**Causal Machine Learning: Beyond Predictive Models and A/B Testing**

In almost all data science jobs, you’ll often find yourself solving problems using machine learning to predict outcomes based on variables. For instance, you might use features to predict whether customers will churn. But from a business perspective, it’s not enough to just know *if* customers will churn — we want to know *why* they churn and, more importantly, how we can prevent it. This is where causal machine learning (causal ML) comes into play. Unlike traditional predictive modeling, which focuses on forecasting outcomes, causal ML is concerned with estimating the causal effect of a treatment T on an outcome Y.

You might wonder, “Isn’t this just A/B testing?” Well, yes and no. A/B testing is a form of causal inference where we randomly assign treatments (e.g., personalized ads) to users and measure the effect on some outcome (e.g., user engagement). But A/B tests aren’t always feasible. For instance, if you want to understand how delivery speed impacts customer lifetime value, running an A/B test could negatively impact business operations and customer satisfaction. Users in the control group would receive intentionally slower deliveries, which might reduce satisfaction, retention, and overall platform performance. In cases like this, causal ML provides an alternative.

At this point, you may think of using a linear regression model to estimate the treatment effect. For instance, if we want to see how personalized ads T affect user engagement Y, we could write a simple regression equation like this:

*Y=βT+γX+ϵ*

Where:

* Y is user engagement
* β is the coefficient we care about, which represents the effect of personalized ads on engagement
* X is a vector of control variables, such as user age, browsing history, and other covariates
* ϵ is the error term

In theory, this looks straightforward. However, there are limitations.

* First, are we sure the relationship between Y and T, or between Y and X, is **linear**? Real-world data is often more complex than a linear relationship can capture, potentially leading to biased estimates of 𝛽.
* Second, we need to ensure that important**confounding variables are included in X**. If key confounders are omitted, **β may capture the effect of those missing variables, rather than the true causal effect of T**.

Given these challenges, both A/B testing and linear regression have limitations. The solution? **Double Machine Learning.**

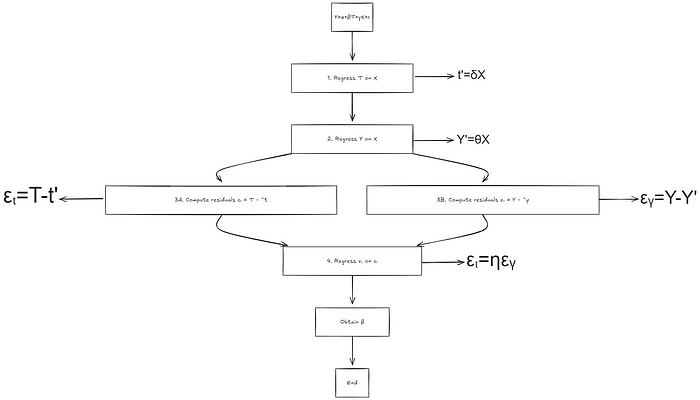
**Double Machine Learning**

Double Machine Learning (DML), also known as de-biased machine learning, was first proposed by MIT statistician and economist Victor Chernozhukov(Chernozhukov et al., 2018). The goal of DML is to develop a:

* Causal estimator that can leverage the flexibility of non-parametric machine learning models
* Reducing bias
* Providing valid confidence intervals
* “Root-n-consistent” estimator, meaning the estimation error approaches zero at a rate of 1 /√n when the sample size (n) goes to infinity.

Let’s revisit our example of estimating the effect of personalized ads T on user engagement Y .In this case, we also have various control variables X, such as user age or browsing history, that could influence both the treatment and the outcome.

Before we dive into how DML addresses this problem, it’s essential to introduce one of its key foundations: the **Frisch-Waugh-Lovell (FWL) theorem**. The FWL theorem is a simple procedure for isolating the effect of T by following these steps:



**Frisch-Waugh-Lovell Procedure**

At the end of this procedure your**true causal effect β is equal to η.**

While this procedure still uses linear regression to model the causal effect, the interesting part is that steps 1 and 2 extract the variation in Tand Y that is independent of X meaning the **residuals are free from the influence of the confounders.**

What Double ML does is enhance this process by allowing you to replace the linear regression in steps 1 and 2 with any machine learning model (e.g., XGBoost, LightGBM). This modification is called **orthogonalization**. The strength **orthogonalization** lies in its ability to capture complex, non-linear relationships in the data. The general equation can be written as:

*Y=βT+f(X)+ϵ ; T=g(X)+ϵ’*

*f(.) and g(.) can be can be any non-linear functions of X*

By using machine learning instead of traditional linear regression, we can more effectively separate the estimation of the causal parameter β from other factors that might confuse our results. In causal inference, we call these confusing factors**“nuisance parameters.”**

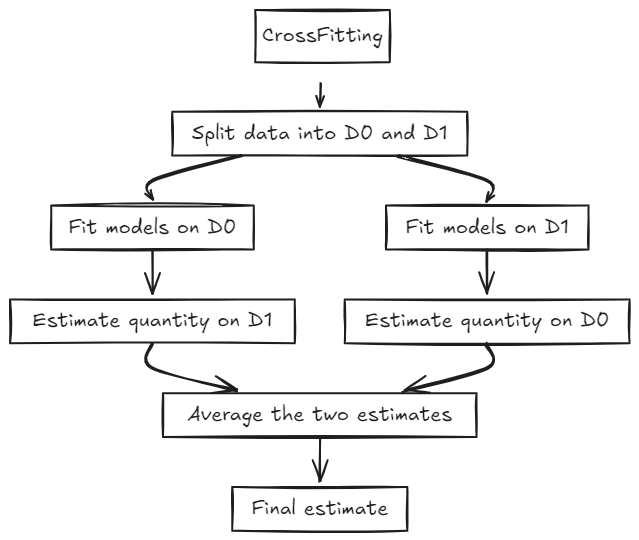
It all sounds straightforward so far, but let’s dive into how Double ML helps reduce bias and the types of biases we’re dealing with.

There are two main types of biases we need to tackle:

* **Regularization Bias**: Regularization methods (e.g., lasso, ridge) can introduce bias into the treatment effect estimates by shrinking coefficients, which may distort the treatment-outcome relationship.
* **Overfitting bias**: This might sound counterintuitive, but when a model overfits, it has low bias but high variance. Essentially, it performs very well on the data it was trained on but poorly on new, unseen data. Overfitting bias occurs when a model captures spurious correlations this can result in high variance and biased causal effect estimates

To address these biases, we apply **orthogonalization for the first one** and **CrossFitting for the second one**.

CrossFitting follow this procedure:



CrossFitting

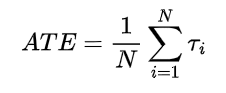
This procedure looks like cross-validation and its right, but the two serves different purpose, CrossFitting helps correct bias, while cross-validation is used to evaluate how well a model performs.

With Double Machine Learning, we overcome many common problems, achieving an estimator with low variance and bias and providing valid confidence intervals.

**DML in practice**

To get our hands dirty, let’s work on a classic example: estimating the causal effect of a training program on earnings.

How do we do this? Basically, we want to find the average difference in earnings between people who received the training and those who didn’t, while taking other factors into account. This difference is known as the **Average Treatment Effect (ATE)**. We can write it like this:



Average treatment effect

https://miro.medium.com/v2/resize:fit:183/1*ib5kPfLWgtJTcqpAe6PS2A.png

Y\_i: in the formula stands for counterfactual outcomes for the unit i:

Yi\_1: is the outcome under treatment

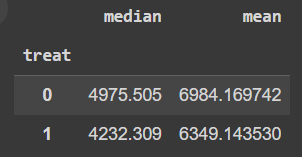
Yi\_0: is the outcome under no treatment.

Since we can’t see both outcomes for the same person, we estimate them.

In our case, we’ll use the**Lalonde dataset**, which includes information on individuals who participated in a job training program and those who did not. The dataset provides several covariates such as age, education, race, and marital status. Our goal is to apply Double Machine Learning (DML) to estimate the ATE of the training program on earnings.

**1- Data Visualization:**

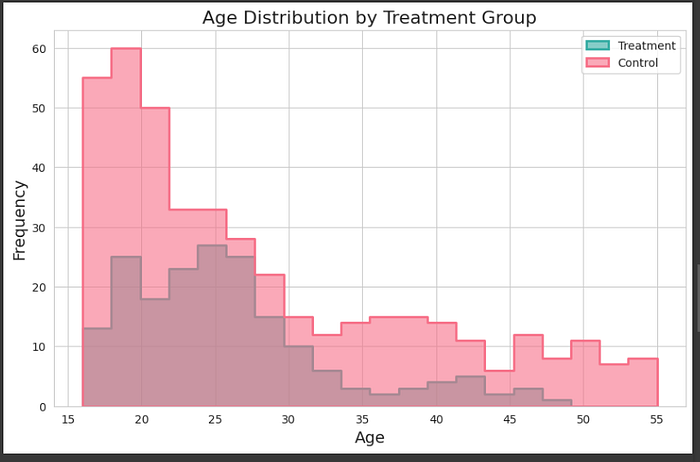
First, let’s take a look at the data. We want to compare the outcomes between the control group and the treatment group. It’s important to examine the characteristics of each group to see if they are similar or if there are noticeable differences.



difference in outcome between control 0 and treatment 1

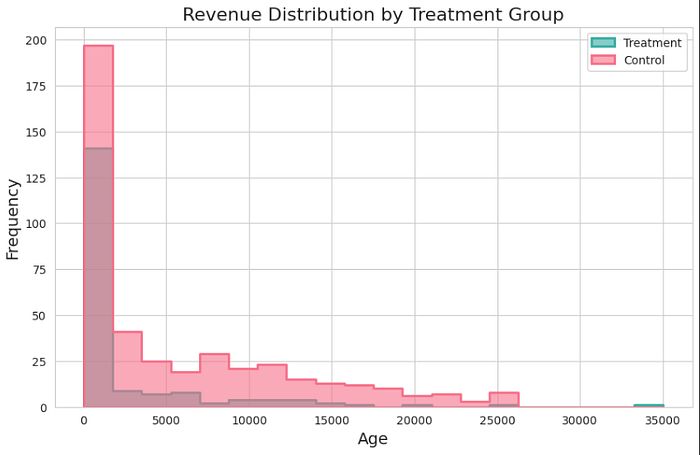
At first glance, the control group might seem to have higher earnings than the treatment group, which could suggest a negative effect of the job training. However, to accurately estimate the causal effect, we need to ensure that the two groups are similar in terms of other factors — this is known as checking for **covariate balance.**Covariate balance it ensures that the groups are comparable and that any observed differences are due to the training itself.

Let’s check the balance between the groups



When we plot the age distribution between the groups, for example, we see a clear **difference in age** between the two groups, which could be influencing the earnings outcome

Next, let’s examine the revenue distributions.



Looking at the revenue, it seems that the treatment assignment might not be independent of each subject’s potential outcomes. For instance, poorer individuals are more represented in the treatment group than in the control group. This suggests that covariate balance has not been achieved. In other words, the observed differences between the treatment and control groups might not be solely due to the treatment itself.

In summary, the groups are not balanced, meaning the differences we see might not be just from the treatment, maybe age or other covariates. To fix this, we use **matching techniques to make sure the groups are similar**

**2- Data Preprocessing**

First, we need to ensure that our treatment and control groups are comparable so that any differences in outcomes are genuinely due to the treatment itself, not other factors. One way to solve the covariate balance issue is by using matching techniques. These methods help us make the groups more similar by controlling for factors like age, education, or prior income that could otherwise bias our estimate of the treatment effect.

* **Propensity Score Matching:** This technique estimates the probability of receiving the treatment based on covariates, known as the p**ropensity score**. In our example,studying the effect of job training on earnings, the propensity score might estimate the likelihood of receiving the training program based on factors like age, education. Individuals with **similar propensity scores are then matched between the treatment and control groups**. This helps create a balanced dataset where these covariates are similar in both groups.

To estimate these propensity scores, we can use logistic regression, but like always we need to be sure that our matching is relevant. Here’s how you can do that:

* **Variance Ratio:** compare the variance of a covariate in both groups. We want this ratio to be close to 1, meaning the groups are now similar.
* **A/A Testing:** We also compare the outcomes of the matched groups. If the groups are well-matched, any differences in outcomes should be small and not statistically significant, showing that the groups are comparable.

Alright, we’ve tackled the covariate balance — time to check that off our list!

Now, let’s dive into the exciting part: using the **DoWhy** and **EconML**packages to start our analysis. This is where the real fun begins!

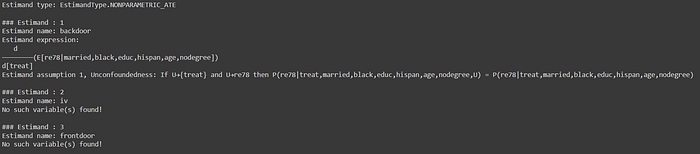
**3- Modeling with DoWhy and EconML**

Now, let’s set up our causal model!

In the code below, we’re defining our **treatment**, **outcome**, and the **factors** we think might influence both. In a real-world scenario, you should always design your **causal graph** (if you’re unfamiliar with causal graphs, check out my article on [causal inference](https://medium.com/@med.hmamouch99/causal-inference-part-2-79d1363d8285) for a deep dive):

from dowhy import CausalModel  
estimand = CausalModel(  
 data = df,  
 treatment='treat',  
 outcome='re78',  
 common\_causes=['nodegree', 'black', 'hispan', 'age', 'educ', 'married']  
)  
identified\_estimand = estimand.identify\_effect()  
print(identified\_estimand)

With DoWhy, it’s really simple to define your causal graph and estimand



Estimand output

The output of the estimand might look a little scary, but trust me, it’s straightforward once we break it down. Let’s analyze it step by step:

* **Estimand name (Backdoor)**: is the method we used to estimate our quantity the treatment effect, its a little bit complicated to explain it in this blog but in nutshell it help you estimate the treatment effect while controlling for specific variables, assuming no other unobserved confounders are affecting the treatment and outcome if you are interested to go deeper in details of this method or other you can chekc [my article](https://medium.com/@med.hmamouch99/causal-inference-part-2-79d1363d8285)
* **Estimand expression**: This represents the Average Treatment Effect, showing how job training affects the earning, after controlling for the common causes.
* **Estimand Assumption 1, Unconfoundedness**: This assumption means that, after we control for all the covariates (like race, age, education, etc.), there are no hidden factors influencing both the treatment and the outcome. In simpler terms, after adjusting for these variables, any remaining difference in earnings between treated and untreated individuals is **because of the treatment itself**.

Easy, right? Now that we’ve set up our estimand, it’s time for EconML to take over!

**3.1 EconML with DML: Estimating Treatment Effect**

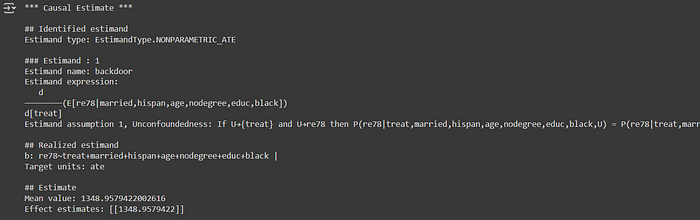
Next, we implement the Double Machine Learning estimator using EconML and apply it to our data. For this, we’ll use the LinearDML estimator.

Notice that we pass *‘backdoor.econml.dml.LinearDML’*to the*method\_name* parameter. Here’s where we add the extra argument *‘discrete\_treatment’* and set it to *True*. This is crucial because our treatment is binary (either you got the training or you didn’t), and by default, EconML assumes treatments are continuous.

In this example, we’ll use **Logistic Regression** to model the propensity score (the probability of receiving the treatment). For simplicity, we won’t dive into checking the quality of the matching here, but keep in mind that in a real-world scenario, it’s super important to evaluate it!

from sklearn.linear\_model import LinearRegression, LogisticRegression, LassoCV  
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor  
  
estimate = model.estimate\_effect(  
 identified\_estimand=estimand,  
 method\_name='backdoor.econml.dml.LinearDML',  
 target\_units='ate',  
 method\_params={  
 'init\_params': {  
 'model\_y': RandomForestRegressor(),  
 'model\_t': LogisticRegression(max\_iter=1000),   
 'discrete\_treatment': True  
 },  
 'fit\_params': {}  
 }  
)  
print(estimate)

Now, let’s calculate our **ATE**!



If you print the estimate, you’ll see the output includes the effect estimate and a few other key metrics. Let’s break it down:

* **Realized estimand**: This is the actual equation that the model has identified as being relevant for estimating the causal effect. You might be wondering: *“Why is the equation linear if we’re using fancy models like Random Forest?”* Great question! Yes, the final equation is linear, but don’t forget how DML works. In the first stage, DML uses non-linear models like Random Forest to predict both the outcome and the treatment based on covariates. This flexible, machine-learning-powered stage helps capture complex relationships. Then, in the second stage, a linear regression is applied to the residuals from those models to estimate the treatment effect.
* **Estimate**: it’s the actual treatment effect estimate, which is the result of all the heavy lifting our model has done behind the scenes.

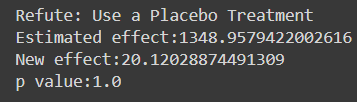
Pretty awesome, right? We’re using machine learning to get a reliable estimate of causal impact, while still keeping things interpretable! And the results speak for themselves — **the job training program had a solid impact**, with participants earning an average of **$1,348.95** more than those who didn’t participate. Not a bad outcome for the training, right?

Of course, we can’t just take our treatment effect estimate at face value. To truly trust our results, we need to validate them. One way to do this is by introducing a clever **refutation test**.

Imagine this: what if we replaced the actual treatment with a completely random, fake treatment? If the model still detects a significant effect, that would be a warning sign that our estimate might not be reliable. This method is known as a **placebo test**, and it’s one of the most useful tools in causal inference to check whether our model is truly picking up the causal effect or just random noise.

Here’s what happened when we ran the placebo test on our model:

print(model.refute\_estimate(estimand, estimate, method\_name="add\_unobserved\_common\_cause"))



Results of refutation test

What do these results tell us?

* **Original Estimated Effect**: The real treatment was estimated to increase earnings by about **$1,348**.
* **Placebo Effect**: Once we swapped the real treatment with a randomly assigned one, the estimated effect shrank dramatically to **$20.12**. This is exactly what we hoped to see — an almost non-existent effect when the treatment is random. If the placebo treatment had led to a large effect, we’d know our model might be overfitting or capturing spurious relationships.
* **p-value**: The p-value of **1.0** means that there is no statistically significant effect when we use the placebo treatment. This reassures us that the model isn’t falsely detecting a causal effect when there is none.

The placebo test essentially asks the question: “Is the effect real, or could it be due to random chance?” By using this refutation method, we’ve confirmed that the original estimate wasn’t just a statistical coincidence. The fact that the placebo test didn’t find a significant effect gives us confidence that our model is properly identifying the true causal impact of the job training program, rather than being misled by random patterns in the data.

In conclusion, after this hands-on experience, we can confidently say that the job training program has a significant impact on earnings. Congratulations to everyone who took part in the program — your commitment has truly paid off!

**Conclusion**

Awesome work on getting through this chapter! We tackled a lot:

* Why we need Causal Machine Learning even with tools like A/B Testing.
* A deep dive into Double Machine Learning and the Frisch-Waugh-Lovell Theorem.
* And of course, we wrapped up with some hands-on experience.

If you enjoyed this post and want to see more, consider following me on: