating conditional expectations $p_0(\mathbf{X}, \mathbf{Z}) = E[D|\mathbf{X}, \mathbf{Z}]$:
 - a. For each k and base learner j , fit the CEF estimator to the sub-sample I_k^c using D_i as outcome and $(\mathbf{X}_i, \mathbf{Z}_i)$ as predictors. Obtain the out-of-sample predicted values $\hat{p}_{I_k^c}^{(j)}(\mathbf{X}_i, \mathbf{Z}_i)$ for $i \in I_k$, and the in-sample predicted values $\tilde{p}_k^{(j)}(\mathbf{X}_i, \mathbf{Z}_i) \equiv \hat{p}_{I_k^c}^{(j)}(\mathbf{X}_i, \mathbf{Z}_i)$ for $i \in I_k^c$.
 - b. For each k , fit a constrained regression of D_i against in-sample predicted values $\tilde{p}_k^{(1)}(\mathbf{X}_i, \mathbf{Z}_i), \dots, \tilde{p}_k^{(J)}(\mathbf{X}_i, \mathbf{Z}_i)$ over the sample I_k^c to obtain the out-of-sample short-stack predicted values $\hat{p}_{I_k^c}^*(\mathbf{X}_i, \mathbf{Z}_i)$ for $i \in I_k$.
 - c. Fit a constrained regression of D_i against $\hat{p}_{I_k^c}^{(1)}(\mathbf{X}_i, \mathbf{Z}_i), \dots, \hat{p}_{I_k^c}^{(J)}(\mathbf{X}_i, \mathbf{Z}_i)$ over the full sample I . The fitted values are the short-stacking estimates $\hat{p}^*(\mathbf{Z}_i, \mathbf{X}_i)$.
3. Estimating conditional expectations $m_0(\mathbf{X}) = E[D|\mathbf{X}]$:
 - a. For each k and base learner j , fit the CEF estimator to the sub-sample I_k^c using in-sample fitted values $\tilde{p}_k^{(j)}(\mathbf{X}_i, \mathbf{Z}_i)$ as the outcome and \mathbf{X}_i as predictors. Obtain the out-of-sample predicted values $\hat{m}_{I_k^c}^{(j)}(\mathbf{X}_i)$ for $i \in I_k$.
 - b. Apply a constrained regression of $\hat{p}_{I_k^c}^*(\mathbf{X}_i, \mathbf{Z}_i)$ against $\hat{m}_{I_k^c}^{(1)}(\mathbf{X}_i), \dots, \hat{m}_{I_k^c}^{(J)}(\mathbf{X}_i)$ over the full sample I . The fitted values are the short-stacking estimates $\hat{m}^*(\mathbf{X}_i)$.
4. Compute

$$\hat{\theta}_n = \frac{\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{\ell}^*(\mathbf{X}_i)) (\hat{p}^*(\mathbf{Z}_i, \mathbf{X}_i) - \hat{m}^*(\mathbf{X}_i))}{\frac{1}{n} \sum_{i=1}^n (D_i - \hat{m}^*(\mathbf{X}_i)) (\hat{p}^*(\mathbf{Z}_i, \mathbf{X}_i) - \hat{m}^*(\mathbf{X}_i))}. \quad (1)$$

□

B Additional simulation results

We briefly summarize the results for DGPs 4 and 5 shown in Table B.1. The stacking weights and MSPEs of the individual learners are reported in Tables B.2 and B.3, respectively.

The bias of DDML with stacking is relatively robust to the inclusion of additional noisy covariates. For $n = 100$, DDML with stacking performs at least as well as feasible full-sample estimators. For $n = 1000$, DDML with stacking outperforms OLS and PDS-Lasso, and exhibits a bias that is only slightly above the infeasible oracle estimator.

Table B.1: Bias and Coverage Rates in the Linear and Nonlinear DGPs

	DGP 4				DGP 5			
	$n = 100$		$n = 1000$		$n = 100$		$n = 1000$	
	MAB	Cov.	MAB	Cov.	MAB	Cov.	MAB	Cov.
Full sample:								
Oracle	0.100	0.928	0.032	0.944	0.100	0.912	0.031	0.956
OLS (Base)	0.127	0.855	0.078	0.613	0.113	0.858	0.077	0.615
PDS-Lasso (Base)	0.119	0.833	0.078	0.611	0.115	0.852	0.077	0.611
PDS-Lasso (Poly 5)	0.128	0.808	0.050	0.843	0.113	0.839	0.041	0.884
PDS-Lasso (Poly 2 + Inter.)	0.132	0.802	0.050	0.828	0.113	0.859	0.036	0.918
DDML methods:								
Base learners								
OLS	0.133	0.678	0.078	0.582	0.117	0.839	0.077	0.608
Lasso with CV (Base)	0.121	0.851	0.081	0.605	0.115	0.853	0.077	0.614
Ridge with CV (Base)	0.161	0.712	0.078	0.583	0.115	0.850	0.077	0.608
Lasso with CV (Poly 5)	0.121	0.852	0.040	0.916	0.125	0.853	0.035	0.928
Ridge with CV (Poly 5)	0.171	0.687	0.104	0.376	0.141	0.795	0.047	0.898
Lasso with CV (Poly 2 + Inter.)	0.130	0.818	0.043	0.900	0.110	0.877	0.036	0.926
Ridge with CV (Poly 2 + Inter.)	0.143	0.736	0.148	0.107	0.133	0.818	0.035	0.914
Random forest (Low)	0.201	0.618	0.117	0.294	0.121	0.842	0.046	0.845
Random forest (Medium)	0.203	0.609	0.141	0.151	0.125	0.838	0.066	0.701
Random forest (High)	0.244	0.477	0.226	0.001	0.174	0.727	0.171	0.038
Gradient boosting (Low)	0.098	0.905	0.033	0.941	0.101	0.899	0.033	0.936
Gradient boosting (Medium)	0.121	0.842	0.049	0.844	0.105	0.888	0.041	0.896
Gradient boosting (High)	0.218	0.552	0.127	0.234	0.171	0.718	0.114	0.311
Neural net	0.094	0.893	0.127	0.186	0.108	0.875	0.034	0.927
Meta learners								
Stacking: CLS	0.114	0.864	0.040	0.915	0.106	0.879	0.035	0.935
Stacking: Single best	0.114	0.875	0.038	0.925	0.107	0.888	0.034	0.940
Short-stacking: CLS	0.114	0.869	0.040	0.914	0.105	0.887	0.033	0.945
Short-stacking: Single best	0.114	0.869	0.039	0.923	0.103	0.898	0.034	0.938

Notes: The table reports median absolute bias (MAB) and coverage rate of a 95% confidence interval (CR). We employ standard errors robust to heteroskedasticity. For comparison, we report the following full sample estimators: infeasible Oracle, OLS, PDS-Lasso with base and two different expanded sets of covariates. DDML estimators use 20 folds for cross-fitting if $n = 100$, and 5 folds if $n = 1000$. Meta-learning approaches rely on all listed base learners. Results are based on 1,000 replications. Results for DGPs 1-3 can be found in Table ?? in the main text.

Table B.2: Stacking weights for the estimation of conditional expectation functions

Observations n	DGP 1		DGP 2		DGP 3		DGP 4		DGP 5	
	100	1000	100	1000	100	1000	100	1000	100	1000
<i>Estimation of $E[Y \mathbf{X}]$:</i>										
OLS	0.027	0.113	0.016	0.	0.034	0.002	0.041	0.017	0.156	0.005
Lasso with CV (Base)	0.182	0.642	0.017	0.	0.206	0.	0.414	0.016	0.120	0.012
Ridge with CV (Base)	0.350	0.041	0.003	0.	0.021	0.	0.033	0.014	0.053	0.018
Lasso with CV (Poly 5)	0.058	0.057	0.205	0.045	0.058	0.	0.149	0.544	0.099	0.272
Ridge with CV (Poly 5)	0.050	0.042	0.121	0.009	0.014	0.	0.023	0.001	0.079	0.050
Lasso with CV (Poly 2 + Inter.)	0.037	0.024	0.222	0.939	0.070	0.008	0.074	0.311	0.092	0.343
Ridge with CV (Poly 2 + Inter.)	0.097	0.011	0.071	0.001	0.039	0.	0.054	0.001	0.043	0.090
Random forest (Low regularization)	0.031	0.004	0.097	0.	0.083	0.064	0.029	0.	0.071	0.054
Random forest (Medium regularization)	0.018	0.	0.066	0.	0.079	0.	0.022	0.	0.041	0.003
Random forest (High regularization)	0.002	0.	0.021	0.	0.018	0.	0.002	0.	0.012	0.
Gradient boosting (Low regularization)	0.055	0.023	0.058	0.002	0.183	0.156	0.077	0.046	0.054	0.033
Gradient boosting (Medium regularization)	0.021	0.013	0.038	0.	0.124	0.687	0.033	0.023	0.021	0.013
Gradient boosting (High regularization)	0.003	0.	0.022	0.	0.025	0.071	0.003	0.	0.006	0.
Neural net	0.069	0.029	0.045	0.004	0.047	0.012	0.045	0.026	0.154	0.105
<i>Estimation of $E[D \mathbf{X}]$:</i>										
OLS	0.029	0.119	0.017	0.	0.034	0.002	0.044	0.016	0.161	0.006
Lasso with CV (Base)	0.171	0.637	0.016	0.	0.182	0.	0.405	0.014	0.123	0.012
Ridge with CV (Base)	0.350	0.034	0.003	0.	0.020	0.	0.031	0.013	0.053	0.018
Lasso with CV (Poly 5)	0.059	0.058	0.218	0.045	0.057	0.	0.151	0.561	0.103	0.272
Ridge with CV (Poly 5)	0.048	0.047	0.116	0.006	0.014	0.	0.026	0.001	0.074	0.049
Lasso with CV (Poly 2 + Inter.)	0.038	0.024	0.237	0.943	0.068	0.007	0.076	0.301	0.099	0.340
Ridge with CV (Poly 2 + Inter.)	0.094	0.011	0.067	0.001	0.039	0.	0.054	0.001	0.044	0.096
Random forest (Low regularization)	0.036	0.005	0.085	0.	0.083	0.047	0.030	0.	0.066	0.055
Random forest (Medium regularization)	0.022	0.	0.063	0.	0.079	0.	0.022	0.	0.036	0.003
Random forest (High regularization)	0.002	0.	0.021	0.	0.024	0.	0.003	0.	0.015	0.
Gradient boosting (Low regularization)	0.055	0.024	0.055	0.001	0.190	0.171	0.079	0.050	0.054	0.034
Gradient boosting (Medium regularization)	0.022	0.012	0.036	0.	0.139	0.731	0.032	0.020	0.022	0.012
Gradient boosting (High regularization)	0.004	0.	0.021	0.	0.024	0.030	0.003	0.	0.006	0.
Neural net	0.069	0.029	0.046	0.003	0.045	0.011	0.045	0.023	0.146	0.103

Notes: This table shows the stacking weights averaged over bootstrap and cross-fitting iterations for each base learner. We report the stacking weights for the estimation of $E[Y|\mathbf{X}]$ and $E[D|\mathbf{X}]$, and for sample sizes of $n = 100$ and $n = 1000$.

Table B.3: Mean-squared prediction error the estimation of conditional expectation functions

Observations n	DGP 1			DGP 2			DGP 3			DGP 4			DGP 5		
	100	1000		100	1000		100	1000		100	1000		100	1000	
<i>Estimation of $E[Y \mathbf{X}]$:</i>															
OLS	2.753	1.372		4.625	2.085		3.292	1.165		3.364	1.766		1.811	1.687	
Lasso with CV (Base)	0.941	1.148		1.900	1.882		0.801	0.909		1.299	1.492		1.572	1.611	
Ridge with CV (Base)	0.760	1.258		1.887	1.875		0.761	0.752		1.147	1.765		1.661	1.687	
Lasso with CV (Poly 5)	1.254	1.030		2.783	2.608		0.988	0.951		1.575	1.575		1.926	1.742	
Ridge with CV (Poly 5)	1.176	0.940		2.445	2.471		0.777	0.849		1.468	1.304		1.591	1.764	
Lasso with CV (Poly 2 + Inter.)	0.545	0.872		2.119	2.614		0.728	0.916		1.096	1.450		1.534	1.677	
Ridge with CV (Poly 2 + Inter.)	3.299	0.787		2.888	2.520		1.392	0.562		2.629	1.247		1.203	1.904	
Random forest (Low regularization)	0.288	0.492		2.016	2.026		0.771	0.985		0.927	1.113		1.327	1.512	
Random forest (Medium regularization)	0.272	0.378		1.991	1.922		0.752	0.816		0.907	0.975		1.290	1.350	
Random forest (High regularization)	0.128	0.110		1.866	1.840		0.567	0.529		0.725	0.698		0.925	0.852	
Gradient boosting (Low regularization)	0.959	1.040		2.458	2.478		1.568	1.440		1.600	1.623		1.806	1.729	
Gradient boosting (Medium regularization)	0.581	0.778		2.118	2.226		1.121	1.260		1.220	1.407		1.392	1.498	
Gradient boosting (High regularization)	0.183	0.317		1.898	1.890		0.699	0.962		0.818	0.995		0.928	1.037	
Neural net	1.614	2.116		2.925	3.469		1.856	2.326		2.035	2.612		1.983	1.944	
<i>Estimation of $E[D \mathbf{X}]$:</i>															
OLS	2.166	1.063		3.801	1.737		2.615	0.925		2.673	1.422		1.458	1.364	
Lasso with CV (Base)	0.733	0.886		1.584	1.571		0.641	0.721		1.043	1.201		1.266	1.303	
Ridge with CV (Base)	0.590	0.950		1.573	1.565		0.609	0.595		0.915	1.342		1.342	1.364	
Lasso with CV (Poly 5)	1.008	0.793		2.274	2.182		0.966	0.751		1.404	1.271		1.497	1.407	
Ridge with CV (Poly 5)	0.953	0.729		2.077	2.061		0.629	0.674		1.220	1.050		1.248	1.423	
Lasso with CV (Poly 2 + Inter.)	0.420	0.671		1.776	2.192		0.586	0.726		0.880	1.170		1.232	1.357	
Ridge with CV (Poly 2 + Inter.)	2.541	0.615		2.415	2.114		1.110	0.443		2.100	1.002		0.967	1.540	
Random forest (Low regularization)	0.221	0.376		1.659	1.680		0.596	0.763		0.725	0.881		1.056	1.209	
Random forest (Medium regularization)	0.210	0.295		1.642	1.600		0.583	0.642		0.712	0.780		1.027	1.087	
Random forest (High regularization)	0.100	0.087		1.549	1.536		0.445	0.420		0.576	0.563		0.742	0.690	
Gradient boosting (Low regularization)	0.732	0.791		2.005	2.041		1.220	1.137		1.250	1.291		1.424	1.378	
Gradient boosting (Medium regularization)	0.440	0.583		1.739	1.841		0.884	0.997		0.951	1.118		1.101	1.191	
Gradient boosting (High regularization)	0.129	0.205		1.573	1.569		0.554	0.733		0.635	0.767		0.723	0.796	
Neural net	1.290	1.673		2.428	2.886		1.491	1.867		1.637	2.111		1.608	1.578	

Notes: This table shows the cross-fitted mean-squared prediction error averaged over bootstrap iterations for the listed conditional expectation function estimators. We report the mean-squared predictor error for the estimation of $E[Y|\mathbf{X}]$ and $E[D|\mathbf{X}]$, and for sample sizes of $n = 100$ and $n = 1000$.

C Applications

Here we continue the 401(k) application from the main text to illustrate estimation of the interactive model and IV models. We use the same data and variables as outlined in the main text. For the IV models, we use eligibility to enroll for the 401(k) pension plan as the instrument and treat participation in a 401(k) as the endogenous variable.

C.1 Interactive Model (interactive)

We allow for heterogenous treatment effects using the **interactive** model. To this end, the conditional expectation of Y given X is fit separately for $D = 1$ and $D = 0$. We also use `reps(5)`. This will execute the `ddml` estimation three times using different random folds. This reduces dependence on a specific fold.

```
. *** initialize
. set seed 123
. ddml init interactive, kfold(5) reps(5)

. *** add learners for E[Y|X,D=0] and E[Y|X,D=1]
. ddml E[Y|X,D]: pystacked $Y $X || ///
> method(ols) || ///
> m(lassocv) xvars(c.($X)##c.($X)) || ///
> m(ridgecv) xvars(c.($X)##c.($X)) || ///
> m(rf) pipe(sparse) opt(max_features(5)) || ///
> m(gradboost) pipe(sparse) opt(n_estimators(250) learning_rate(0.01)) , ///
> njobs(5)
Learner Y1_pystacked added successfully.
. *** add learners for E[D|X]
. ddml E[D|X]: pystacked $D $X || ///
> method(ols) || ///
> m(lassocv) xvars(c.($X)##c.($X)) || ///
> m(ridgecv) xvars(c.($X)##c.($X)) || ///
> m(rf) pipe(sparse) opt(max_features(5)) || ///
> m(gradboost) pipe(sparse) opt(n_estimators(250) learning_rate(0.01)) , ///
> njobs(5)
Learner D1_pystacked added successfully.

. ddml estimate

Model: interactive, crossfit folds k=5, resamples r=5
Mata global (mname): m0
Dependent variable (Y): net_tfa
net_tfa learners: Y1_pystacked
D equations (1): e401
e401 learners: D1_pystacked
DDML estimation results (ATE):
spec r Y(0) learner Y(1) learner D learner b SE
st 1 Y1_pystacked Y1_pystacked D1_pystacked 8026.894 (1126.459)
st 2 Y1_pystacked Y1_pystacked D1_pystacked 7879.717 (1122.815)
st 3 Y1_pystacked Y1_pystacked D1_pystacked 8050.997 (1119.537)
st 4 Y1_pystacked Y1_pystacked D1_pystacked 8157.737 (1113.299)
st 5 Y1_pystacked Y1_pystacked D1_pystacked 7753.948 (1138.377)
```

```

Mean/med Y(0) learner  Y(1) learner      D learner      b      SE
st mn  Y1_pystacked  Y1_pystacked  D1_pystacked  7973.859 (1132.658)
st md  Y1_pystacked  Y1_pystacked  D1_pystacked  8026.894 (1126.459)

Median over 5 stacking resamples (ATE)
E[y|X,D=0] = Y1_pystacked0      Number of obs =      9915
E[y|X,D=1] = Y1_pystacked1
E[D|X]      = D1_pystacked

```

net_tfa	Robust		z	P> z	[95% conf. interval]	
	Coefficient	std. err.				
e401	8026.894	1126.459	7.13	0.000	5819.075	10234.71

```

Stacking final estimator: nnls1
Warning: 5 resamples had propensity scores trimmed to lower limit .01.
Summary over 5 resamples:
      D eqn      mean      min      p25      p50      p75      max
e401  7973.8585  7753.9478  7879.7173  8026.8940  8050.9966  8157.7368

```

One-line syntax (output omitted).

```
. qui qddml $Y $D ($X), model(interactive)
```

C.2 IV model (iv)

```

. use "sipp1991.dta", clear
. global Y net_tfa
. global X age inc educ fsize marr twoearn db pira hown
. global Z e401
. global D p401

```

Step 1: Initialize ddml model.

```

. set seed 123
. ddml init iv

```

Step 2: Add supervised machine learners for estimating conditional expectations.

```

. *** E[Y|X]
. ddml E[Y|X]: pystacked $Y $X || ///
> method(ols) || ///
> m(lassocv) xvars(c.($X)##c.($X)) || ///
> m(ridgecv) xvars(c.($X)##c.($X)) || ///
> m(rf) pipe(sparse) opt(max_features(5)) || ///
> m(gradboost) pipe(sparse) opt(n_estimators(250) learning_rate(0.01)) , ///
> njobs(4)
Learner Y1_pystacked added successfully.

```



```

. *** E[D|X]
. ddml E[D|X]: pystacked $D $X || ///
> method(ols) || ///
> m(lassocv) xvars(c.($X)##c.($X)) || ///
> m(ridgecv) xvars(c.($X)##c.($X)) || ///
> m(rf) pipe(sparse) opt(max_features(5)) || ///
> m(gradboost) pipe(sparse) opt(n_estimators(250) learning_rate(0.01)) , ///
> njobs(4)
Learner D1_pystacked added successfully.

. *** E[Z|X]
. ddml E[Z|X]: pystacked $Z $X || ///
> method(ols) || ///
> m(lassocv) xvars(c.($X)##c.($X)) || ///
> m(ridgecv) xvars(c.($X)##c.($X)) || ///
> m(rf) pipe(sparse) opt(max_features(5)) || ///
> m(gradboost) pipe(sparse) opt(n_estimators(250) learning_rate(0.01)) , ///
> njobs(4)
Learner Z1_pystacked added successfully.

```

Step 3: Perform cross-fitting.

```

. ddml crossfit
Cross-fitting E[y|X] equation: net_tfa
Cross-fitting fold 1 2 3 4 5 ...completed cross-fitting
Cross-fitting E[D|X] equation: p401
Cross-fitting fold 1 2 3 4 5 ...completed cross-fitting
Cross-fitting E[Z|X]: e401
Cross-fitting fold 1 2 3 4 5 ...completed cross-fitting
. ddml extract, show(pystacked)
pystacked weights for Z1_pystacked (e401)

```

	learner	resample	fold_1	fold_2	fold_3	fold_4	fold_5
ols	1	1	0	0	1.577e-17	0	.04870791
lassocv	2	1	0	.36068464	.19641246	.23065829	0
ridgecv	3	1	.50644273	.32367528	.36796975	.30346779	.45676698
rf	4	1	.04990979	.03383132	8.761e-18	.04246409	.02379444
gradboost	5	1	.44364749	.28180876	.43561779	.42340983	.47073066

```

mean stacking weights across folds/resamples for Z1_pystacked (e401)
final stacking estimator: nnls1

```

	learner	mean_weight	rep_1
ols	1	.00974158	.00974158
lassocv	2	.15755108	.15755108
ridgecv	3	.3916645	.3916645
rf	4	.02999993	.02999993
gradboost	5	.4110429	.4110429

```

pystacked MSEs for Z1_pystacked (e401)

```

	learner	resample	fold_1	fold_2	fold_3	fold_4	fold_5
ols	1	1	.19946988	.20137585	.20302906	.20030431	.2013705
lassocv	2	1	.19523544	.19689361	.19807185	.19569792	.19799156
ridgecv	3	1	.19512701	.19728582	.19849531	.19614749	.1981114
rf	4	1	.21307726	.215616	.21841171	.21446185	.21691469
gradboost	5	1	.19529958	.19797459	.19862763	.19615183	.19795431

```

mean stacking MSEs across folds/resamples for Z1_pystacked (e401)
final stacking estimator: nnls1

```

	learner	mean_MSE	rep_1
ols	1	.20110992	.20110992

lassocv	2	.19677808	.19677808					
ridgecv	3	.19703341	.19703341					
rf	4	.2156963	.2156963					
gradboost	5	.19720159	.19720159					

pystacked weights for Y1_pystacked (net_tfa)

	learner	resample	fold_1	fold_2	fold_3	fold_4	fold_5
ols	1	1	.15175107	.01201745	.14501553	.07072362	.03857199
lassocv	2	1	0	.86668516	0	0	.26575544
ridgecv	3	1	.76036209	.10111911	.80863601	.75672014	.61229963
rf	4	1	.0802917	0	.0208563	0	.06587322
gradboost	5	1	0	.01161239	.02847589	.19812117	.01749972

mean stacking weights across folds/resamples for Y1_pystacked (net_tfa)

final stacking estimator: nnls1

	learner	mean_weight	rep_1
ols	1	.08361593	.08361593
lassocv	2	.22648812	.22648812
ridgecv	3	.60782739	.60782739
rf	4	.03340424	.03340424
gradboost	5	.05114183	.05114183

pystacked MSEs for Y1_pystacked (net_tfa)

	learner	resample	fold_1	fold_2	fold_3	fold_4	fold_5
ols	1	1	3.135e+09	3.040e+09	2.674e+09	3.438e+09	3.378e+09
lassocv	2	1	2.958e+09	2.843e+09	2.463e+09	3.107e+09	3.025e+09
ridgecv	3	1	2.952e+09	2.857e+09	2.434e+09	3.099e+09	3.024e+09
rf	4	1	3.344e+09	3.213e+09	2.765e+09	3.541e+09	3.493e+09
gradboost	5	1	3.217e+09	2.990e+09	2.656e+09	3.317e+09	3.344e+09

mean stacking MSEs across folds/resamples for Y1_pystacked (net_tfa)

final stacking estimator: nnls1

	learner	mean_MSE	rep_1
ols	1	3.133e+09	3.133e+09
lassocv	2	2.879e+09	2.879e+09
ridgecv	3	2.873e+09	2.873e+09
rf	4	3.271e+09	3.271e+09
gradboost	5	3.105e+09	3.105e+09

pystacked weights for D1_pystacked (p401)

	learner	resample	fold_1	fold_2	fold_3	fold_4	fold_5
ols	1	1	.05947061	8.397e-18	.06447698	.06941268	.06619485
lassocv	2	1	.05876314	.32796901	.20842013	.39455972	.21484529
ridgecv	3	1	.41441662	.2760156	.36272642	.22474909	.47123129
rf	4	1	.03741096	.05709511	.02323724	.0711886	.08544984
gradboost	5	1	.42993867	.33892028	.34113922	.24008991	.16227873

mean stacking weights across folds/resamples for D1_pystacked (p401)

final stacking estimator: nnls1

	learner	mean_weight	rep_1
ols	1	.05191103	.05191103
lassocv	2	.24091146	.24091146
ridgecv	3	.34982781	.34982781
rf	4	.05487635	.05487635
gradboost	5	.30247336	.30247336

pystacked MSEs for D1_pystacked (p401)

	learner	resample	fold_1	fold_2	fold_3	fold_4	fold_5
ols	1	1	.17102922	.17168453	.17532105	.17229367	.17290828
lassocv	2	1	.16965393	.17005015	.17383906	.17092387	.17131014
ridgecv	3	1	.16988454	.17045619	.17415227	.17155958	.17134841
rf	4	1	.1850616	.1848705	.19065443	.18625053	.1863959
gradboost	5	1	.16981445	.17040097	.17420905	.17155253	.17207733

mean stacking MSEs across folds/resamples for D1_pystacked (p401)

```

final stacking estimator: nnls1
      learner  mean_MSE    rep_1
      ols      1  .17264735 .17264735
      lassocv   2  .17115543 .17115543
      ridgecv   3  .1714802  .1714802
      rf        4  .18664659 .18664659
      gradboost 5  .17161087 .17161087

```

Step 4: Estimate causal effects.

```

. ddml estimate

Model:                iv, crossfit folds k=5, resamples r=1
Mata global (mname):  m0
Dependent variable (Y): net_tfa
net_tfa learners:     Y1_pystacked
D equations (1):       p401
p401 learners:         D1_pystacked
Z equations (1):       e401
e401 learners:         Z1_pystacked

DDML estimation results:
spec  r      Y learner    D learner      b      SE      Z learner
st   1  Y1_pystacked  D1_pystacked 13528.537 (1726.023)  Z1_pystacked

Stacking DDML model
y-E[y|X] = y-Y1_pystacked_1      Number of obs =      9915
D-E[D|X] = D-D1_pystacked_1
Z-E[Z|X] = Z-Z1_pystacked_1

```

net_tfa	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
p401	13528.54	1726.023	7.84	0.000	10145.59	16911.48
_cons	-42.83654	531.1319	-0.08	0.936	-1083.836	998.1628

```

Stacking final estimator: nnls1

```

One-line syntax.

```

. qui qddml $Y ($X) ($D = $Z), model(iv)

```

C.3 Interactive IV Model interactiveiv

```

. use "sipp1991.dta", clear
. global Y net_tfa
. global X age inc educ fsize marr twoearn db pira hown
. global Z e401
. global D p401

```

Step 1: Initialize ddml model.

```
. set seed 123
. ddml init interactiveiv
```

Step 2: Add supervised machine learners for estimating conditional expectations.

```
. * add learners for E[Y|X,Z=0] and E[Y|X,Z=1]
. ddml E[Y|X,Z]: pystacked $Y $X || ///
> method(ols) || ///
> m(lassocv) xvars(c.($X)##c.($X)) || ///
> m(ridgecv) xvars(c.($X)##c.($X)) || ///
> m(rf) pipe(sparse) opt(max_features(5)) || ///
> m(gradboost) pipe(sparse) opt(n_estimators(250) learning_rate(0.01)) , ///
> njobs(4)
Learner Y1_pystacked added successfully.
. * add learners for E[D|X,Z=0] and E[D|X,Z=1]
. ddml E[D|X,Z]: pystacked $D $X || ///
> method(ols) || ///
> m(lassocv) xvars(c.($X)##c.($X)) || ///
> m(ridgecv) xvars(c.($X)##c.($X)) || ///
> m(rf) pipe(sparse) opt(max_features(5)) || ///
> m(gradboost) pipe(sparse) opt(n_estimators(250) learning_rate(0.01)) , ///
> njobs(4)
Learner D1_pystacked added successfully.
. * add learners for E[Z|X]
. ddml E[Z|X]: pystacked $Z $X || ///
> method(ols) || ///
> m(lassocv) xvars(c.($X)##c.($X)) || ///
> m(ridgecv) xvars(c.($X)##c.($X)) || ///
> m(rf) pipe(sparse) opt(max_features(5)) || ///
> m(gradboost) pipe(sparse) opt(n_estimators(250) learning_rate(0.01)) , ///
> njobs(4)
Learner Z1_pystacked added successfully.
```

Step 3: Perform cross-fitting.

```
. ddml crossfit
Cross-fitting E[y|X,Z] equation: net_tfa
Cross-fitting fold 1 2 3 4 5 ...completed cross-fitting
Cross-fitting E[D|X,Z] equation: p401
Cross-fitting fold 1 2 3 4 5 ...completed cross-fitting
Cross-fitting E[Z|X]: e401
Cross-fitting fold 1 2 3 4 5 ...completed cross-fitting
```

Step 4: Estimate causal effects.

```
. ddml estimate

Model: interactiveiv, crossfit folds k=5, resamples r=1
Mata global (mname): m0
Dependent variable (Y): net_tfa
net_tfa learners: Y1_pystacked
D equations (1): p401
p401 learners: D1_pystacked
```

```
Z equations (1):          e401
e401 learners:           Z1_pystacked
DDML estimation results (LATE):
spec  r  Y(0) learner  Y(1) learner  D(0) learner  D(1) learner      b      SE      Z learner
st  1  Y1_pystacked  Y1_pystacked  D1_pystacked  D1_pystacked  11579.166 (1613.058)  Z1_pystacked

Stacking DDML model (LATE)
E[y|X,Z=0] = Y1_pystacked0_1
E[y|X,Z=1] = Y1_pystacked1_1
E[D|X,Z=0] = D1_pystacked0_1
E[D|X,Z=1] = D1_pystacked1_1
E[Z|X]     = Z1_pystacked_1
Number of obs = 9915
```

net_tfa	Robust					[95% conf. interval]
	Coefficient	std. err.	z	P> z		
p401	11579.17	1613.058	7.18	0.000	8417.631	14740.7

```
Stacking final estimator: nnls1
Warning: 13 propensity scores trimmed to lower limit .01.
```

One-line syntax.

```
. qui qddml $Y ($X) ($D=$Z), model(interactiveiv)
```