Causal ML for the Aspiring Data Scientist

Ross Lauterbach

January 4th, 2026

1 Introduction: Limitations of Machine Learning

As a data scientist in today's digital age, you must be equipped to answer a variety of questions that go far beyond simple pattern recognition. Typical machine learning is built on association; it seeks to find patterns in existing data to predict future observations under the assumption that the underlying system remains constant. If you train a model to predict house prices, you are asking the algorithm to find the most likely price given a set of features.

However, causal analysis introduces a "what if" component. It goes beyond observation to ask how the system would react if we actively changed a variable. This is the difference between noticing that people who buy expensive lattes are also likely to buy sports cars and understanding whether lowering the price of that coffee will actually cause an increase in car sales. In the world of causal inference, we are essentially trying to learn the underlying laws of a business or social system, allowing us to predict the outcomes of actions we haven't taken yet.

Causal analysis is critical in a variety of fields when we need to move beyond observing patterns to make decisions. Consider a medical researcher evaluating a new blood pressure medication and its effect on heart attack severity. With historical data, you might see that patients taking the medication actually have more severe heart attacks. A standard ML (Machine Learning) model would suggest the drug is harmful. However, this is likely due to confounding: doctors only prescribe the medication to patients who already have lower health. To find the truth, we must isolate the drug's actual impact from the noise of the patients' existing conditions.

In this article, I will introduce some of the important concepts and tools in causal ML in an accessible manner. I will only use libraries that manage data, calculate probabilities, and estimate regression parameters. This article is not a tutorial, but a starting point for those interested in but intimidated by causal inference methods. I was inspired by the online reading *Causal Inference for the Brave and True* by Matheus Facure Alves. Note: For those unfamiliar with probability, E[X] refers to the average value that a random variable/quantity x takes.

2 The Potential Outcomes Framework

When we start a causal study, the questions we ask are far more specific than loss minimization or prediction accuracy. We typically start with the Average Treatment Effect (ATE), which tells us the mean impact of an intervention or action across an entire population.

In our medical example, we want to know the difference in heart attack severity if the entire population took the drug versus if the entire population did not. To define this mathematically, we use the Potential Outcomes Framework. First, let's define our specific some variables:

- Y: The Outcome (e.g., a heart attack severity score from 0 to 100).
- T: The Treatment indicator. This is a binary "switch":
  - $T = 1$ means the patient took the drug.
  - $T = 0$ means the patient did not take the drug (the Control).
- $Y(1)$: The outcome we would see if the patient was treated.

- $Y(0)$: The outcome we would see if the patient was not treated.

The theoretical ATE is the expected difference between these two potential outcomes across the entire population:

$$ATE = E[Y(1) - Y(0)] \tag{1}$$

To address the dilemma of unobserved outcomes, researchers use the Potential Outcomes Framework as a conceptual guide. In this framework, we assume that for every individual, there exist two "potential" results: $Y(1)$ and $Y(0)$. We only ever observe one of these two values, which is known as the Fundamental Problem of Causal Inference.

If a patient takes the medication ($T = 1$), we see their factual outcome, $Y(1)$. Their outcome without the medication, $Y(0)$, is now a counterfactual, a state of the world that could have existed but didn't.

The limitation of causal inference is that for any given person, we only ever observe one of these two values. If a patient takes the medication, we see their factual outcome, $Y(1)$, while their outcome without the medication, $Y(0)$, remains a counterfactual, a state that could have existed but did not.

3 ATE in a Perfect World

Since the individual treatment effect is the difference between these two values, it remains hidden to us. This shifts the entire goal of causal estimation away from the individual and toward the group. Because we cannot subtract a counterfactual from a factual for one person, we must find clever ways to compare groups of people.

If the group receiving the treatment is statistically comparable to the group that is not, we can use the average observed outcome of one group to stand in for the missing counterfactual of the other. This allows us to estimate the Average Treatment Effect by calculating the difference between the mean outcome of the treated group and the mean outcome of the control group:

$$\widehat{ATE} = E[Y|T = 1] - E[Y|T = 0] \tag{2}$$

Suppose that for those who took the drug, we observed a mean heart attack severity of 56/100, compared to 40/100 for those who did not. If we attempt to estimate the causal effect by taking a simple difference in means, the data suggests that taking the drug led to a 16 point increase in severity.

$$E[Y|T = 1] = 56, \quad E[Y|T = 0] = 40 \implies ATE_{biased} = 16 \tag{3}$$

Unless this drug is among the most dangerous created, there is likely another mechanism at play. This discrepancy arises because we can only interpret a simple difference in means as the Average Treatment Effect if the treatment was assigned through a Randomized Controlled Trial (RCT), which ensures complete random assignment of treatment groups. Without randomization, the treated and control groups are not exchangeable and differ in ways that make a direct comparison difficult to do.

4 Randomization

The reason an RCT is the default method for calculating the ATE is that it helps eliminate Selection Bias. In our medical example, the 16-point harm we observed likely occurred because doctors gave the drug to the highest-risk patients. In this scenario, the treated group was already predisposed to higher severity scores before they ever took the pill. When we use an RCT, we remove the human element of choice. With this randomized selection, we ensure that high-risk and low-risk patients are distributed equally between both groups.

Mathematically, randomization ensures that the treatment assignment is independent of the potential outcomes:

$$(Y(1), Y(0)) \perp T \tag{4}$$

Now, we can assume that the average outcome of the treated group is a perfect proxy for what would have happened if the entire population had been treated. Because the "Treated" and "Control" groups start as statistical clones of one another, any difference we see at the end of the study must be caused by the drug itself.

5 Observational Data and Confounders

In the real world, we are often forced to work with observational data. In these situations, the simple difference in means fails us because of the presence of confounders. A confounder is a variable that influences both the treatment and the outcome, creating a "backdoor path" that allows a non-causal correlation to flow between them.

In order to visualize these hidden relationships, causal researchers use Directed Acyclic Graphs (DAGs). A DAG is a specialized graph where variables are represented as nodes and causal relationships are represented as arrows. Directed that the arrows have a specific direction, indicating a one-way causal flow from a cause to an effect. Acyclic means the graph contains no cycles you cannot follow a sequence of arrows and end up back at the first variable, mainly because transitioning from one node to the next should represent a lapse in time. A confounder will reveal itself in a DAG by its directed connection to both the treatment and the outcome, as seen below.

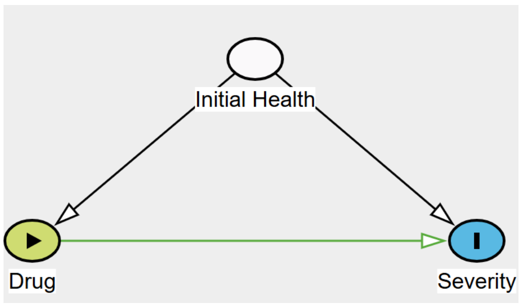


Figure 1: Causal DAG with Initial Health as a Confounder

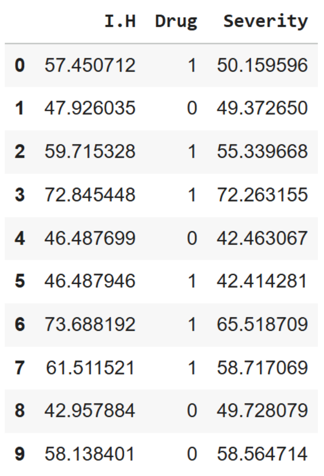
6 Adjustment via Linear Regression

Once we have identified the confounders through our DAG, the next step is to mathematically account for them. If we want to isolate the true effect of the drug, we need to compare patients who are similar in every way except for whether they took the medicine. In causal analysis, the most important tool for this is Linear Regression.

By including the confounder as an independent variable, the model calculates the relationship while holding the initial health of the patient constant. For our example, I generated a mock dataset where treatment assignment was dependent on initial health (I.H). This can be seen in the code below, where both the probability of receiving the drug and the severity is dependent on the initial health score.

```
import pandas as pd
import numpy as np

np.random.seed(42)
n = 1000

# 1. Initial Health
# Healthy patients have low scores, while sicker patients have higher scores
initial_health = np.random.normal(50, 15, n).clip(0, 100)

prob_drug = np.where(initial_health > 50, 0.8, 0.2)
drug = np.random.binomial(1, prob_drug)

# Adds the true drug effect plus the effect of health to the baseline,
    ensuring the confounder relationship holds
```

```
# TRUE EFFECT = -10 (The drug actually reduces severity by 10 points)
baseline = 10
health_effect = 0.9 * initial_health
true_drug_effect = -10
noise = np.random.normal(0, 5, n)

severity = baseline + health_effect + (true_drug_effect * drug) + noise
df = pd.DataFrame({'I.H': initial_health, 'Drug': drug, 'Severity': severity
    })

#mean_untreated = 48.67, mean_treated = 52.14
print(df.groupby('Drug')['Severity'].mean())
```



	I.H	Drug	Severity
0	57.450712	1	50.159596
1	47.926035	0	49.372650
2	59.715328	1	55.339668
3	72.845448	1	72.263155
4	46.487699	0	42.463067
5	46.487946	1	42.414281
6	73.688192	1	65.518709
7	61.511521	1	58.717069
8	42.957884	0	49.728079
9	58.138401	0	58.564714

Figure 2: Data Preview for Synthetic Dataset

In this view, individuals who received the drug had an average severity increase of 3.47 points. To find the truth, we run an OLS (Ordinary Least Squares) multiple linear regression model to control for the participants' initial health rating.

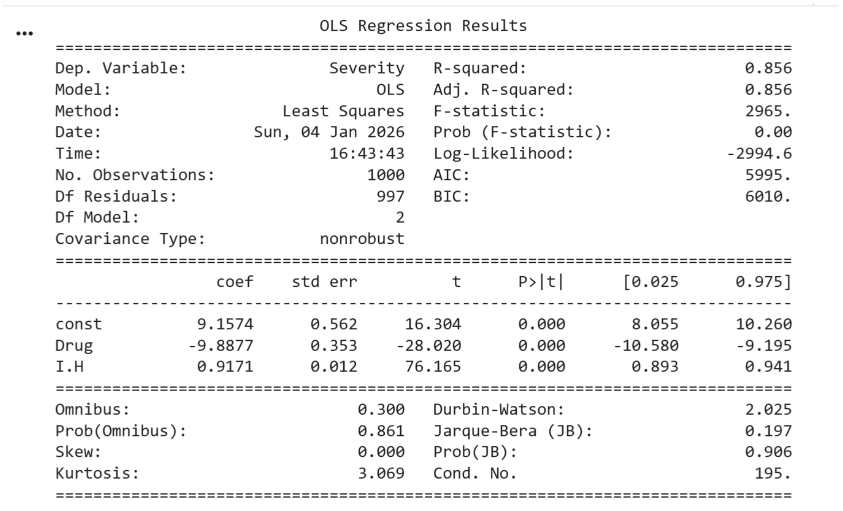
```
import statsmodels.api as sm
X = df[['Drug', 'I.H']]
X = sm.add_constant(X) #Needed for statsmodels OLS
y = df['Severity']
model = sm.OLS(y, X).fit()
print(model.summary())
```

The most important finding here is the coefficient of the treatment variable (drug). While the raw data suggested the drug was harmful, our coefficient is approximately -9.89. This suggests that when we control for the confounder of initial health, taking the drug actually decreases heart attack severity by nearly 10 points. This is very close to our true effect which was a decrease of exactly 10 points!

This is a result that was more in line with our expectations, and that is because we eliminated a large source of selection bias by controlling for confounders. The great thing about linear regression in this context is that the setup is similar to that of a typical regression problem. Transformations can be applied, diagnostic plots can be produced, and slopes can be interpreted as normal. However, because we are including confounders in our model, their effect on the outcome will not be absorbed into the treatment coefficient, something known as de-biasing or adjusting as previously mentioned.

7 Matching and Propensity Scoring

While multiple linear regression is a powerful tool for de-biasing, it relies heavily on the assumption that the relationship between your confounders and the outcome is linear. In many real-world situations, your treated and control groups might be so fundamentally different that a regression model is forced

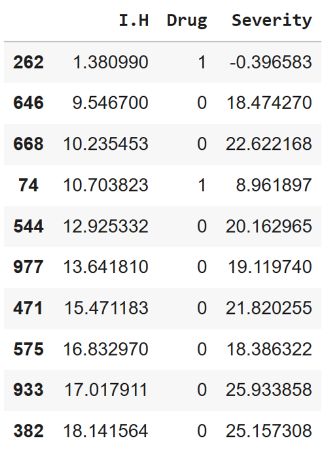


```
                            OLS Regression Results
==============================================================================
Dep. Variable:               Severity   R-squared:                       0.856
Model:                            OLS   Adj. R-squared:                  0.856
Method:                 Least Squares   F-statistic:                     2965.
Date:                Sun, 04 Jan 2026   Prob (F-statistic):               0.00
Time:                        16:43:43   Log-Likelihood:                -2994.6
No. Observations:                1000   AIC:                             5995.
Df Residuals:                     997   BIC:                             6010.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          9.1574      0.562     16.304      0.000       8.055      10.260
Drug          -9.8877      0.353    -28.020      0.000     -10.580      -9.195
I.H            0.9171      0.012     76.165      0.000       0.893       0.941
==============================================================================
Omnibus:                        0.300   Durbin-Watson:                   2.025
Prob(Omnibus):                  0.861   Jarque-Bera (JB):                0.197
Skew:                           0.000   Prob(JB):                        0.906
Kurtosis:                       3.069   Cond. No.                         195.
==============================================================================
```

Figure 3: OLS Output from Python

to guess results in areas where it has no actual data. To solve this, researchers often turn to Matching, a technique that shifts the focus from mathematical adjustment to data restructuring. Instead of using a formula to hold health constant, matching searches the control group for a ”twin” for every treated individual. When we pair a patient who took the drug ($T = 1$) with a patient of nearly identical initial health who did not ($T = 0$), we effectively prune our dataset into a Synthetic RCT. In this balanced subset, the groups are finally exchangeable, allowing us to compare their outcomes directly to reveal the true Average Treatment Effect (ATE). It is almost as if each pair allows us to observe both the factual and the counterfactual states for a single type of observation.

When we think of how to match two entries in a dataset, consider that each entry is represented by a vector in an n-dimensional space, where $n - 1$ is the number of features or confounders. At first glance, it seems we could simply calculate the distance between these vectors using Euclidean distance. However, the issue with this approach is that all covariates are weighted equally, regardless of their actual causal impact. In high dimensions, a problem known as the curse of dimensionality, even an entry's closest match could still be fundamentally different in the ways that actually matter for the treatment. In our mock dataset, looking at participants with the lowest health scores below, we see that treated participant 74 and untreated participant 668 have nearly identical initial health scores. Because we are only dealing with one confounder here, these two are ideal candidates to be matched together. However, as dimensionality increases, it becomes impossible to find these matches by just looking at the numbers, and simple Euclidean distance fails to prioritize the variables that truly drive the selection bias.



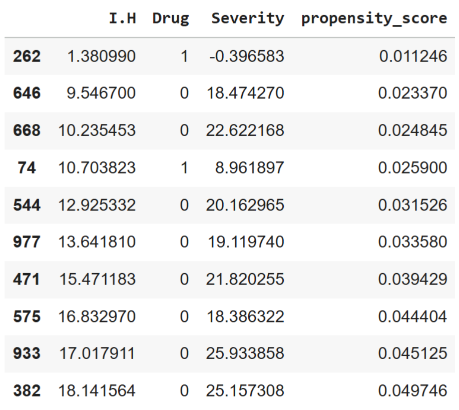
	I.H	Drug	Severity
262	1.380990	1	-0.396583
646	9.546700	0	18.474270
668	10.235453	0	22.622168
74	10.703823	1	8.961897
544	12.925332	0	20.162965
977	13.641810	0	19.119740
471	15.471183	0	21.820255
575	16.832970	0	18.386322
933	17.017911	0	25.933858
382	18.141564	0	25.157308

Figure 4: Data Preview for Low Initial Health Entries

In practice, this process is most commonly executed as one-to-one matching, where each treated unit is paired with its single closest neighbor in the control group. To ensure these matches are high-quality, we use the Propensity Score: a single number representing the probability that a participant would receive the treatment given their characteristics, $P(T = 1|X)$. This score collapses our high-dimensional space into a single dimension that specifically reflects the likelihood of treatments given a set of covariates. We then use a k-Nearest Neighbors (k-NN) algorithm to perform a "fuzzy" search on this score. To prevent poor matches, we can choose a threshold to serve as the maximum allowable distance to match. We can calculate propensity in a number of ways, the most common being logistic regression, but other ML methods capable of outputting probabilities such as XGBoost or Random Forest work as well In the below code, I calculated propensities by setting up a logistic regression model that predicts drug participation from just initial health. In practice, you would have more confounders in your model.

As mentioned, the first step of propensity score matching is the calculation of the propensity score. In our example, we only have initial health as a confounder, so that will be the sole covariate in our simple logistic regression.

```
from sklearn.linear_model import LogisticRegression
X = df[['I.H']]
y = df['Drug']
propensity_model = LogisticRegression().fit(X, y)
df['propensity_score'] = propensity_model.predict_proba(X)[:, 1]

df.sort_values(by = 'I.H', ascending = False)
```



	I.H	Drug	Severity	propensity_score
262	1.380990	1	-0.396583	0.011246
646	9.546700	0	18.474270	0.023370
668	10.235453	0	22.622168	0.024845
74	10.703823	1	8.961897	0.025900
544	12.925332	0	20.162965	0.031526
977	13.641810	0	19.119740	0.033580
471	15.471183	0	21.820255	0.039429
575	16.832970	0	18.386322	0.044404
933	17.017911	0	25.933858	0.045125
382	18.141564	0	25.157308	0.049746

Figure 5: Data Preview for Low Initial Health Entries with Propensity Score

As expected, participants 74 and 668 were assigned a very similar propensity, and would likely be matched. It is also often helpful to generate what is known as a Common Support plot, which displays the density of calculated propensity scores separated by treated and control. Ideally, we want to see as much overlap and symmetry as possible, as that implies matching units will be less difficult. As seen below, selection bias is present in our dataset. It is a good exercise to investigate the data generation code above and determine why.

Although not necessary in the one dimensional case, we can then use k-NN to match treated with untreated based on their propensity score.

```
from sklearn.linear_model import LogisticRegression

treated = df[df['Drug'] == 1]
control = df[df['Drug'] == 0]

nn = NearestNeighbors(n_neighbors=1) #One neighbor for every treated unit
nn.fit(control[['propensity_score']])
distances, indices = nn.kneighbors(treated[['propensity_score']])
```

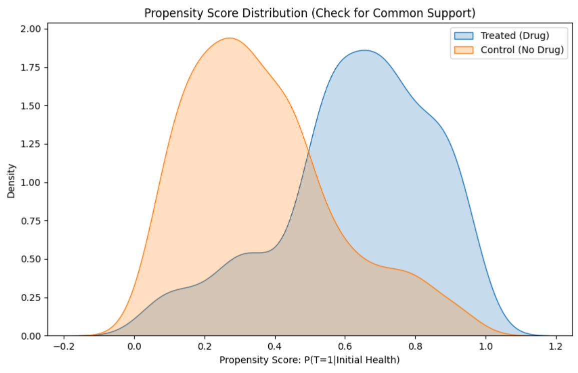


Figure 6: Common Support Plot for Propensity Scoring

```
matched_control = control.iloc[indices.flatten()]
ate_matched = treated['Severity'].mean() - matched_control['Severity'].mean
    ()

print(f"Matched ATE: {ate_matched:.2f}") #ATE = -10.16
```

If you recall from before, our linear regression yielded an ATE of -9.89 compared to our now calculated value of -10.16. As we increase the complexity and number of covariates in our model, our propensity score matching ATE will likely get closer and closer to the underlying causal effect of -10.

8 Time Invariant Effects Using Difference-in-Differences

While matching is excellent for de-biasing based on the variables we can see, it falls short when there are hidden factors, like a patient's genetic predisposition or a hospital's specific management style, that we haven't recorded in our data. If these unobserved confounders are time-invariant (meaning they stay constant over the study period), we can use Difference-in-Differences (DiD) to cancel them out.

Instead of just comparing the treated group to the control group at a single point in time, DiD looks at two groups over two periods: before and after the treatment. The logic is simple yet elegant: we calculate the change in the control group and assume the treated group would have changed by that same amount if they hadn't received the treatment. Any additional change observed in the treated group is attributed to the treatment itself. The equation for the DiD estimator is as follows:

$$\hat{\delta}_{DiD} = (\mathbb{E}[Y|T=1,\text{Post}] - \mathbb{E}[Y|T=1,\text{Pre}]) - (\mathbb{E}[Y|T=0,\text{Post}] - \mathbb{E}[Y|T=0,\text{Pre}]) \tag{5}$$

While this formula may appear intimidating at first glance, it is best read as the difference in changes occurring before and after treatment. For example, imagine two ice cream shops in different towns. Before the weekend, Store A (our treatment group) sells 200 cones, and Store B (our control group) sells 300. On Saturday, a heat wave hits Store A's town, but not Store B's. By the end of the day, Store A's sales jump to 500, while Store B's sales rise to 400. A simple analysis of Store A would suggest the heat wave caused a +300 increase. However, the control shop (Store B) grew by +100 in the same period without any heat wave, perhaps due to a holiday or general summer weather. The Difference-in-Differences approach subtracts this natural time trend of +100 from Store A's total growth. It effectively cancels out any time-invariant confounders—factors like the store's location or its base popularity that would have otherwise skewed our results. This reveals that the true causal impact of the heat wave was +200 units.

A significant limitation of the basic Difference-in-Differences (DiD) is that it does not account for factors that change over time. While the "change-in-change" logic successfully cancels out static, time-invariant confounders (like someone's genetic history or a hospital's geographic location), it remains vulnerable to time-varying confounders. These are factors that shift during the study period and affect the treatment and control groups differently.

In our heart attack study, for instance, even a DiD analysis could be biased if the hospitals administering the drug also underwent significant staffing changes or received upgraded equipment during the "Post" period. If we fail to account for these changing variables, the DiD estimator will incorrectly attribute their impact to the drug itself, leading to a "polluted" causal estimate.

It is important to note that the simple cross-sectional data structure we utilized for Regression and Matching is insufficient for this method. To calculate a "change in the change," we need a temporal dimension in our dataset. Specifically, we need a variable indicating whether an observation occurred in the Pre-treatment or Post-treatment period for both the treated and control groups.

To resolve this, we move beyond simple subtraction and implement DiD within a Multiple Linear Regression framework. This allows us to explicitly "control" for time-varying factors, effectively isolating the treatment effect while holding external shifts constant.

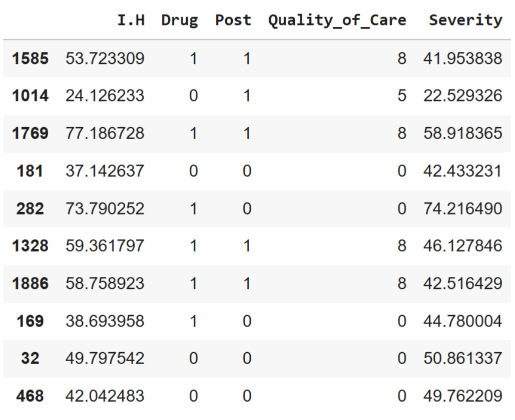
The regression model is defined as:

$$Y_{it} = \beta_0 + \beta_1 \text{Drug}_i + \beta_2 \text{Post}_t + \beta_3 (\text{Drug}_i \times \text{Post}_t) + \gamma \mathbf{X_{it}} + \epsilon_{it} \tag{6}$$

Where:

- β_0: The baseline severity for the control group in the "Pre" period.
- β_1: Captures the permanent, time-invariant differences between the two groups.
- β_2: Captures the natural time trend (e.g., general improvements in medical standards over time).
- β_3: The DiD Estimator. This represents the true causal effect of the drug.
- $\gamma \mathbf{X_{it}}$: Represents our time-varying controls, such as Staffing Levels or Quality of Care indices.

Below, a new synthetic dataset is constructed to reflect the required structure. I also added a Quality of Care variable for demonstration purposes. I did not include the full simulation code due to its length, but it essentially modifies the previous logic by duplicating our observations across two distinct time periods.



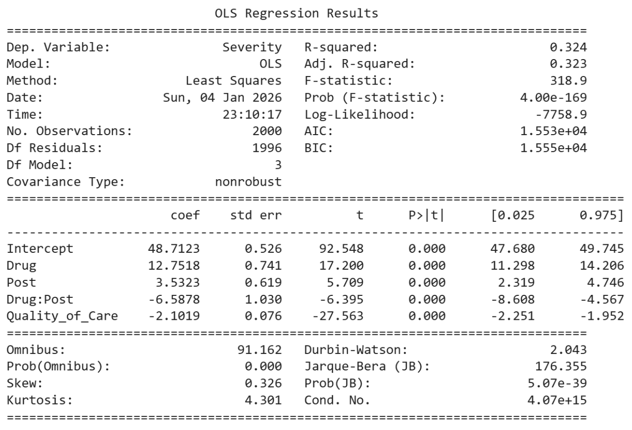
	I.H	Drug	Post	Quality_of_Care	Severity
1585	53.723309	1	1	8	41.953838
1014	24.126233	0	1	5	22.529326
1769	77.186728	1	1	8	58.918365
181	37.142637	0	0	0	42.433231
282	73.790252	1	0	0	74.216490
1328	59.361797	1	1	8	46.127846
1886	58.758923	1	1	8	42.516429
169	38.693958	1	0	0	44.780004
32	49.797542	0	0	0	50.861337
468	42.042483	0	0	0	49.762209

Figure 7: Data Preview for DiD Linear Regression

Since we have our data in the correct format, we can fit a linear regression model using the specifications just mentioned.

```
m = smf.ols('Severity ~ Drug * Post + Quality_of_Care', data=df_did).fit()
```

The R-squared value of 0.324 indicates that the model explains approximately 32.4 percent of the variance in heart attack severity. In causal analysis, this is common as many unmeasured factors like genetics are treated as noise. The intercept of 48.71 represents the baseline severity for the control group during the pre-treatment period. The drug coefficient of 12.75 confirms selection bias, showing the treated group initially had higher severity scores. Additionally, the quality of care coefficient suggests each unit increase in that index corresponds to a 2.10-point reduction in severity.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:               Severity   R-squared:                       0.324
Model:                            OLS   Adj. R-squared:                  0.323
Method:                 Least Squares   F-statistic:                     318.9
Date:                Sun, 04 Jan 2026   Prob (F-statistic):          4.00e-169
Time:                        23:10:17   Log-Likelihood:                -7758.9
No. Observations:                2000   AIC:                         1.553e+04
Df Residuals:                    1996   BIC:                         1.555e+04
Df Model:                           3
Covariance Type:            nonrobust
===================================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------
Intercept          48.7123      0.526     92.548      0.000      47.680      49.745
Drug               12.7518      0.741     17.200      0.000      11.298      14.206
Post                3.5323      0.619      5.709      0.000       2.319       4.746
Drug:Post          -6.5878      1.030     -6.395      0.000      -8.608      -4.567
Quality_of_Care    -2.1019      0.076    -27.563      0.000      -2.251      -1.952
==============================================================================
Omnibus:                       91.162   Durbin-Watson:                   2.043
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              176.355
Skew:                           0.326   Prob(JB):                     5.07e-39
Kurtosis:                       4.301   Cond. No.                     4.07e+15
==============================================================================
```

Figure 8: DiD Linear Regression Results

The interaction term, drug:post, provides the difference-in-differences estimator, which reveals an estimated drug effect of -6.58. This tells us the medication reduced severity after adjusting for group differences and time trends, though the estimate is notably lower than the true effect of -10. This discrepancy occurs because quality of care improved specifically for the treated group during the post-treatment period, due to the data generation process. Since these two changes happened simultaneously to the same group, they are perfectly correlated, or collinear. The model essentially faces a mathematical stalemate where it cannot determine if the improvement came from the drug or the better care, so it splits the credit between them. As for any linear regression, if two variables are perfectly correlated, a model might drop one entirely or provide highly unstable estimates. Nevertheless, all variables maintain p-values of 0.000, confirming that despite the split credit, the results remain statistically significant. In real data and analysis, we will deal with these kinds of situations, and it is important to know of all the tools in your data science shed before you tackle a problem.

9 Conclusion and Final Thoughts

In this article, we explored the transition from standard ML to the logic of causal inference. We saw through synthetic examples that while simple differences in means can be misleading due to selection bias, methods like linear regression, propensity score matching, and difference-in-differences allow us to strip away confounders and isolate true impact.

Having these tools is our arsenal is not enough. As seen with our final model, even sophisticated techniques can yield issues when interventions overlap. While these methods are powerful in adjusting for confounding, they require a deep understanding of their underlying mechanics. Relying on model outputs without acknowledging the reality of collinearity or time-varying factors can lead to misleading conclusions. At the same time, knowing when and how to apply these tools can serve as a valuable skill to any data scientist. In my opinion, one of the best parts of doing statistical programming for causal inference is the fact that most of the methods stem from a few fundamental statistical models, making implementation easier than one might expect.

The real world is undeniably messy and full of data issues, and it is rare that we will observe a perfectly clean causal signal. Causal machine learning is ultimately about exploiting the right data while having the confidence that our variables allow for true adjustment. This article is my first step in the documentation of my causal inference journey, and I plan to release a part two that dives deeper into more topics, including Instrumental Variables (IV), Panel Regression, Double Machine Learning (DML), and Meta-Learners.

Suggested Reading

[1] Facure, Matheus. *Causal Inference for the Brave and True.* Available at: `https://matheusfacure.github.io/python-causality-handbook/`