Unlocking Insights: Estimating Causal Effect Using Propensity Score Matching

Nir Shlomo · Follow
Published in Riskified Tech
10 min read · Jul 25

At the end of reading this post, you will be able to understand how to estimate a causal effect with retrospective data, get familiar with the concept of confounders, learn how to balance your data using Propensity Score Matching, and most importantly, you will have fully available code to use for your research. If you feel comfortable with Propensity Score Matching, skip to the real-life example and use the code snippets.

It often occurs that we want to estimate the effect of a new intervention or method (new model, new feature) on our KPIs (pre-defined outcomes), or try to explore the causal path of some variable (is the difference we observed in the data, dependent or independent of other variables?). In both cases, the ultimate and most definitive way to do it would be by experimenting (aka A/B testing), assigning each observation randomly to intervention or control, and measuring the outcome.

There are a few drawbacks if we think of running an experiment:

- Resources: it is both expensive and time-consuming.

- Design: The experiment goals and methods should be pre-specified and well-defined, and changes during the experiment might decrease validity.
- Allocation to a treatment group: Mostly done in a 1:1 or 2:1 allocation, meaning that many subjects don't get the optimal intervention during the experiment.

In many cases, the solution for estimating the causal effect without experimenting is by analyzing retrospective data (RD). RD is available to us and is usually much cheaper to collect. The field of using RD to conclude a particular phenomenon is called Observational Studies.

Today, data is everywhere, available almost instantly by just pressing a button, so we might assume that we only need to analyze it and estimate the causal effect. Unfortunately, this is not an easy task; we are all familiar with the adage "association is not causation."

By simply performing a crude RD analysis, we can only conclude the association between the intervention we are exploring and the outcome we are measuring. Why is that? Because we didn't consider factors affecting the probability of a subject being or not being under intervention. Those factors are called confounding variables (or just confounders).

More about confounders

Confounders are those variables that augment the effect we want to measure and cause false estimates. A very famous example is the association between drinking coffee and having lung cancer, where in observational studies, an increased risk was observed. While if we control for smoking behavior, the observed risk of coffee drinking is diminished.

The idea is that people who drink coffee tend to smoke during their coffee drinking, and smoking is a well-established risk factor for lung cancer.

The main problem with using retrospective data is the difficulty of handling confounder variables and, therefore, estimating a wrong effect. This means that what we see in a retrospective analysis wouldn't necessarily happen in real life.

A formal definition of a causal effect

Suppose Y represents the outcome of a random event. $Y = 1$ is where the event occurred and $Y = 0$ is where the event did not occur. In addition, consider a dichotomous intervention A, where $a=1$ means the intervention was taken and $a = 0$ was not. For example, $Y^{a=1} = 1$ is where event Y did occur under intervention A.

We define a causal effect if $Pr[Y^{a=1} = 1]\ Pr[Y^{a=0} = 1]$ in the population of interest. Meaning, the probability of event Y under intervention A is not equal to the probability of event Y under no intervention A.

In the case where randomization is missing, as we are dealing with retrospective data, the probability of an event Y is conditional on both A and Z, where Z is a set of covariates and $Pr[Y^{a=1} = 1| Z]\ Pr[Y^{a=0} = 1| Z]$. If the distribution of Z differs between A, causal interpretation is questionable due to confounding variables Z.

Prior assumptions we should think about before

When trying to use RD to estimate the causal effect, it is important to think of a few assumptions that, without them being met, our ability to measure the true effect is questionable.

- **Exchangeability** — The average outcome Y is independent of the group it was allocated to. It's easy to think of it: if by mistake, we assign treatment to the control group, the average outcome would be the same for both treated and control groups. In experiments, exchangeability is expected by design due to randomization. In RD, exchangeability can be achieved within the level of the data, this point will be demonstrated in the following sections.

- **Positivity** — Each example in our data set should have a probability greater than zero of being assigned to one of the study groups. $Pr[A = a| Z = z] > 0$, for example, in research where we investigate the effect of a new feature on two different operating systems, but the new feature is unavailable to the latest version of one operating system. Unlike exchangeability, positivity is something that can be empirically tested.

- **Consistency** — A well-defined intervention A that is consistent across all observations. We generally don't want multiple versions of the intervention. If the intervention is not well-defined, the average causal effect is not well-defined either.

In retrospective data, we often encounter inconsistency in both the intervention and the outcome. sometimes because definitions of A or Y are changing over time.

Estimating the causal effect using PS Matching

Propensity score analysis is one of the most commonly used methods for causal inference in observational studies. The basic idea is to estimate the conditional probability of receiving the intervention, given a set of explanatory variables Z $Pr[A = 1| Z]$. This process should be done for each observation in the dataset, regardless of what the assignment actually was. $P(Z)$ is referred to as the propensity score.

When we conduct randomized A/B testing with a 1:1 assignment, the propensity score is $P(Z) = 0.5$. But in retrospective data, $P(Z)$ is unknown and should be estimated from the data. Using the propensity score, we create a balanced population where $P(Z| A = 1) = P(Z| A = 0)$, forming a matched population in which the group under intervention and the group under no intervention are exchangeable because they have the same distribution $P(Z)$.

Using Propensity Score Matching for groups balancing

The most common approach to estimating propensity score $P(Z)$ is by fitting a Logistic Regression. Where (and this is the tricky part) the dependent variable is A ($A = 1$ is the group under intervention and $A = 0$ is the group under no intervention), conditional on a set of covariates Z. Having the logistic regression model fitted, we calculate the propensity score (the conditional probability) of each observation in the data, and for each observation from the intervention group, we try to match one or more observations, with the same propensity score according to the similarity criteria we define.

Bias-variance tradeoff in propensity matching

The process where we define the similarity criteria entails a bias-variance tradeoff with hyperparameters we can tweak. If the similarity criteria are too loose, the balanced population will differ between intervention and no intervention $P(Z| A = 1) \neq P(Z| A = 0)$. Two important factors should be taken into account when performing Propensity Score Matching:

Selecting the set of covariates Z:

There are no clear guidelines on how to select the set of covariates *Z* to estimate the propensity score. On one hand, the most important consideration should be around the inclusion of potential confounders and exchangeability. On the other hand, we can't always tell for sure what covariates are considered potential confounders.

Because we are not dealing with a prediction model, the goal is not to minimize a cost function. Moreover, a highly discriminative model could decrease the balanced population size.

My suggestion is to use as many covariates as possible that improve the model fit of the logistic regression while maintaining fair discrimination. The ROC curve is a good way to assess model fit and discrimination.

Setting the matching caliper:

For each observation in the intervention group, with propensity score value *P(Z)*, we match one or more observations from the control group with the same *P(Z)*. Of course, it's unlikely that two observations will have exactly the same *P(Z)* so we match with a close value; this is called a caliper tuning and is beyond the scope of this post. Caliper tuning has a direct effect on the resulting balanced population and exchangeability.

Estimating the balance of the matched population

At the end of the matching process, we end up with a matched population that we believe is exchangeable $P[Z| A = 1] = P[Z| A = 0]$. Or simply we want to make sure that there is no substantial difference in the distribution of the covariates *Z*, but more importantly, the distribution of the potential confounders.

One good approach is to compare the Standardized Mean Difference (SMD) of each covariate between the intervention and the control group in the matched population. Here's more information about how to calculate SMD and its definition.

Post-matching analysis

The Propensity Score Matching forces us to restrict the post-matching analysis to the intervention group with the overlapping estimation of the propensity score. This is a serious drawback of PS Matching.

Imagine that many observations from the control group did not overlap and therefore, were not included in the balanced data set. We can't estimate the true marginal causal

effect.

How exactly should we estimate the causal effect between *A* and *Y*? It is highly dependent on the balanced population, and our confidence in dealing with confounders. If we believe all the prior assumptions are met, the balanced population is exchangeable, and with no residual confounders left, we can treat our balanced population as if it was randomly assigned and estimate the causal effect.

A real-life example

The example I'm about to walk you through is a hyper-simplified version of research we performed at Riskified. Riskified, a fraud management platform, collects data from multiple and varied data sources, digests, and leverages it to evaluate the risk of online, real-time, transactions being fraudulent.

The purpose of the analysis described below was to estimate the causal effect of using our calculated risk score to optimally route an online transaction to different acquirer banks *(A)*, in order to improve the bank authorization rate *(Y)*. The fraud risk score and additional characteristics represent a given transaction play as our set of covariates *(Z)*.

Obviously, the ultimate solution to this question would be conducting an A/B test where we randomly assign consecutive transactions to each acquirer A or B. Since A/B testing might not only cost time and resources, but also drive non-optimal recommendations for a period of time, and since we have available retrospective data, we decided to use Propensity Score Matching analysis to estimate the causal effect.

To download the Python code, go to [https://github.com/Riskified/ps-matching.git]

Or just clone to the repo: git checkout https://github.com/Riskified/ps-matching.git

We can run the entire matching process from the main.py file:

```
1  PS_GROUP = 'acquirer' #set the group variable (treatment/control)
2  TARGET = 'target' #set the target variable, the outcome of interest
3  FILE_PATH = 'data/df.csv' # dataframe contains all dependant and independent variables
```

initiate_params.py hosted with ❤ by GitHub view raw

```python
if __name__ == "__main__":
    data = PrepData(FILE_PATH, group=PS_GROUP, target=TARGET,  index_col="id")
    scorer = PScorer()
    scorer.fit(data.input, data.group_label)
    ps_scores: Series = scorer.predict(data.input)
    ScorePlotter.plot_roc_curve(ps_scores, data.group_label)

    matcher = ObsMatcher(n_matches=1, caliper=0.001)
    matched_index: List[int] = matcher.match_scores(ps_scores, data.group_label)

    ScorePlotter.plot_smd_comparison(
        data=data.input,
        matched_index=matched_index,
        treatment=data.group_label
        )
```

run_main.py hosted with ❤️ by GitHub view raw

A quick way to assess the difference between acquirer A and acquirer B is by comparing the Standardized Mean Difference (SMD) for each covariate. We can use the tableone package:

			Grouped by acquirer			
		Missing	Overall	acquirer_A	acquirer_B	SMD (acquirer_A,acquirer_B)
n			49995	38708	11287	
Total Spent, mean (SD)		0	131.8 (63.7)	128.4 (63.9)	143.6 (61.5)	0.242
credit_card_type, n (%)	CREDIT	0	22998 (46.0)	16743 (43.3)	6255 (55.4)	0.245
	DEBIT		26997 (54.0)	21965 (56.7)	5032 (44.6)	
credit_card_company, n (%)	company_1	0	28435 (56.9)	24081 (62.2)	4354 (38.6)	0.487
	company_2		21560 (43.1)	14627 (37.8)	6933 (61.4)	
model_score_category, n (%)	high_score	0	12586 (25.2)	10725 (27.7)	1861 (16.5)	0.385
	intermediate_score		24909 (49.8)	19658 (50.8)	5251 (46.5)	
	low_score		12500 (25.0)	8325 (21.5)	4175 (37.0)	
authorization, n (%)	1	0	41901 (83.8)	32957 (85.1)	8944 (79.2)	0.155

Difference between acquirer A and B in the set of covariates Z (mean and s.d for continuous variables; number and percent for categorical variables). The differences are very clear. The outcome of interest is the authorization rate where acquirer A has 85.1% and acquirer B 79.2%.

Creating a balanced dataset (step by step)

Step 1: Initiate data preparation class

```
data = PrepData(FILE_PATH, group=PS_GROUP, target=TARGET,  index_col="id")
```

data_prep.py hosted with ❤️ by GitHub view raw

Step 2: fit a logistic regression:

Now we fit a logistic regression with the formula:

$$\log\left(\frac{P(A)}{1-P(A)}\right) = \beta_0 + \beta_1 Z_1 + \beta_2 Z_2 + \beta_3 Z_3 + \beta_4 Z_4$$

where Z_1 is the order_total_spent, Z_2 is the credit_card_type, Z_3 is the credit_card_company and Z_4 is the model_score_categoty

```
scorer = PScorer()
scorer.fit(data.input, data.group_label)
ps_scores: Series = scorer.predict(data.input)
```

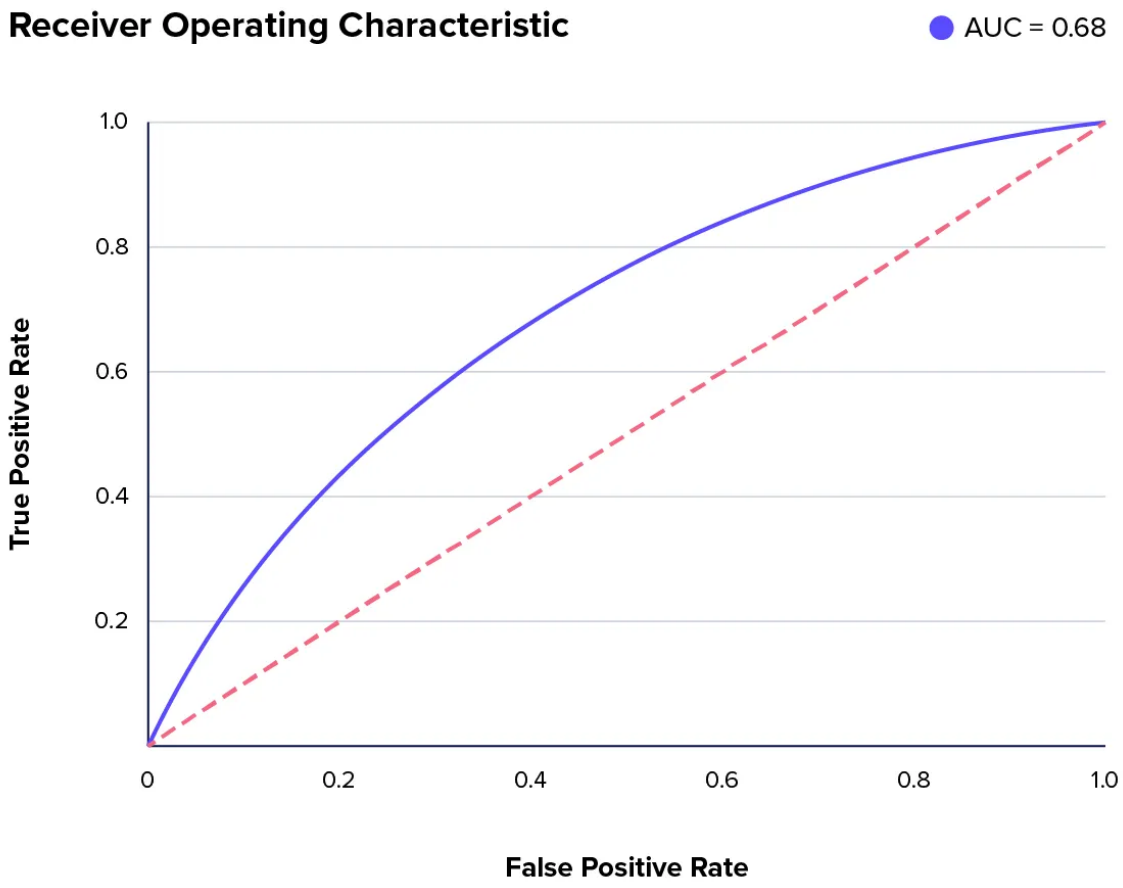
fit_logistic.py hosted with ❤️ by GitHub view raw

Step 3: Assess model fit

ROC curve can help us evaluate model discrimination and understand the overlapping area, i.e., exchangeability. In addition, it can help us choose which variables should be included in the regression model. ***We would like to keep the AUC somewhere around 70% to 80%.*** Too high AUC would reduce the overlapping area, and result in a small subset of the matched population. Too low AUC might indicate untreated residual confounding created by unmeasured factors

```
ScorePlotter.plot_roc_curve(ps_scores, data.group_label)
```

roc_curve.py hosted with ❤️ by GitHub view raw



Step 4: Create a balanced dataset

In the below example, I use a simple algorithm to find similarities. For each observation in acquirer B, we calculate the propensity score delta for all the observations in acquirer A and take the n closest observations. Currently, the method is without resampling.

We need to specify two parameters:

1. n_matches — the number of matching ratio (n_matches=2 get 2:1 matching)
2. caliper- defines the minimal distance that is allowed for the matching

```
1  matcher = ObsMatcher(n_matches=1, caliper=0.001)
2  matched_index: List[int] = matcher.match_scores(ps_scores, data.group_label)
```

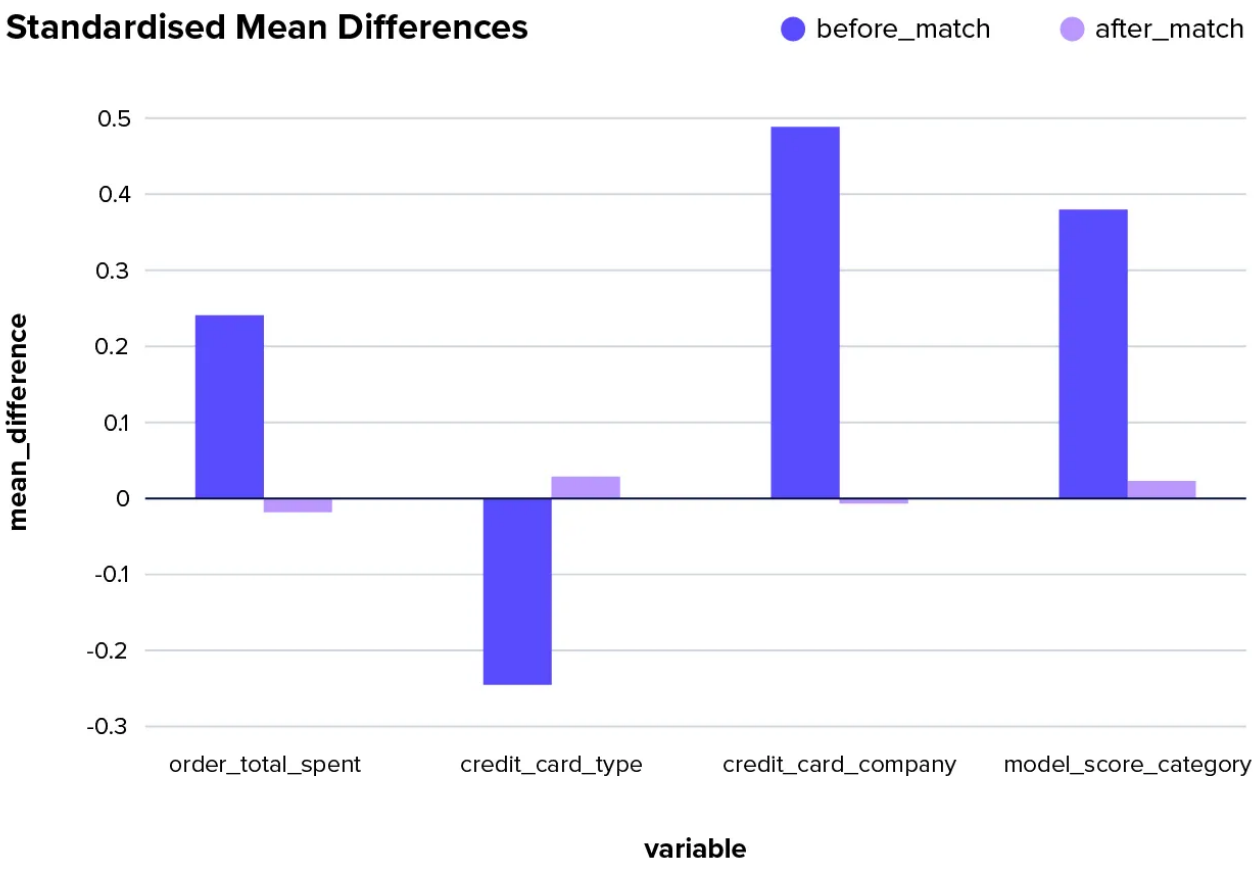
balanced.py hosted with ❤️ by GitHub view raw

Step 4: Evaluate the balanced population

Using the SMD plot, we can easily compare the differences between the original dataset and the matched dataset. In general, we would like to keep the SMD below 0.1 in the matched dataset.

```
ScorePlotter.plot_smd_comparison(
        data=data.input,
        matched_index=matched_index,
        treatment=data.group_label
        )
```

smd_plot.py hosted with ❤️ by GitHub view raw



In the plot above: before matching, the set of covariates *Z* has an SMD value above 0.1 and in the balanced population, SMD values are reduced.

Step 5: Estimate the causal effect

We run the tableone code again on the **matched** dataset:

			Grouped by acquirer			
		Missing	Overall	acquirer_A	acquirer_B	SMD (acquirer_A,acquirer_B)
n			29918	19908	10010	
Total Spent, mean (SD)		0	140.8 (61.9)	141.2 (62.4)	140.1 (61.1)	-0.016
credit_card_type, n (%)	CREDIT	0	16418 (54.9)	11020 (55.4)	5398 (53.9)	0.029
	DEBIT		13500 (45.1)	8888 (44.6)	4612 (46.1)	
credit_card_company, n (%)	company_1	0	12697 (42.4)	8430 (42.3)	4267 (42.6)	0.006
	company_2		17221 (57.6)	11478 (57.7)	5743 (57.4)	
model_score_category, n (%)	high_score	0	5506 (18.4)	3661 (18.4)	1845 (18.4)	0.038
	intermediate_score		15252 (51.0)	10261 (51.5)	4991 (49.9)	
	low_score		9160 (30.6)	5986 (30.1)	3174 (31.7)	
authorization, n (%)	1	0	25279 (84.5)	17011 (85.4)	8268 (82.6)	0.078

Here's the magic happening; the differences in the set of covariates *Z* almost completely diminished. In addition, the authorization rate difference we saw in the crude data is reduced to 85.4% in acquirer_A and 82.6% in acquirer_B. Using the balanced dataset, we were able to estimate the causal effect of the acquirer on the authorization rate.

Causal Inference | Propensity Score | Matching | Retrospective Data | Data Science

Follow

Written by Nir Shlomo

11 Followers · Writer for Riskified Tech