Assignment 2 - Machine Learning for Economic Analysis

By Flurina Schneider (14-319-644), Lukas Augustin (13-560-487), Yuning Yu (20-745-691), Amir Hossein Nadjafabadi Farahani (20-718-912)

Over the past decade, many countries of the Western hemisphere were repeatedly shaken by terror attacks. Consequently, terrorism has emerged to be one of the essential topics in political and public policy discussions of recent years. In response, a small economics literature has started to exploit the events of terror attacks to study the relationship between police presence and crime prevalence. In our project, we will apply Double/Debiased Machine Learning (DML) as introduced by Chernozhukov et al. (2018) to estimate a partially linear IV model for aiming to obtain an unbiased estimate for the effect of police presence on crime. To do so, we will replicate the main findings from the paper *Panic on the Streets of London: Police, Crime and the July 2005 Terror Attacks* by Draca, Machin and Witt (2011) who have studied the crime-police relationship in the light of the 2005 London terror attacks using an IV strategy. Our project will be organised as follows. Section I provides an overview on Draca, Machin and Witt (2011)'s context and the data used before we will briefly outline our methods in section II. Finally, we will discuss our findings in section III and conclude in section IV.

I. Data

On July 7th 2005, at 08:50 local time, the first bomb detonated in London's Underground near Liverpool Street. Three more bombs detonated near the stations Russell Square, Edgware Road, and Tavistock Square in the following minutes. Those four bombs caused 56 fatal victims and more than 700 injured. Two weeks later on July 21st, four additional but this time unsuccessful attempts at detonating bombs in London's public transportation happened. Those attacks caused a temporary sharp discontinuity in police deployment within London over the following weeks.

This project aims to exploit the increase in police deployment in the aftermath of the attacks as a natural experiment to identify the causal relationship between police presence and crime. The attacks thereby caused an independent and unanticipated increase in police presence. The primary data to investigate that effect is provided by the Office of National Statistics (ONS) and the London Metropolitan Police Service (LMPS). Additional data on weekly tube journeys and local labor market conditions are obtained from Transport for London (TFL) and the UK Labour Force Survey (LFS).

The data covers the total number of crimes per 1,000 population and total police hours worked per 1,000 population on a weekly basis from January 1st, 2004, to December 31st, 2005. They are both aggregated on the borough level for the 32 London boroughs. Data on police hours worked was produced under confidential data-sharing with the LMPS. The dataset consists of 3,328 observations (32 boroughs multiplied by 104 weeks). (Draca, Machin and Witt (2011))

II. Methods

To investigate the effect of police presence on crime in this context, it is crucial to establish the exclusion restriction that the terror attacks are related to crimes only through the attack-related police presence increase and not through any other channel. The data supports this exclusion restriction since it shows that police presence was only increased in London's inner boroughs and that crime prevalence only decreased in those same boroughs. As the police presence was discretely reduced from the inner boroughs after six weeks, the data shows an increase in crime that is timed precisely to this removal. Therefore, it can be argued that the terror attacks are only related to crime through the police presence and not through any other channel. If there were another channel, crime would not have increased with the police presence's discrete removal.

According to Belloni et al. (2015), instrumental variable techniques often result in imprecise inference. One reason for this is that the structural equation model might actually only be partially linear and hence classic OLS fails to recover the true coefficient on an endogenous variable. Using DML allows us to relax the above stated exclusion restriction and replace it by a weaker assumption that all other channels affecting crime can be sufficiently controlled for by an unknown function of all the control variables which can be learned by Machine Learning methods. (Chernozhukov et al. (2018)) In the following we will introduce both the original estimation approach under II.A as well as a proposed application of DML to improve inference in II.B.

A. Econometric Model Suggested by Draca, Machin and Witt (2011)

To be able to perform an instrumental variable regression, one first needs to come up with an instrument. The authors therefore defined London's five inner boroughs Camden, Kensington, Chelsea, Islington, Tower Hamlets and Westminster as the treatment group within the first six weeks after the attacks. Hence, the instrument is denoted as an interaction term between the treatment boroughs and a dummy variable equal to one in the post-attack period. The first stage of the 2SLS is a regression of the total police hours worked per 1,000 population on this interaction term along with multiple controls. The controls contain borough unemployment rate, employment rate, males under 25 as proportion of population and whites as proportion of population. In specific, the reduced form equation can be written as follows:

$$\Delta_{52Police_{bt}} = \alpha_0 + \alpha_1 POST_t + \alpha_2(T_b \times POST_t) + \alpha_3\Delta_{52Xbt} + \Delta_{52}\epsilon_{bt} \tag{1}$$

The second stage is a regression of the total number of crimes per 1,000 population on the estimated police hours worked from the first stage. The structural equation thus is denoted as:

$$\Delta_{52Crime_{bt}} = \beta_0 + \beta_1 POST_t + \beta_2 Police_{bt} + \beta_3\Delta_{52Xbt} + \Delta_{52}\epsilon_{bt} \tag{2}$$

where the variation in police deployment induced by the terror attacks identifies the causal impact of police on crime (β_2). Due to the high seasonality of crime, (1) and (2) are differenced across weeks of the year. Nevertheless, as discussed above those estimated results might be imprecise due to potentially non-linear relationships. To account for that, we further explore the DML method as proposed by Chernozhukov et al. (2018).

B. *Double/Debiased Machine Learning*

Since the goal of this analysis is to derive policy-relevant recommendations, it is crucial that we can derive valid confidence statements from estimated coefficient of police deployment. Using ML methods to estimate the above mentioned non-linearities in our first stage regression (1) seems tempting but might in fact induce a heavy bias in estimators of β_2 when applied naively. Chernozhukov et al. (2018) hence propose the following two ingredients to make the estimation of our treatment effect of interest $\sqrt{N}$ - consistent:

1) Using Neyman-orthogonal moments/scores that have reduced sensitivity with respect to nuisance parameters to estimate β_2
2) Making use of cross-fitting which provides an efficient form of data-splitting

As we are facing a semi-parametric problem of inference in a high-dimensional case, we are convinced that DML allows us to retrieve an approximately unbiased point estimator of the true parameter value. The package *DoubleML* provides a convenient implementation of the method for both R and Python, which we have used to compute our estimates. (Bach and Spindler (2020) We thus rewrite the above introduced IV regression model as a partially linear IV regression model:

$$\Delta_{52Crime_{bt}} = \delta_0 Police_{bt} + \gamma_0(X_{bt}) + u \tag{3}$$

$$\Delta_{52Police_{bt}} = \theta_0(X_{bt}, POST_t, (T_b \times POST_t)) + u \tag{4}$$

and follow the DML procedure:

1) Split the sample in five samples: 1 main sample and 4 auxiliary samples

2) Use auxiliary samples to estimate $\gamma_0(X_{bt})$ from equation (3)

3) Use auxiliary samples to estimate $\theta_0(X_{bt}, POST_t, (T_b \times POST_t))$ from equation (4) where $POST_t$ and $(T_b \times POST_t)$ are our instruments for police deployment

4) Use the main sample to compute the orthogonalized component of $\Delta_{52Police_{bt}}$ on our instruments and first-stage controls as:

$$\hat{v} = \Delta_{52Police_{bt}} - \theta_0(X_{bt}, POST_t, (T_b \times POST_t)) \tag{5}$$

5) Use the main sample to estimate the double-residualized IV-estimator as

$$\hat{\delta_0} = (\frac{1}{n}\sum_{n\in N} \hat{v}x\Delta_{52Police_{bt}})^{-1}\frac{1}{n}\sum_{n\in N} \hat{v}x(\Delta_{52Crime_{bt}} - \gamma_0(X_{bt})) \tag{6}$$

For estimating the non-parametric nuisance parameters we have applied several Machine Learning methods whose specifications are briefly described in the following. All of the following methods were applied with splitting the data into 5 folds whereof 1 was used for testing. Moreover, we have repeated cross-fitting 5 times. The code for replicating our findings can be found in the Appendix V.

Random Forest

We fit a random forest with the following parameters:

- Number of trees: 100
- Number of variables to split at each node: $\sqrt{N}$
- Minimum Node Size: 2
- Max. Depth: 5

All other parameters were set to the default of the function (ranger).

Regression Tree

All parameters were set to the default of the function (rpart).

LASSO

We fit LASSO with the following parameters:

- Lambda: 0.05

The optimal lambda was determined with a random cross-validation grid search. All other parameters were set to the default of the function (glmnet).

Boosting

We applied extreme gradient boosting with the following parameters:

- Eta: 0.3
- Max. Depth: 5

All other parameters were set to the default of the function (xgboost).

III. Findings

Table 1 reports results from an application of DML with 5-fold cross-fitting. We find - with the exception of Gradient Boosting - negative and statistically significant point estimates at the 1%-level. We believe that there potentially was an issue with the parametrization of the gradient boosting algorithm and will hence abstract from its results in the following discussion of our results. In comparison with the baseline IV estimate reported in Draca, Machin and Witt (2011), we find an even more negative effect of police deployment on crime. Qualitatively however, our estimates are similar and indicating an average elasticity of crime with respect to police of around -0.25. This implies that a 10 percent increase in police activity reduces crime by around 2.5 percent, depending on the Machine Learning algorithm used for estimating the nuisance functions. In terms of standard errors, LASSO seems to yield a point estimate with the lowest variability out of all methods we presented.

Table 1—Estimated Effect of Police Deployment on Crime

	Coefficient	Standard-Error	t-value	p-value	95%-CI Lower	95%-CI Upper
IV	-0.178	0.062	-2.890	0.004	-0.298	-0.057
Random Forest	-0.248	0.060	-4.125	0.00004	-0.366	-0.130
Regression Tree	-0.211	0.064	-2.738	0.006	-0.311	-0.052
LASSO	-0.247	0.054	-4.551	0.00001	-0.356	-0.142
Gradient Boosting	1.063	0.009	117.219	0	1.045	1.080

This table shows DML estimates of the coefficient of police deployment on crime. For each method, we split the data into 5 folds and repeated the procedure 5 times. Own calculations based on Draca, Machin and Witt (2011).

Finally, as in Chernozhukov et al. (2018), the choice of the ML method used in estimating nuisance functions does not substantively change the conclusion of our estimates.

IV. Conclusion

Table 1 provides strong evidence for a negative impact of police on crime. It shows that a 10& increase in police presence is associated with a reduction in crime of up to -2.5%. To obtain an unbiased estimate of police presence on crime, we replicated the parametric instrumental variable regression presented in Draca, Machin and Witt (2011). Furthermore, we applied a double debiased machine learning approach to introduce a semi-parametric model that allows for controlling for non-linearities We thereby took advantage of the advance machine learning methods random forest, regression tree, LASSO and gradient boosting. The double debiased machine learning approach provides qualitatively similar results compared to the original findings of the paper. However, except for gradient boosting, we obtained more negative effects of police presence on crime than the baseline instrumental variable approach. Therefore, we conclude that the double machine learning approach was able to unveil relevant non-linearities that remained undetected in the parametric instrumental variable regression. Finally, given the flexible nature of machine learning methods, hyperparameter-tuning could lead to even more precise results and interesting future research projects.

REFERENCES

Bach, P., Chernozhukov V. Kurz M. S., and M. Spindler. 2020. "DoubleML - Double Machine Learning in R." URL: https://github.com/DoubleML/doubleml-for-r, R-Package version 0.1.0.

Belloni, Alexandre, Daniel Chen, Victor Chernozhukov, and Christian Hansen. 2015. "Sparse Models and Methods for Optimal Instruments with an Application to Eminent Domain."

Chernozhukov, Victor, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. 2018. "Double/debiased machine learning for treatment and structural parameters." *The Econometrics Journal*, 21(1): C1–C68. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/ectj.12097.

Draca, Mirko, Stephen Machin, and Robert Witt. 2011. "Panic on the Streets of London: Police, Crime, and the July 2005 Terror Attacks." *American Economic Review*, 101(5): 2157–2181.

V. Appendix

```
library(DoubleML)
library(mlr3)
library(mlr3learners)
library(paradox)
library(data.table)
library(stargazer)
library(ranger)
library(magicfor)
library(haven)
library(ivreg)
library(magrittr)
library(mlr3tuning)
lgr::get_logger("mlr3")$set_threshold("warn")
setwd("~/Dropbox/HS2020/Machine Learning for Economic Analysis/Assignments/Assignment 2")
assignment2 <- read_dta("assignment2.dta")
######################################################
#Define Variables
data = na.omit(assignment2)
z_names = names(data)[grepl("Z", names(data))]
x_names = names(data)[grepl("X", names(data))]
x_iv    = dput(grep("X", names(data), value = TRUE))
d_names = names(data)[grepl("dlhpop", names(data))]
#results matrix
result <- data.frame(matrix(nrow = 5, ncol = 6))
colnames(result) <- c("Coefficient", "Standard-Error","t-value", "p-value",
                      "95%-CI Lower", "95%-CI Upper")
```

```
rownames(result) <- c("IV","regr.ranger", "regr.rpart","regr.glmnet", "regr.xgboost")
#######################################################
#1.Benchmark model: IV Regression
#######################################################
xnam <- paste("X", 1:140, sep="")
fmla_ss <- as.formula(paste("y ~ dlhpop +", paste(xnam, collapse= "+")))
iv2 = ivreg(y ~ dlhpop +X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9 +
              X10 + X11 + X12 + X13 + X14 + X15 + X16 + X17 + X18 + X19 +
              X20 + X21 + X22 + X23 + X24 + X25 + X26 + X27 + X28 + X29 +
              X30 + X31 + X32 + X33 + X34 + X35 + X36 + X37 + X38 + X39 +
              X40 + X41 + X42 + X43 + X44 + X45 + X46 + X47 + X48 + X49 +
              X50 + X51 + X52 + X53 + X54 + X55 + X56 + X57 + X58 + X59 +
              X60 + X61 + X62 + X63 + X64 + X65 + X66 + X67 + X68 + X69 +
              X70 + X71 + X72 + X73 + X74 + X75 + X76 + X77 + X78 + X79 +
              X80 + X81 + X82 + X83 + X84 + X85 + X86 + X87 + X88 + X89 +
              X90 + X91 + X92 + X93 + X94 + X95 + X96 + X97 + X98 + X99 +
              X100 + X101 + X102 + X103 + X104 + X105 + X106 + X107 + X108 +
              X109 + X110 + X111 + X112 + X113 + X114 + X115 + X116 + X117 +
              X118 + X119 + X120 + X121 + X122 + X123 + X124 + X125 + X126 +
              X127 + X128 + X129 + X130 + X131 + X132 + X133 + X134 + X135 +
              X136 + X137 + X138 + X139 + X140 | Z1 + Z2 + X1 + X2 + X3 + X4
              + X5 + X6 + X7 + X8 + X9 +
              X10 + X11 + X12 + X13 + X14 + X15 + X16 + X17 + X18 + X19 +
              X20 + X21 + X22 + X23 + X24 + X25 + X26 + X27 + X28 + X29 +
              X30 + X31 + X32 + X33 + X34 + X35 + X36 + X37 + X38 + X39 +
              X40 + X41 + X42 + X43 + X44 + X45 + X46 + X47 + X48 + X49 +
              X50 + X51 + X52 + X53 + X54 + X55 + X56 + X57 + X58 + X59 +
              X60 + X61 + X62 + X63 + X64 + X65 + X66 + X67 + X68 + X69 +
              X70 + X71 + X72 + X73 + X74 + X75 + X76 + X77 + X78 + X79 +
              X80 + X81 + X82 + X83 + X84 + X85 + X86 + X87 + X88 + X89 +
              X90 + X91 + X92 + X93 + X94 + X95 + X96 + X97 + X98 + X99 +
              X100 + X101 + X102 + X103 + X104 + X105 + X106 + X107 + X108 +
              X109 + X110 + X111 + X112 + X113 + X114 + X115 + X116 + X117 +
              X118 + X119 + X120 + X121 + X122 + X123 + X124 + X125 + X126 +
              X127 + X128 + X129 + X130 + X131 + X132 + X133 + X134 + X135 +
              X136 + X137 + X138 + X139 + X140,
            data = data)
summary(iv2)

result[1,1] = iv2$coefficients[[2]]
result[1,2] = 0.0615307
result[1,3] = -2.890
result[1,4] =  0.003911
result[1,5] = iv2$coefficients[[2]] - 1.96 *0.0615307
result[1,6] = iv2$coefficients[[2]] + 1.96 *0.0615307
#######################################################
#2.Apply DML and loop over different algorithms
#######################################################
#random forest, regression tree, elastic net (lasso), gradient boosting
learners <- c("regr.ranger", "regr.rpart","regr.glmnet", "regr.xgboost" )
#loop over different algorithms to estimate nuisance parameters
```

```
for (i in learners) {

  if (i=="regr.ranger") {
    learner = lrn(i, num.trees=100, min.node.size = 2, max.depth = 5)
    ml_g = learner$clone()
    ml_m = learner$clone()
    ml_r = learner$clone()
    #fit model
    set.seed(2222)
    obj_dml_data = double_ml_data_from_data_frame(data, y_col="y",
                    d_col = d_names , z_cols=z_names, use_other_treat_as_covariate = TRUE)
    dml_pliv_obj = DoubleMLPLIV$new(obj_dml_data, ml_g, ml_m, ml_r,
                    n_folds = 5, n_rep = 5, dml_procedure = "dml2", draw_sample_splitting = T)
    dml_pliv_obj$fit()

    result[i,1] = dml_pliv_obj$all_coef[1]
    result[i,2] = dml_pliv_obj$all_se[[1]]
    result[i,3] = dml_pliv_obj$t_stat[[1]]
    result[i,4] = dml_pliv_obj$pval[[1]]
    result[i,5] = dml_pliv_obj$confint()[1]
    result[i,6] = dml_pliv_obj$confint()[2]
  } else if (i=="regr.rpart") {
    learner = lrn(i)
    ml_g = learner$clone()
    ml_m = learner$clone()
    ml_r = learner$clone()

    #fit model
    set.seed(2222)
    obj_dml_data = double_ml_data_from_data_frame(data, y_col="y",
                    d_col = d_names , z_cols=z_names, use_other_treat_as_covariate = TRUE)
    dml_pliv_obj = DoubleMLPLIV$new(obj_dml_data, ml_g, ml_m, ml_r,
                    n_folds = 5, n_rep = 5, dml_procedure = "dml2", draw_sample_splitting = T)
    dml_pliv_obj$fit()

    result[i,1] = dml_pliv_obj$all_coef[1]
    result[i,2] = dml_pliv_obj$all_se[[1]]
    result[i,3] = dml_pliv_obj$t_stat[[1]]
    result[i,4] = dml_pliv_obj$pval[[1]]
    result[i,5] = dml_pliv_obj$confint()[1]
    result[i,6] = dml_pliv_obj$confint()[2]
  } else if (i=="regr.glmnet") {
    learner = lrn(i)
    ml_g = learner$clone()
    ml_m = learner$clone()
    ml_r = learner$clone()

    set.seed(2222)
    obj_dml_data = double_ml_data_from_data_frame(data, y_col="y",
                    d_col = d_names , z_cols=z_names, use_other_treat_as_covariate = TRUE)
    dml_pliv_obj = DoubleMLPLIV$new(obj_dml_data, ml_g, ml_m, ml_r,
                   n_folds = 5, n_rep = 5, dml_procedure = "dml2", draw_sample_splitting = T)
    #hyperparameter tuning for LASSO (lambda)
    par_grids = list("ml_g" = ParamSet$new(list(
      ParamDbl$new("lambda", lower = 0.05, upper = 0.1))),
      "ml_m" =  ParamSet$new(list(
        ParamDbl$new("lambda", lower = 0.05, upper = 0.1))),
      "ml_r" = ParamSet$new(list(
      ParamDbl$new("lambda", lower = 0.05, upper = 0.1))))

    tune_settings = list(n_folds_tune = 5,
                         rsmp_tune = "cv",
                         measure = list("ml_g" = "regr.mse",
                                         "ml_m" = "regr.mse",
                                         "ml_r" = "regr.mse"),
                         terminator = mlr3tuning::trm("evals", n_evals = 100),
                         algorithm = "grid_search",
                         resolution = 10)
    dml_pliv_obj$tune(param_set=par_grids,
                      tune_settings=tune_settings, tune_on_fold=FALSE)
    dml_pliv_obj$params

    #fit model
    dml_pliv_obj$fit()

    result[i,1] = dml_pliv_obj$all_coef[1]
    result[i,2] = dml_pliv_obj$all_se[[1]]
    result[i,3] = dml_pliv_obj$t_stat[[1]]
    result[i,4] = dml_pliv_obj$pval[[1]]
    result[i,5] = dml_pliv_obj$confint()[1]
    result[i,6] = dml_pliv_obj$confint()[2]
  } else if (i=="regr.xgboost") {
    learner = lrn(i,eta=0.3, verbose=0, objective="reg:squarederror", max_depth=5)
    ml_g = learner$clone()
    ml_m = learner$clone()
    ml_r = learner$clone()
    set.seed(2222)
    obj_dml_data = double_ml_data_from_data_frame(data, y_col="y",
                   d_col = d_names , z_cols=z_names, use_other_treat_as_covariate = TRUE)
    dml_pliv_obj = DoubleMLPLIV$new(obj_dml_data, ml_g, ml_m, ml_r,
                   n_folds = 5, n_rep = 5, dml_procedure = "dml2", draw_sample_splitting = T)
    #fit model
    dml_pliv_obj$fit()
    result[i,1] = dml_pliv_obj$all_coef[1]
    result[i,2] = dml_pliv_obj$all_se[[1]]
    result[i,3] = dml_pliv_obj$t_stat[[1]]
    result[i,4] = dml_pliv_obj$pval[[1]]
    result[i,5] = dml_pliv_obj$confint()[1]
    result[i,6] = dml_pliv_obj$confint()[2]
  }

}
```

```
######################################################
# 3.Export Results to LaTex
######################################################

# use the stargazer package to export result data frame
rownames(result) <- c("IV","Random Forest",
                      "Regression Tree","LASSO", "Gradient Boosting")

stargazer(result, summary=FALSE, rownames=T, out=c("results_assignment2.tex"))
```