Case Study #4: Long-Term Return-on-Investment at Microsoft via Short-Term Proxy

ALICE Team, MSR New England

Maggie Hei

Microsoft EconML

Session Goals

- Understand the common challenges of learning **holistic ROI** from both business and technical perspective
- Introduce a **unified pipeline** to estimate the long-term effect of multiple investments from observational data in a high dimensional manner
- Learn how to use **EconML** Python package to solve this problem in a few lines of code

Outline

- Business Background
- Technical Challenges
- Methodology
- EconML Solution
- Wrap up

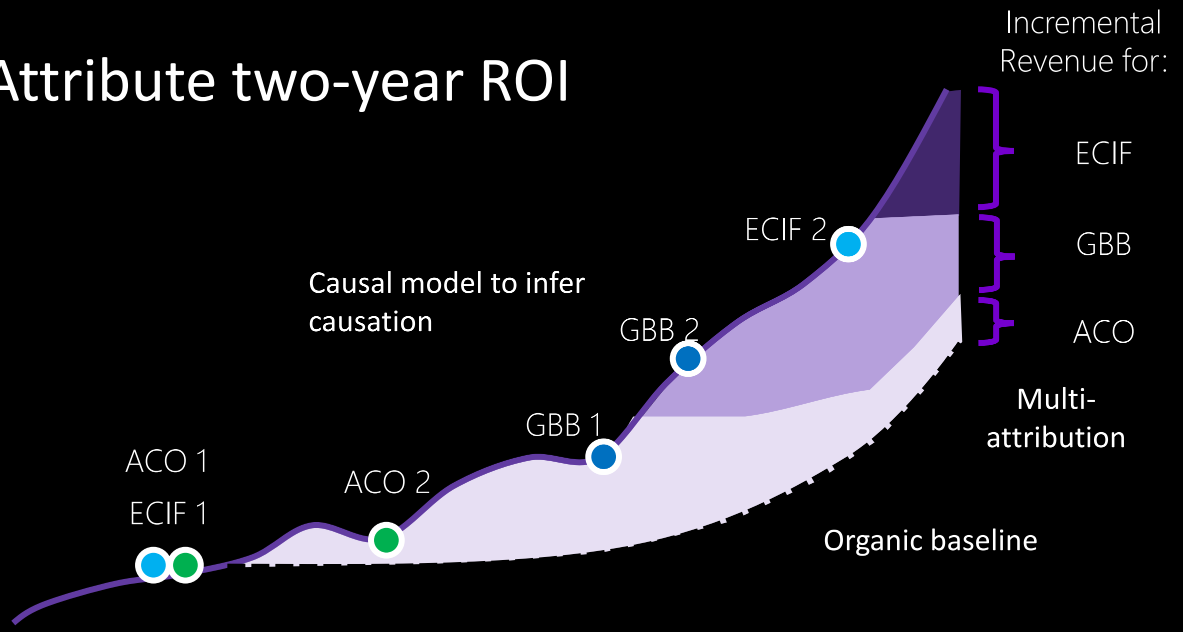
Business Background

- Microsoft provides multiple monetary fundings to enterprise customers in support of products adoption. Which of these programs (“investment”) are more successful than others?
- **Issues:**
  - Different models used to calculate Return on Investment separately for each program.
  - We want to estimate long-term success (e.g. two-year effect), but we can’t wait that long to evaluate a program.
- **Goal:** Can we **attribute the long-term ROI** of multiple programs in a holistic manner with only **short-term data**?

Why Holistic ROI?

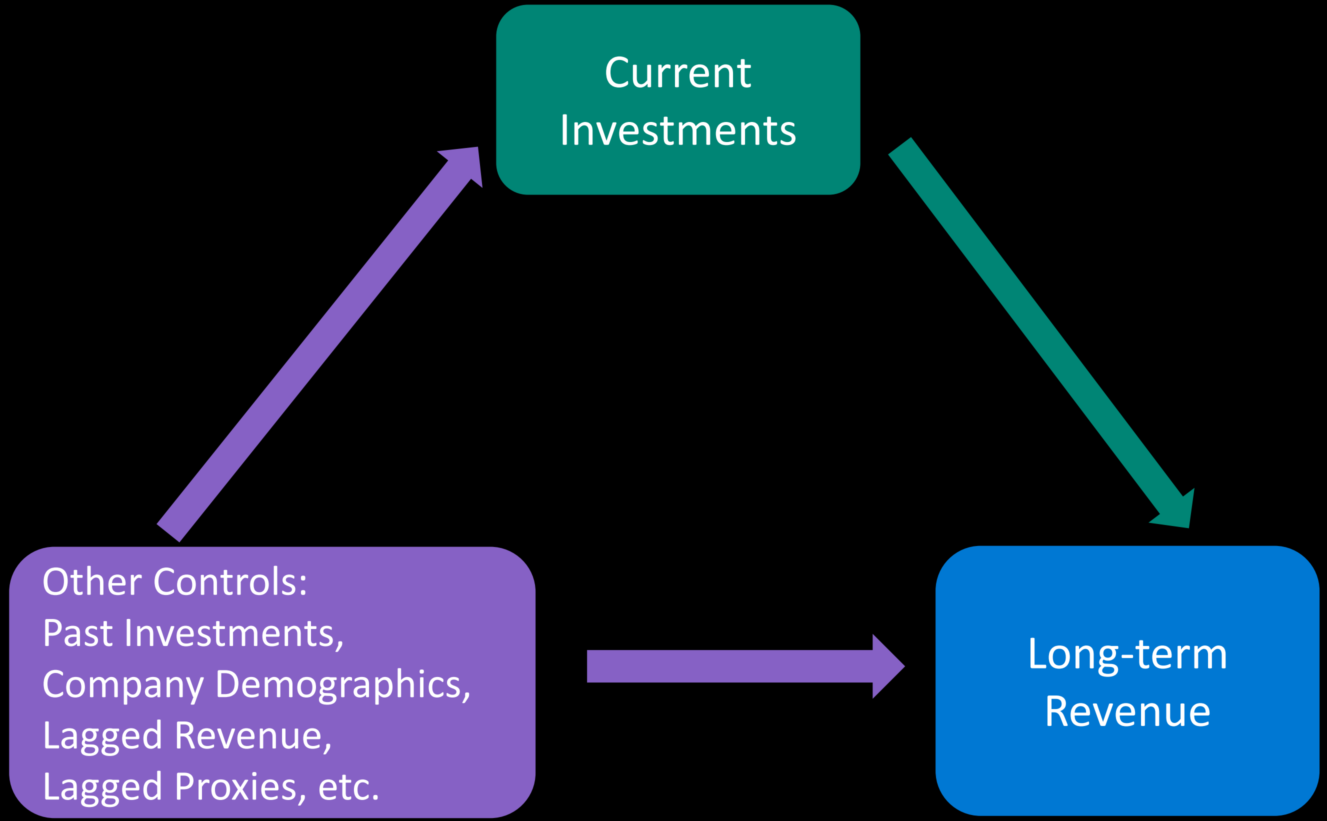
- Holistic ROI model gives us a way to assess the **relative impact of multiple investments across accounts, over time**.
  - Holistic view
  - Organic growth
  - Causal model
  - Attribution across investments

Attribute two-year ROI



Technical Challenge: Confounding Effect

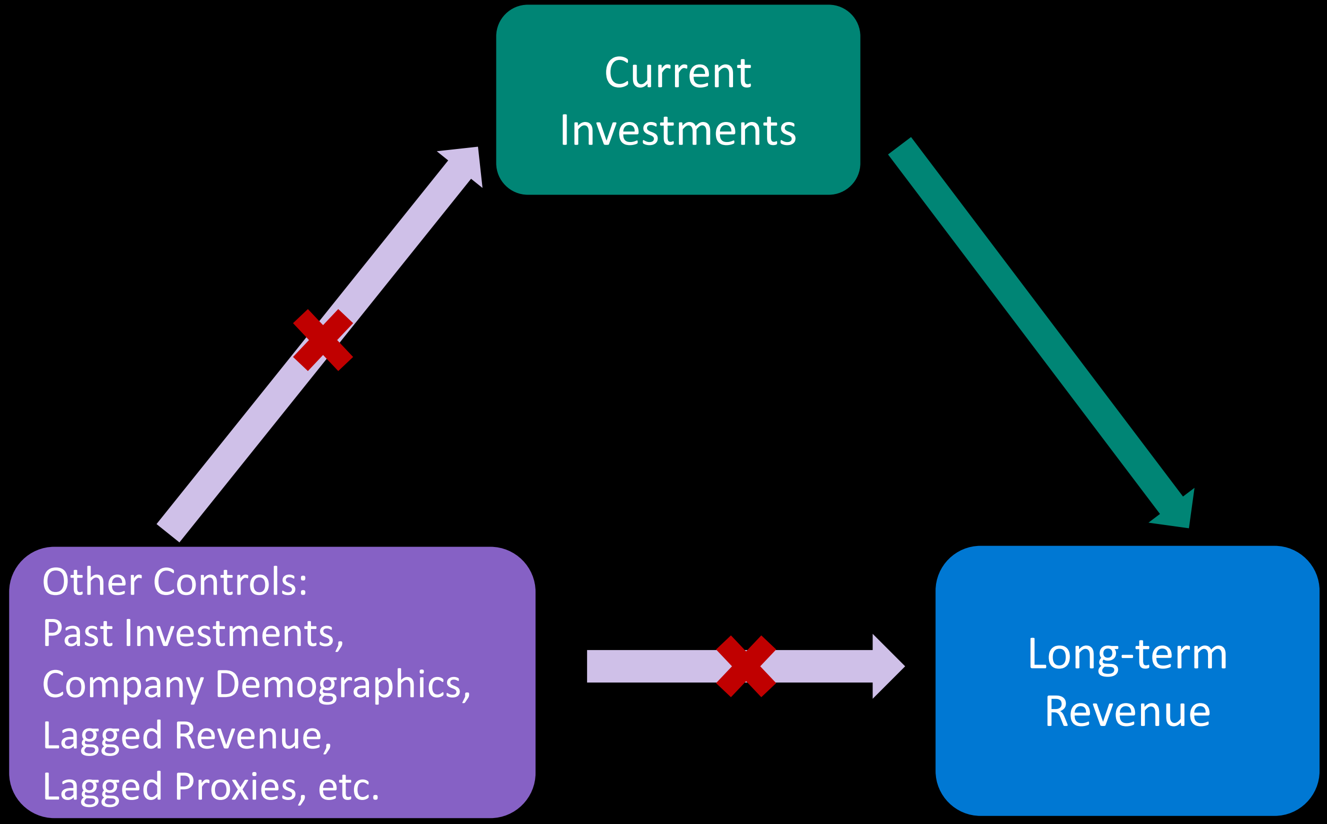
- Start with simple causal graph



- **Hidden Factors** that affect both the target investment and the long-term revenue:
  - Larger customers might receive more investments
  - Customers would purchase some products even with no incentives
  - The effect might come from the past incentives

Methodology

- Careful causal modeling



- **Double Machine Learning (DML)** nets out effects from confounders by fitting a two stage ML model
  - Allow high dimensional dataset
  - Construct confidence intervals

Only **direct effect** of target investments on future revenue remains!

Double Machine Learning

(Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen,Whitney Newey, and James Robins. "Double/debiased machine learning for treatment and structural parameters")

$$Y = \boldsymbol{\theta}(\boldsymbol{X}) \times T + g(X, W) + \varepsilon$$
$$T = f(X, W) + \eta$$

1. Regress $\mathrm{Y} \sim$ W, learn $\hat{\mathrm{Y}}$
   - **Predict future revenue** using past investments and current customer attributes
2. Regress $\mathrm{T} \sim$ W, learn $\hat{\mathrm{T}}$
   - **Predict current investments** using past investments and current customer attributes
3. Linear regression on residuals: $(\mathrm{Y} - \hat{\mathrm{Y}}) \sim (\mathrm{T} - \hat{\mathrm{T}}) \otimes \phi(\mathrm{X})$
   - **Calculate residual**, "surprise" components of current investment and future revenue
4. $\theta(\mathrm{X}) = \langle \vec{a}, \phi(x) \rangle$ where $\vec{a}$ is the coefficient vector from the final regression

   Note: for θ constant, θ is the coefficient from $(\mathrm{Y} - \hat{\mathrm{Y}}) \sim (\mathrm{T} - \hat{\mathrm{T}})$
   - **Correlation** of these residuals identifies average causal effect of investments

EconML Solution

```python
# print data shape
print("Outcome shape: ", panelY.shape)
print("Treatment shape: ", panelT.shape)
print("Controls shape: ", panelX.shape)
```

```
Outcome shape:  (5000, 4)
Treatment shape:  (5000, 4, 3)
Controls shape:  (5000, 4, 89)
```

```python
# imports
from econml.dml import LinearDML
from sklearn.linear_model import LassoCV, MultiTaskLassoCV

# initiate DML estimator
est = LinearDML(
    model_y=LassoCV(max_iter=1000),  # nuisance model y
    model_t=MultiTaskLassoCV(max_iter=1000),  # nuiasnce model t
)
# fit treatment_0 on total revenue
est.fit(np.sum(panelY, axis=1), panelT[:, 0], X=None, W=panelX[:, 0])
# print treatment effect summary
est.summary(alpha=0.05)
```

```
Coefficient Results:  X is None, please call intercept_inference to learn the constant!
```

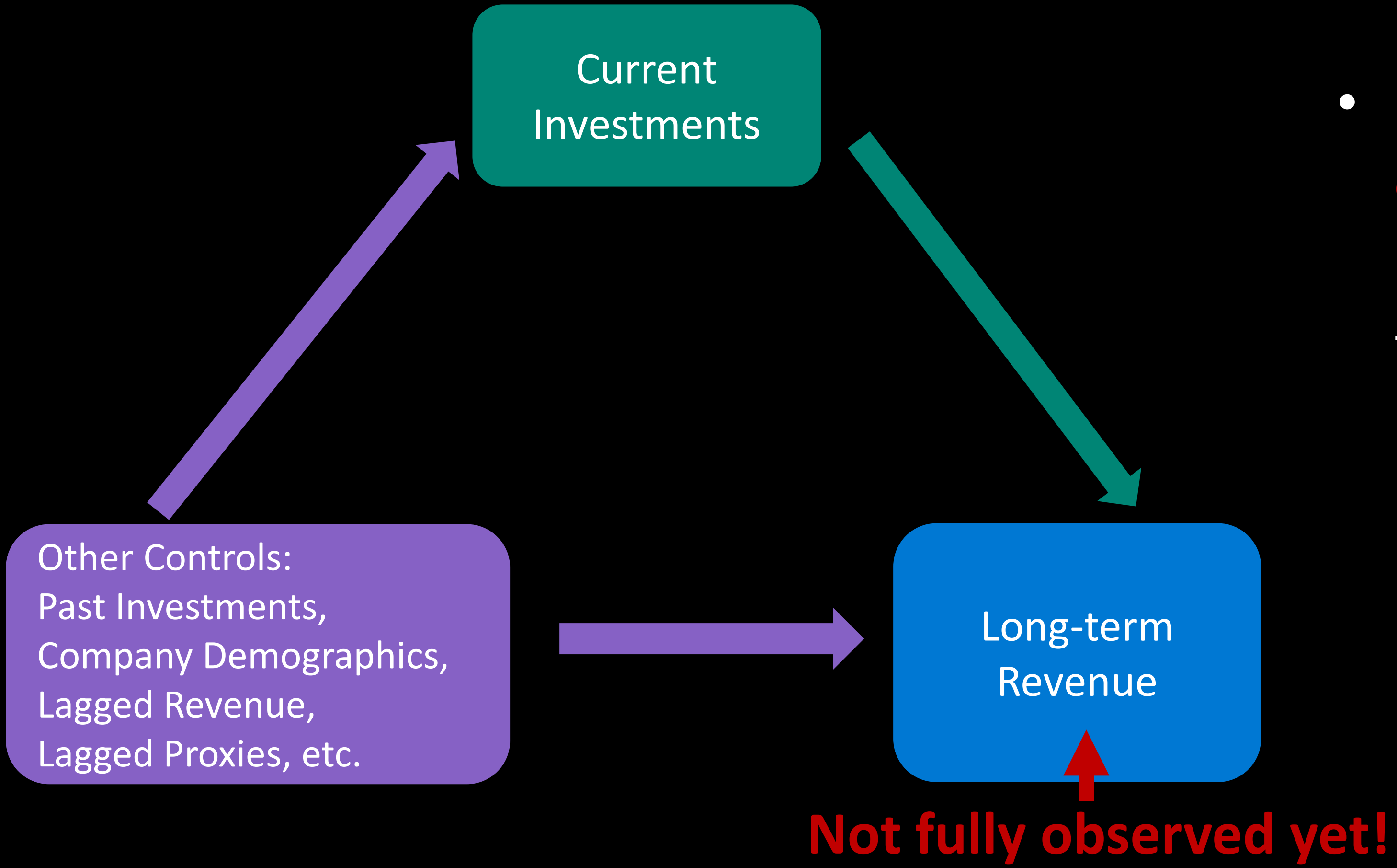
CATE Intercept Results

	point_estimate	stderr	zstat	pvalue	ci_lower	ci_upper
cate_intercept|T0	2.927	0.206	14.219	0.0	2.523	3.33
cate_intercept|T1	1.13	0.038	29.951	0.0	1.056	1.204
cate_intercept|T2	0.528	0.058	9.135	0.0	0.415	0.642

In this case, X= None. *cate_intercept* will be the constant effect θ we want to estimate!

Technical Challenge: Unobserved Outcome

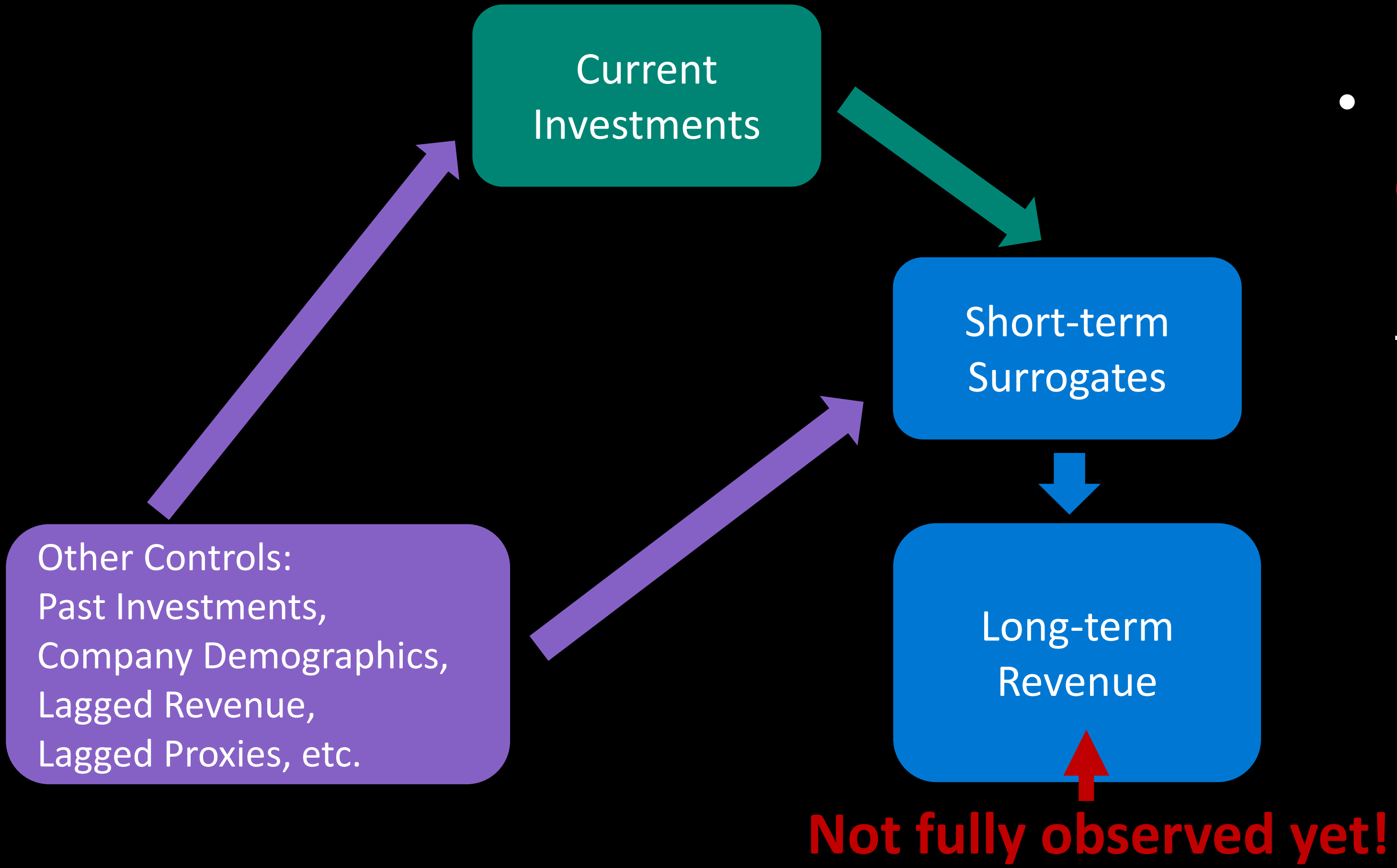
- Long-term revenue is not fully observed yet.



- Can we find **short-term observed surrogates** that are indicative of a customer's long-term revenue?

Technical Challenge: Unobserved Outcome

- Long-term revenue is not fully observed yet.

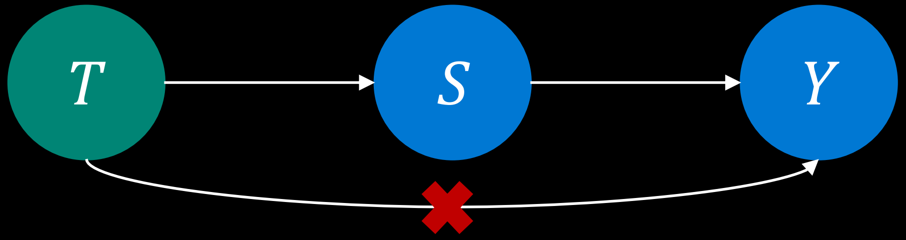


- Can we find **short-term observed surrogates** that are indicative of a customer's long-term revenue?

Methodology

- Estimation of Long-Term Effects with Surrogates

  (Prentice, 1989; Begg & Leung, 2000; Frangakis & Rubin, 2002; Freedman et al., 1992; Athey et al., 2020)

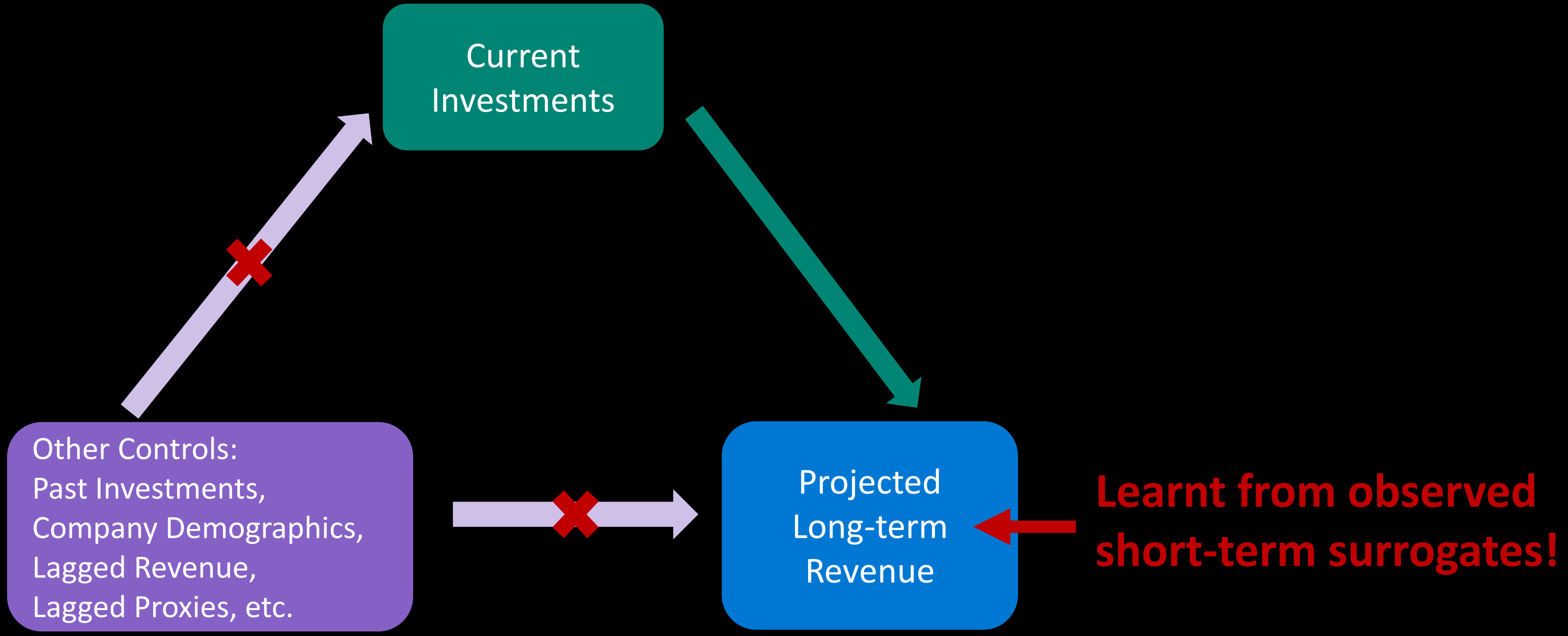


Assumption: The treatment effect of investments T on revenue Y could only go through surrogate S.

- Suppose we have a historical dataset (O) contains both long-term revenue Y and surrogates S, and a current dataset (E) contains only surrogates S.
- We can estimate $g(S) \coloneqq E[Y|S]$ from O by running a regression $Y \sim S$ using any ML models.
- Then we could predict the current long-term $\hat{Y} = g(S)$ from E.
- $\hat{Y}$ will be the projected long-term revenue, used as the outcome for DML model.

Updated Casual Graph

- Surrogates Index + DML



EconML Solution

Historical Data

	Company	Year	Features	Controls/Surrogates	T1	T2	T3	Revenue
1	A	2018	...	...	$1,000	...	...	$10,000
2	A	2019	...	...	$2,000	...	...	$12,000
3	A	2020	...	...	$3,000	...	...	$15,000
4	B	2018	...	...	$0	...	...	$5,000
5	B	2019	...	...	$100	...	...	$10,000
6	B	2020	...	...	$1,200	...	...	$7,000
7	C	2018	...	...	$1,000	...	...	$20,000
8	C	2019	...	...	$1,500	...	...	$25,000
9	C	2020	...	...	$500	...	...	$15,000

Current Data with New Investment

	Company	Year	Features	Controls/Surrogates	T1	T2	T3	SurrRev
1	A	2021	...	...	~~$1,000~~	...		$10,000
2	B	2021	...	...	~~$0~~	...		$5,000
3	C	2021	...	...	~~$2,000~~	...		$15,000

```python
# train surrogate index
XS_O = np.hstack([panelX_O[:, 0], panelY_O[:, :1]])  # concatenate controls and surrogates from historical dataset
XS_E = np.hstack([panelX_E[:, 0], panelY_E[:, :1]])  # concatenate controls and surrogates from current dataset
TotalY_O = np.sum(panelY_O, axis=1)  # total revenue from historical dataset
unadjusted_proxy_model = LassoCV().fit(XS_O, TotalY_O)  # train proxy model from historical dataset
sindex = unadjusted_proxy_model.predict(XS_E)  # predict current long term revenue
```

EconML Solution

Historical Data

	Company	Year	Features	Controls/Surrogates	T1	T2	T3	Revenue
1	A	2018	...	...	$1,000	...	...	$10,000
2	A	2019	...	...	$2,000	...	...	$12,000
3	A	2020	...	...	$3,000	...	...	$15,000
4	B	2018	...	...	$0	...	...	$5,000
5	B	2019	...	...	$100	...	...	$10,000
6	B	2020	...	...	$1,200	...	...	$7,000
7	C	2018	...	...	$1,000	...	...	$20,000
8	C	2019	...	...	$1,500	...	...	$25,000
9	C	2020	...	...	$500	...	...	$15,000

Current Data with New Investment

	Company	Year	Features	Controls/Surrogates	T1	T2	T3	SurrRev
1	A	2021	...	...	$1,000	...	...	$10,000
2	B	2021	...	...	$0	...	...	$5,000
3	C	2021	...	...	$2,000	...	...	$15,000

```python
# train surrogate index
XS_O = np.hstack([panelX_O[:, 0], panelY_O[:, :1]])  # concatenate controls and surrogates from historical dataset
XS_E = np.hstack([panelX_E[:, 0], panelY_E[:, :1]])  # concatenate controls and surrogates from current dataset
TotalY_O = np.sum(panelY_O, axis=1)  # total revenue from historical dataset
unadjusted_proxy_model = LassoCV().fit(XS_O, TotalY_O)  # train proxy model from historical dataset
sindex = unadjusted_proxy_model.predict(XS_E)  # predict current long term revenue
```

```python
# learn treatment effect on surrogate index
from econml.dml import LinearDML

est = LinearDML(model_y=LassoCV(max_iter=1000), model_t=MultiTaskLassoCV(max_iter=1000))
# fit treatment_0 on total revenue from current dataset
est.fit(sindex, panelT_E[:, 0], X=None, W=panelX_E[:, 0])
# print treatment effect summary
est.summary(alpha=0.05)
```

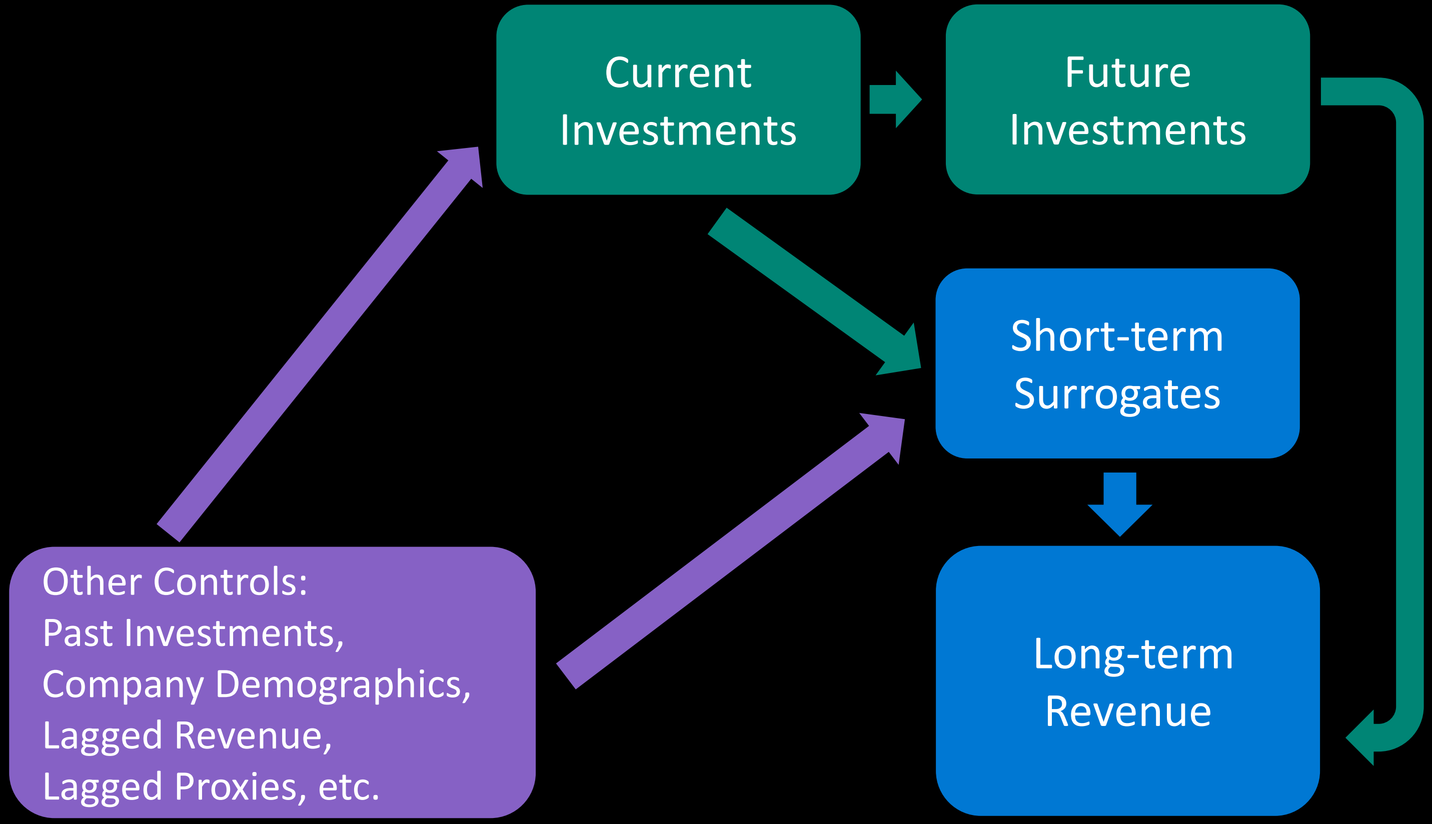
```
Coefficient Results:  X is None, please call intercept_inference to learn the constant!
```

CATE Intercept Results

	point_estimate	stderr	zstat	pvalue	ci_lower	ci_upper
cate_intercept|T0	1.415	0.03	46.428	0.0	1.356	1.475
cate_intercept|T1	0.662	0.018	37.064	0.0	0.627	0.697
cate_intercept|T2	0.261	0.006	44.417	0.0	0.249	0.272

Technical Challenge: Double Counting

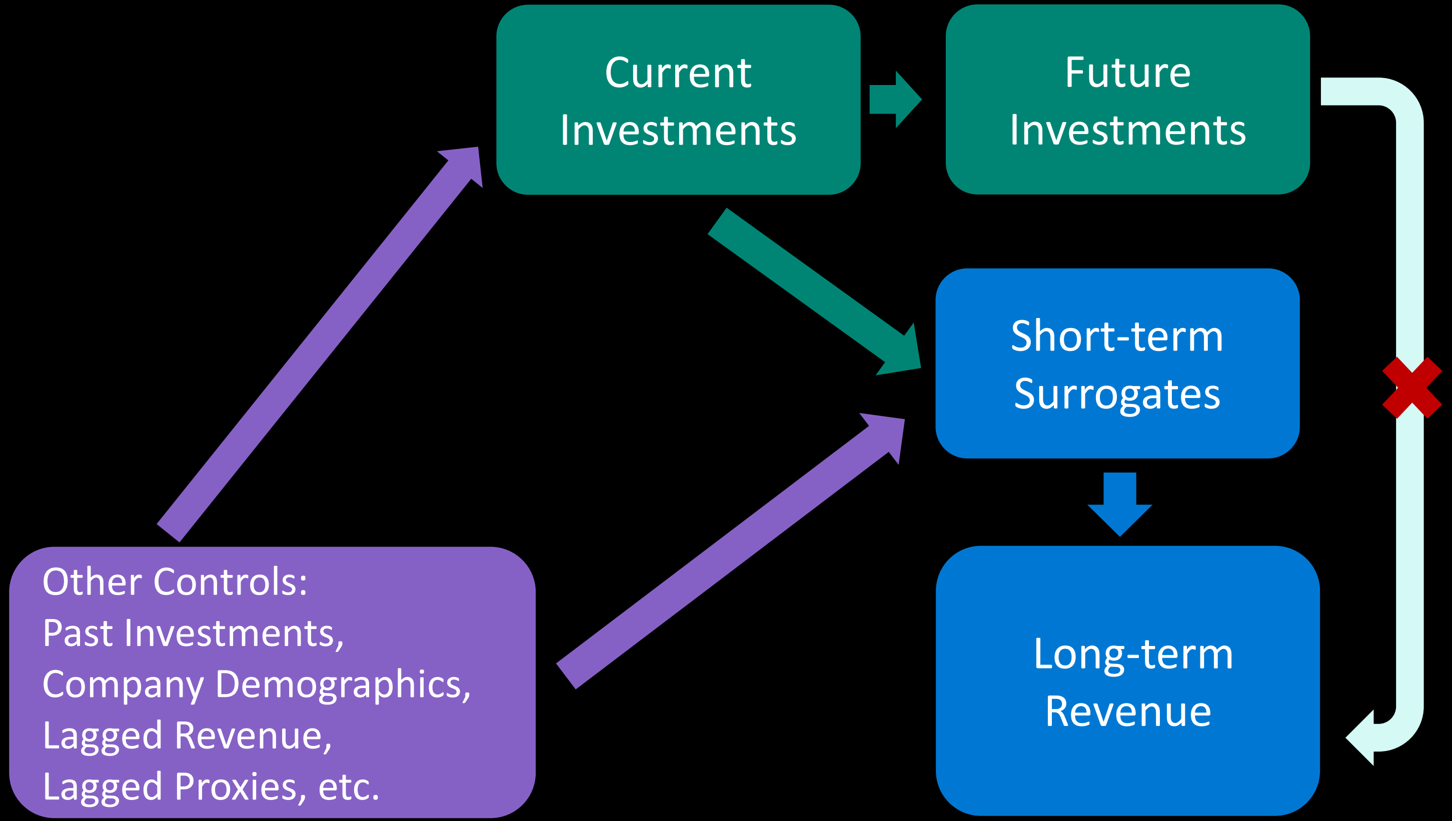
- Effect coming from future investments lead to **upwards biased effect**



- In the **historical dataset (O)**, long-term revenue includes the effect from future investment in the past, it will be **double attributed** to the current investments.

Methodology

- Double Machine Learning for dynamic effects (Lewis, Syrgkanis, “Double/Debiased Machine Learning for Dynamic Treatment Effects”)



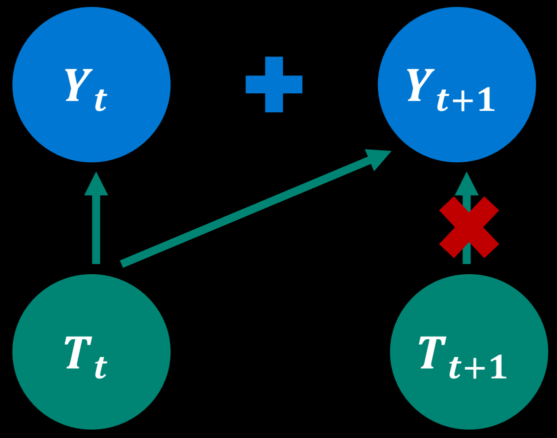
- **Dynamic treatment effect approach** removes the effects of future incentives from the **historical outcome** to create an adjusted long-term revenue as if those future incentives never happened.

Dynamic Double Machine Learning

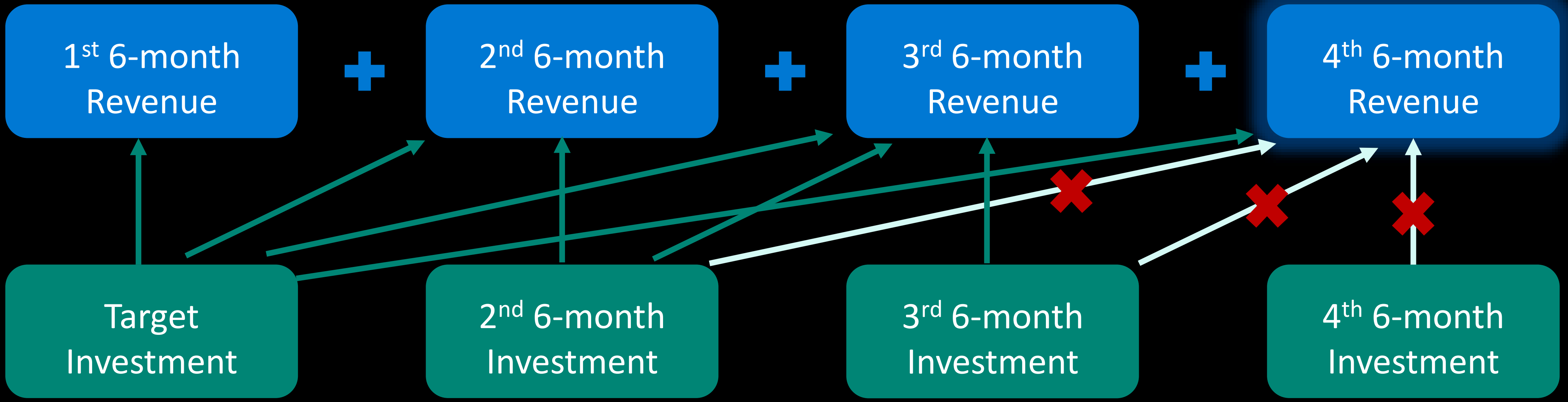
- Start with simple 2 periods:
  - Run DML to estimate effect $\boldsymbol{\theta_{t+1}}$ of T_{t+1} on Y_{t+1}
  - Subtract that effect to create the adjusted outcome

$$Y_{adj} = Y_t + (Y_{t+1} - \theta_{t+1} * T_{t+1})$$

- Use $\boldsymbol{Y_{adj}}$ as outcome to train surrogate model

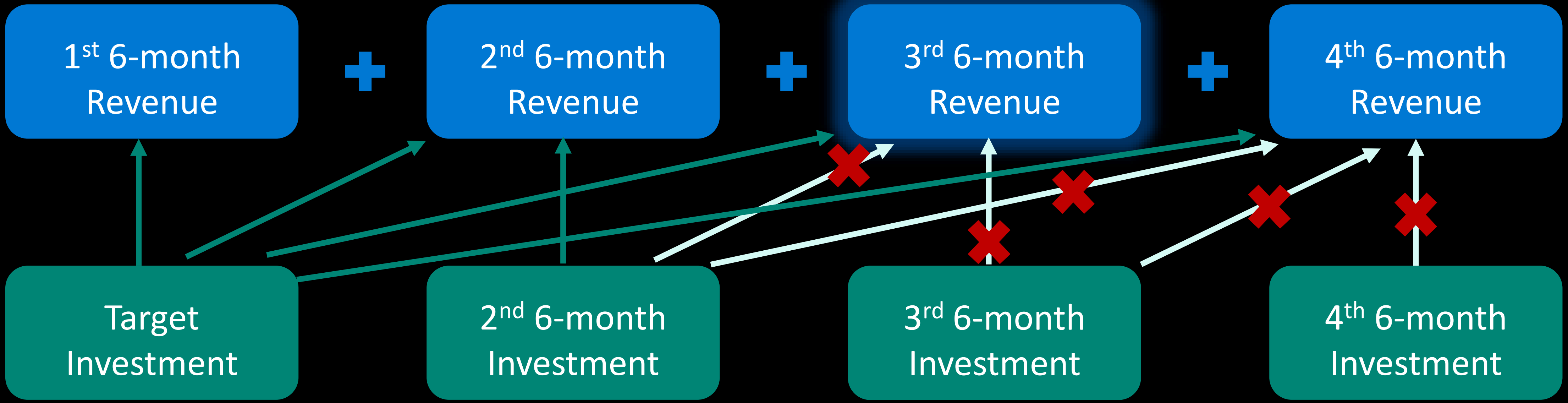


Dynamic DML Adjustment in ROI project



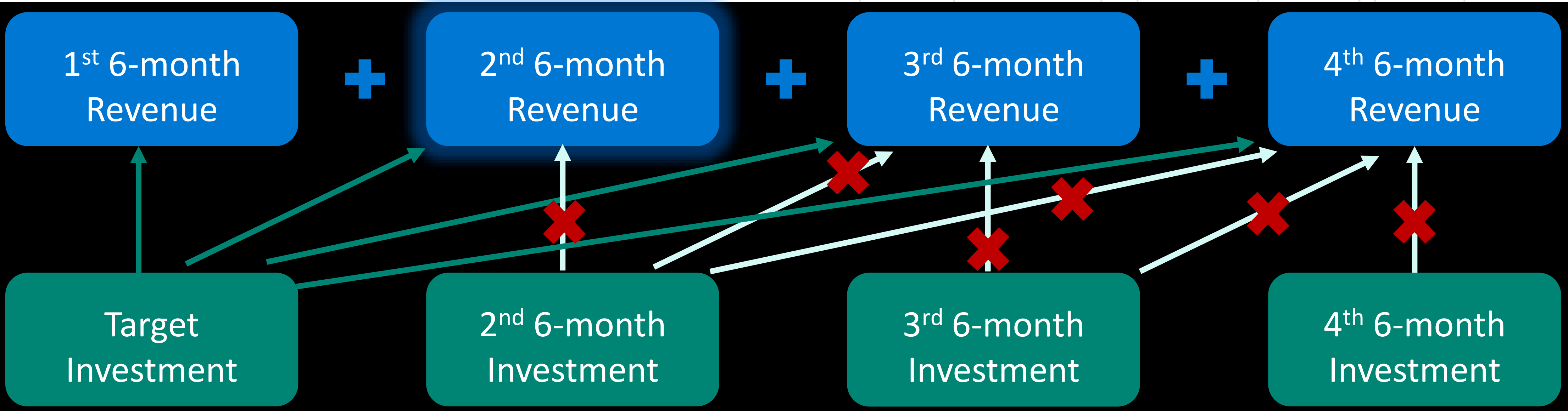
- From **historical dataset**, break the 2-year period into 6-month chunks
- **Recursively** subtract the effect of future investment from each revenue chunk
- **Sum up** the revenue chunks to get adjusted revenue, net of predicted effect of later investments
- Train surrogate index using **adjusted 2-year revenue**

Dynamic DML Adjustment in ROI project



- From **historical dataset**, break the 2-year period into 6-month chunks
- **Recursively** subtract the effect of future investment from each revenue chunk
- **Sum up** the revenue chunks to get adjusted revenue, net of predicted effect of later investments
- Train surrogate index using **adjusted 2-year revenue**

Dynamic DML Adjustment in ROI project



- From **historical dataset**, break the 2-year period into 6-month chunks
- **Recursively** subtract the effect of future investment from each revenue chunk
- **Sum up** the revenue chunks to get adjusted revenue, net of predicted effect of later investments
- Train surrogate index using **adjusted 2-year revenue**

EconML Solution for Dynamic DML

```python
# print data shape
print("Outcome shape: ", panelY.shape)
print("Treatment shape: ", panelT.shape)
print("Controls shape: ", panelX.shape)
print("Group ID shape: ", panelGroups.shape)
```

```
Outcome shape:  (5000, 4)
Treatment shape:  (5000, 4, 3)
Controls shape:  (5000, 4, 89)
Group ID shape:  (5000, 4)
```

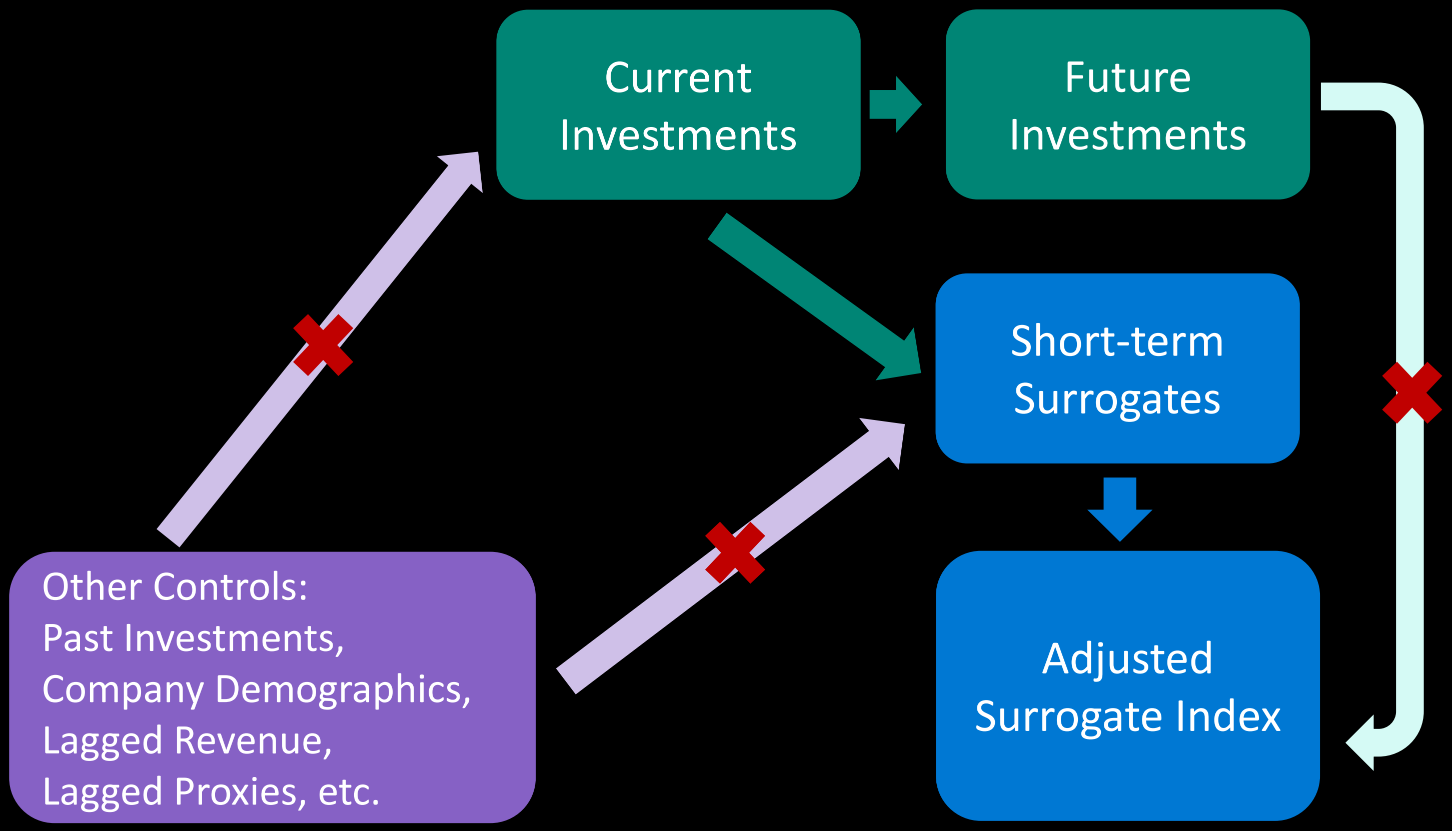
```python
# imports
from econml.dml import DynamicDML

# initiate dynamic DML estimator
est = DynamicDML(model_y=LassoCV(max_iter=1000), model_t=MultiTaskLassoCV(max_iter=1000))
# Learn period effect for each period T on last period revenue
est.fit(
    long(panelY),  # reshape the panel data n = n_groups * n_periods
    long(panelT),  # reshape the panel data n = n_groups * n_periods
    X=None,
    W=long(panelX),  # reshape the panel data n = n_groups * n_periods
    groups=long(panelGroups),  # reshape the panel data n = n_groups * n_periods
)
est.const_marginal_effect().reshape(-1, n_treatments)
```

```
array([[0.6897052 , 0.29836498, 0.1234395 ],
       [0.38710883, 0.1940497 , 0.09248946],
       [0.23237571, 0.0980073 , 0.03435173],
       [0.15665969, 0.05032955, 0.04140905]])
```

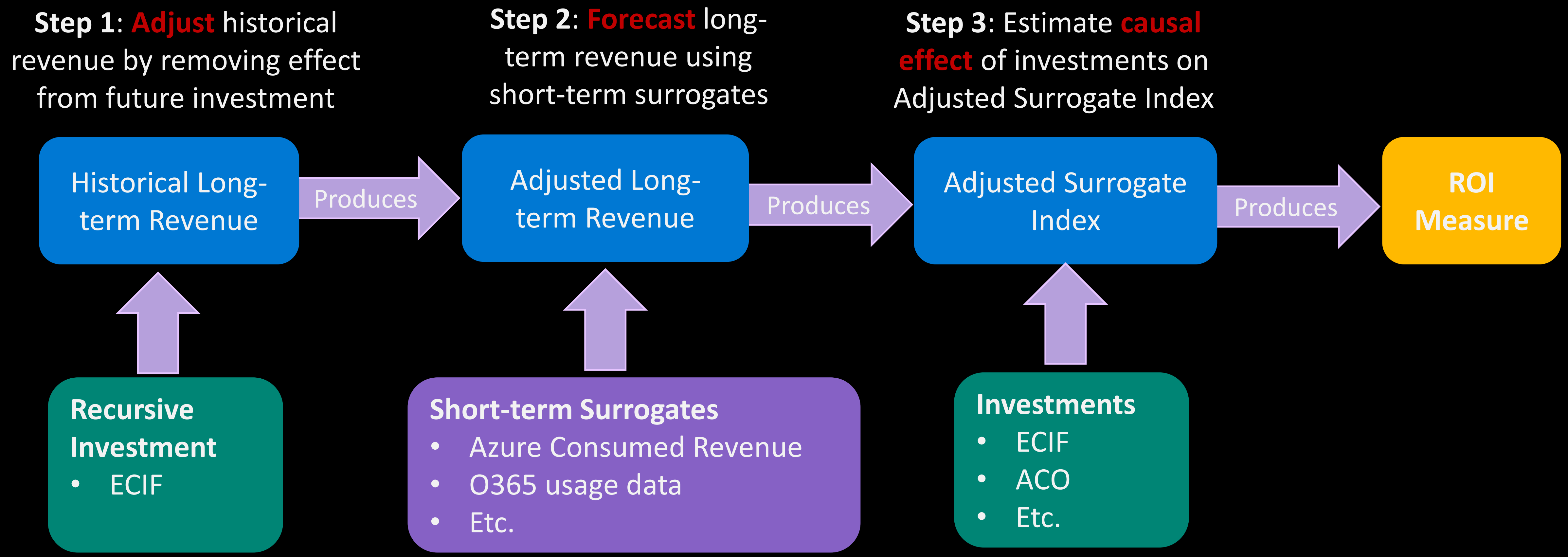
Return constant marginal effect for each period T on last period revenue for each treatment with shape (n_periods, d_t).

Updated Causal Graph

- Dynamic DML + Surrogate Index + DML



Unified Pipeline



EconML Solution

Historical Data

	Company	Year	Features	Controls/Surrogates	T1	T2	T3	Revenue
1	A	2018	...	...	$1,000	...	...	$10,000
2	A	2019	...	...	$2,000	...	...	$12,000
3	A	2020	...	...	$3,000	...	...	$15,000
4	B	2018	...	...	$0	...	...	$5,000
5	B	2019	...	...	$100	...	...	$10,000
6	B	2020	...	...	$1,200	...	...	$7,000
7	C	2018	...	...	$1,000	...	...	$20,000
8	C	2019	...	...	$1,500	...	...	$25,000
9	C	2020	...	...	$500	...	...	$15,000

Current Data with New Investment

	Company	Year	Features	Controls/Surrogates	T1	T2	T3	Revenue
1	A	2021	...	...	$1,000	...	...	$10,000
2	B	2021	...	...	$0	...	...	$5,000
3	C	2021	...	...	$2,000	...	...	$15,000

EconML Solution

Historical Data

	Company	Year	Features	Controls/Surrogates	T1	T2	T3	AdjRev
1	A	2018	...	...	$1,000	...	...	$10,000
2	A	2019	...	...	$2,000	...	...	$12,000
3	A	2020	...	...	$3,000	...	...	$15,000
4	B	2018	...	...	$0	...	...	$5,000
5	B	2019	...	...	$100	...	...	$10,000
6	B	2020	...	...	$1,200	...	...	$7,000
7	C	2018	...	...	$1,000	...	...	$20,000
8	C	2019	...	...	$1,500	...	...	$25,000
9	C	2020	...	...	$500	...	...	$15,000

Current Data with New Investment

	Company	Year	Features	Controls/Surrogates	T1	T2	T3	Revenue
1	A	2021	...	...	$1,000	...	...	$10,000
2	B	2021	...	...	$0	...	...	$5,000
3	C	2021	...	...	$2,000	...	...	$15,000

```python
# on historical data construct adjusted outcomes
from econml.dml import DynamicDML

est = DynamicDML(model_y=LassoCV(max_iter=1000), model_t=MultiTaskLassoCV(max_iter=1000))
for t in range(1, n_periods):  # for each target period 1...m
    est.fit(long(panelY_O[:, 1 : t + 1]), long(panelT_O[:, 1 : t + 1, :]), # reshape data to long format
            X=None, W=long(panelX_O[:, 1 : t + 1, :]), groups=long(panelGroups_O[:, 1 : t + 1]))
    # remove effect of observed treatments
    panelYadj_O[:, t] = panelY_O[:, t] - est.effect(T0=0, T1=wide(panelT_O[:, 1 : t + 1, :])) # reshape data to wide format
```

EconML Solution

Historical Data

	Company	Year	Features	Controls/Surrogates	T1	T2	T3	AdjRev
1	A	2018	...		$1,000	...	...	$10,000
2	A	2019	...	...	$2,000	...	...	$12,000
3	A	2020	...	...	$3,000	...	...	$15,000
4	B	2018	...		$0	...	...	$5,000
5	B	2019	...	...	$100	...	...	$10,000
6	B	2020	...	...	$1,200	...	...	$7,000
7	C	2018	...		$1,000	...	...	$20,000
8	C	2019	...	...	$1,500	...	...	$25,000
9	C	2020	...	...	$500	...	...	$15,000

Current Data with New Investment

	Company	Year	Features	Controls/Surrogates	T1	T2	T3	Revenue
1	A	2021	...	...	$1,000	...	...	$10,000
2	B	2021	...	...	$0	...	...	$5,000
3	C	2021	...	...	$2,000	...	...	$15,000

```
# on historical data construct adjusted outcomes
from econml.dml import DynamicDML

est = DynamicDML(model_y=LassoCV(max_iter=1000), model_t=MultiTaskLassoCV(max_iter=1000))
for t in range(1, n_periods):  # for each target period 1...m
    est.fit(long(panelY_O[:, 1 : t + 1]), long(panelT_O[:, 1 : t + 1, :]), # reshape data to long format
            X=None, W=long(panelX_O[:, 1 : t + 1, :]), groups=long(panelGroups_O[:, 1 : t + 1]))
    # remove effect of observed treatments
    panelYadj_O[:, t] = panelY_O[:, t] - est.effect(T0=0, T1=wide(panelT_O[:, 1 : t + 1, :])) # reshape data to wide format
```

```
# train surrogate index
XS_O = np.hstack([panelX_O[:, 0], panelYadj_O[:, :1]])  # concatenate controls and surrogates from historical dataset
TotalYadj_O = np.sum(panelYadj_O, axis=1)  # total revenue from historical dataset
adjusted_proxy_model = LassoCV().fit(XS_O, TotalYadj_O)  # train proxy model from historical dataset
```

EconML Solution

Historical Data

	Company	Year	Features	Controls/Surrogates	T1	T2	T3	AdjRev
1	A	2018	...	...	$1,000	...	...	$10,000
2	A	2019	...	...	$2,000	...	...	$12,000
3	A	2020	...	...	$3,000	...	...	$15,000
4	B	2018	...	...	$0	...	...	$5,000
5	B	2019	...	...	$100	...	...	$10,000
6	B	2020	...	...	$1,200	...	...	$7,000
7	C	2018	...	...	$1,000	...	...	$20,000
8	C	2019	...	...	$1,500	...	...	$25,000
9	C	2020	...	...	$500	...	...	$15,000

Current Data with New Investment

	Company	Year	Features	Controls/Surrogates	T1	T2	T3	AdjSurrRev
1	A	2021	...	...	$1,000	...	...	$10,000
2	B	2021	...	...	$0	...	...	$5,000
3	C	2021	...	...	$2,000	...	...	$15,000

```python
# on historical data construct adjusted outcomes
from econml.dml import DynamicDML

est = DynamicDML(model_y=LassoCV(max_iter=1000), model_t=MultiTaskLassoCV(max_iter=1000))
for t in range(1, n_periods):  # for each target period 1...m
    est.fit(long(panelY_O[:, 1 : t + 1]), long(panelT_O[:, 1 : t + 1, :]), # reshape data to long format
            X=None, W=long(panelX_O[:, 1 : t + 1, :]), groups=long(panelGroups_O[:, 1 : t + 1]))
    # remove effect of observed treatments
    panelYadj_O[:, t] = panelY_O[:, t] - est.effect(T0=0, T1=wide(panelT_O[:, 1 : t + 1, :])) # reshape data to wide format
```

```python
# train surrogate index
XS_O = np.hstack([panelX_O[:, 0], panelYadj_O[:, :1]])  # concatenate controls and surrogates from historical dataset
TotalYadj_O = np.sum(panelYadj_O, axis=1)  # total revenue from historical dataset
adjusted_proxy_model = LassoCV().fit(XS_O, TotalYadj_O)  # train proxy model from historical dataset
```

```python
# predict current long term revenue
XS_E = np.hstack([panelX_E[:, 0], panelY_E[:, :1]])  # concatenate controls and surrogates from current dataset
sindex_adj = adjusted_proxy_model.predict(XS_E)
```

EconML Solution

Historical Data

	Company	Year	Features	Controls/Surrogates	T1	T2	T3	AdjRev
1	A	2018	...	...	$1,000	...	...	$10,000
2	A	2019	...	...	$2,000	...	...	$12,000
3	A	2020	...	...	$3,000	...	...	$15,000
4	B	2018	...	...	$0	...	...	$5,000
5	B	2019	...	...	$100	...	...	$10,000
6	B	2020	...	...	$1,200	...	...	$7,000
7	C	2018	...	...	$1,000	...	...	$20,000
8	C	2019	...	...	$1,500	...	...	$25,000
9	C	2020	...	...	$500	...	...	$15,000

```python
# on historical data construct adjusted outcomes
from econml.dml import DynamicDML

est = DynamicDML(model_y=LassoCV(max_iter=1000), model_t=MultiTaskLassoCV(max_iter=1000))
for t in range(1, n_periods):  # for each target period 1...m
    est.fit(long(panelY_O[:, 1 : t + 1]), long(panelT_O[:, 1 : t + 1, :]), # reshape data to long format
            X=None, W=long(panelX_O[:, 1 : t + 1, :]), groups=long(panelGroups_O[:, 1 : t + 1]))
    # remove effect of observed treatments
    panelYadj_O[:, t] = panelY_O[:, t] - est.effect(T0=0, T1=wide(panelT_O[:, 1 : t + 1, :])) # reshape data to wide format
```

```python
# train surrogate index
XS_O = np.hstack([panelX_O[:, 0], panelYadj_O[:, :1]])  # concatenate controls and surrogates from historical dataset
TotalYadj_O = np.sum(panelYadj_O, axis=1)  # total revenue from historical dataset
adjusted_proxy_model = LassoCV().fit(XS_O, TotalYadj_O)  # train proxy model from historical dataset
```

Current Data with New Investment

	Company	Year	Features	Controls/Surrogates	T1	T2	T3	AdjSurrRev
1	A	2021	...	...	$1,000	...	...	$10,000
2	B	2021	...	...	$0	...	...	$5,000
3	C	2021	...	...	$2,000	...	...	$15,000

```python
# predict current long term revenue
XS_E = np.hstack([panelX_E[:, 0], panelY_E[:, :1]])  # concatenate controls and surrogates from current dataset
sindex_adj = adjusted_proxy_model.predict(XS_E)
```

CATE Intercept Results

	point_estimate	stderr	zstat	pvalue	ci_lower	ci_upper
cate_intercept|T0	1.284	0.024	52.626	0.0	1.236	1.332
cate_intercept|T1	0.603	0.011	53.221	0.0	0.581	0.626
cate_intercept|T2	0.242	0.005	46.863	0.0	0.232	0.253

```python
# learn treatment effect on surrogate index
from econml.dml import LinearDML
est = LinearDML(model_y=LassoCV(max_iter=1000), model_t=MultiTaskLassoCV(max_iter=1000))
# fit treatment_0 on total revenue from current dataset
est.fit(sindex_adj, panelT_E[:, 0], X=None, W=panelX_E[:, 0])
# print treatment effect summary
est.summary(alpha=0.05)
```

Wrap Up

- **Surrogate index approach** is a widely useful technique for measuring **long-term** effects from **short-term** data (healthcare, business, marketing)
- Typical assumptions for the method to work can **be severely violated** if historical data contain other treatments
- Adaptivity and auto-correlation of the historical treatment policy can severely **bias effects**
- These biases can be corrected by combining techniques from **estimation of treatment effects in the dynamic treatment regime with the surrogate approach**
- If careful in the estimation method (Neyman orthogonal moments + sample splitting), **Machine learning** can be used to enable estimation with **high-dimensional** surrogates or controls

Applying EconML to Your Causal Problem

- Jupyter notebooks for similar use cases:
  - https://github.com/microsoft/EconML/tree/master/notebooks/CustomerScenarios
- Learn more about all the uses of EconML on our website
  - https://aka.ms/econml
- Github and Doc link:
  - https://github.com/Microsoft/EconML
  - https://econml.azurewebsites.net/
- Get started with EconML in Python: `pip install econml`

Reference

[1] Susan Athey, Raj Chetty, Guido Imbens, and Hyunseung Kang. Estimating treatment effectsusing multiple surrogates: The role of the surrogate score and the surrogate index, 2020.

[2] Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen,Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters. The Econometrics Journal, 21(1):C1–C68, 01 2018.

[3] Greg Lewis and Vasilis Syrgkanis. Double/debiased machine learning for dynamic treatment effects, 2020.

[4] Keith Battocchi, Eleanor Dillon, Maggie Hei, Greg Lewis, Miruna Oprescu, Vasilis Syrgkanis. Estimating the long-term effects of novel treatments, 2021.

[5] Daniel Yehdego, Jane Huang, Saurabh Kumar, and Siddharth Kumar. An application ofcausal modeling for azure investment attribution, 2020.

QA