nference-propensity-score-matching

November 10, 2025

1 Step 03: Causal Inference & Uplift Modeling

Project: AI Experimentation Platform for Predictive Insights
Notebook: 03_Causal_Inference_and_Uplift_Modeling
Author: Jagadeesh N
Date: 10-Nov-2025
Environment: Python (DoWhy · EconML · Scikit-learn · Matplotlib · Seaborn)

1.0.1 Objective

This notebook estimates the **true causal impact** of being in the *treatment group (B)* on customer conversion and revenue.
We will move beyond correlation and apply **causal inference** and **uplift modeling** to measure how much the experimental treatment *caused* behavior change.

1.0.2 Background

A/B testing shows performance differences between two groups, but not *why* those differences occur. Causal inference methods (like **DoWhy** and **EconML**) help estimate **treatment effects** while controlling for confounders (e.g., spending, number of transactions, high-value behavior).

1.0.3 Notebook Outline

1. Load simulation dataset & baseline model
2. Estimate Average Treatment Effect (ATE) using DoWhy
3. Conduct causal graph-based validation
4. Apply EconML's Double ML method for uplift estimation
5. Visualize treatment effect distribution
6. Interpret business insights

[1]:
```
#importing
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split

sns.set(style="whitegrid")
np.random.seed(42)

# Load simulated dataset
data = pd.read_csv("D:\\Project for job\\ab_test_simulated.csv")
data.head()
```

[1]:

	Customer ID	Revenue	NumTransactions	TotalUnits	Converted	Group
0	12346.0	93843.3166	12	74285	0	A
1	12347.0	5955.0513	8	2967	1	B
2	12348.0	2221.3400	5	2714	0	A
3	12349.0	5358.7149	4	1624	0	B
4	12350.0	367.8400	1	197	0	A

[2]:
```
# Treatment: being in group B (1)
data['Treatment'] = (data['Group'] == 'B').astype(int)

# Outcome: whether the customer converted
outcome = 'Converted'

# Confounders (features that influence both treatment and outcome)
confounders = [
    'TotalUnits', 'NumTransactions', 'Revenue',
    'RevenuePerUnit', 'RevenuePerTransaction', 'HighValueCustomer'
]

print("Treatment:", "Treatment")
print("Outcome:", outcome)
print("Confounders:", confounders)
```

```
Treatment: Treatment
Outcome: Converted
Confounders: ['TotalUnits', 'NumTransactions', 'Revenue', 'RevenuePerUnit',
'RevenuePerTransaction', 'HighValueCustomer']
```

[5]:
```
print(data.columns.tolist())
```

```
['Customer ID', 'Revenue', 'NumTransactions', 'TotalUnits', 'Converted',
'Group', 'Treatment']
```

[8]:
```
# Create binary treatment variable
data['treatment'] = (data['Group'] == 'B').astype(int)
```

[9]:
```
data[['Group', 'treatment']].head()
```

[9]:

	Group	treatment
0	A	0
1	B	1
2	A	0
3	B	1
4	A	0

[10]:
```
from dowhy import CausalModel

treatment = 'treatment'              # just created above
outcome = 'Converted'                # your binary outcome
confounders = [col for col in data.columns if col not in ['Group', 'Converted',␣
 ↪'Revenue', 'treatment']]
```

[12]:
```
model = CausalModel(
    data=data,
    treatment=treatment,
    outcome=outcome,
    common_causes=confounders
)

identified_estimand = model.identify_effect()
estimate = model.estimate_effect(
    identified_estimand,
    method_name="backdoor.linear_regression"
)

print(" Estimated Average Treatment Effect (ATE):", estimate.value)

refutation = model.refute_estimate(
    identified_estimand,
    estimate,
    method_name="placebo_treatment_refuter"
)
print(refutation)
```

```
 Estimated Average Treatment Effect (ATE): 0.009893893320552238
Refute: Use a Placebo Treatment
Estimated effect:0.009893893320552238
New effect:-0.0008547437230948847
p value:0.88
```

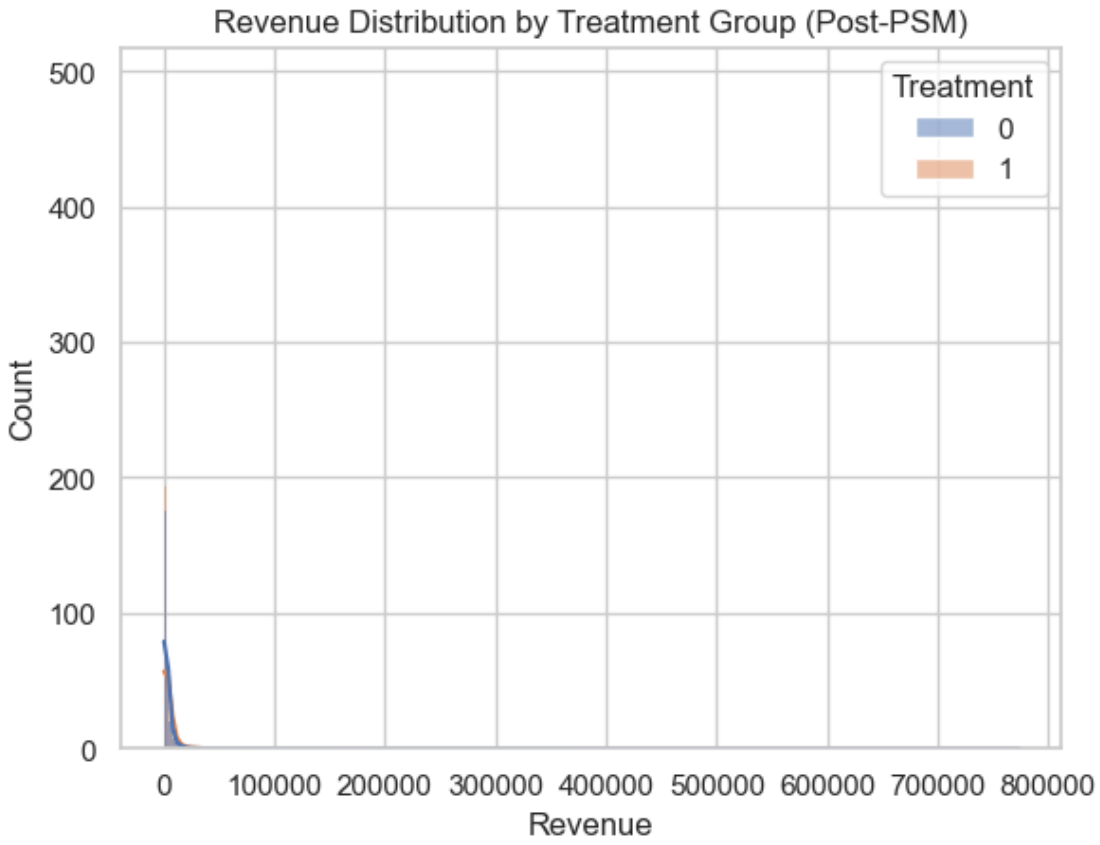
[15]:
```
from sklearn.linear_model import LogisticRegression
estimate = model.estimate_effect(
    identified_estimand,
    method_name="backdoor.propensity_score_matching",
    method_params={"propensity_score_model": LogisticRegression(max_iter=1000)}
)
```

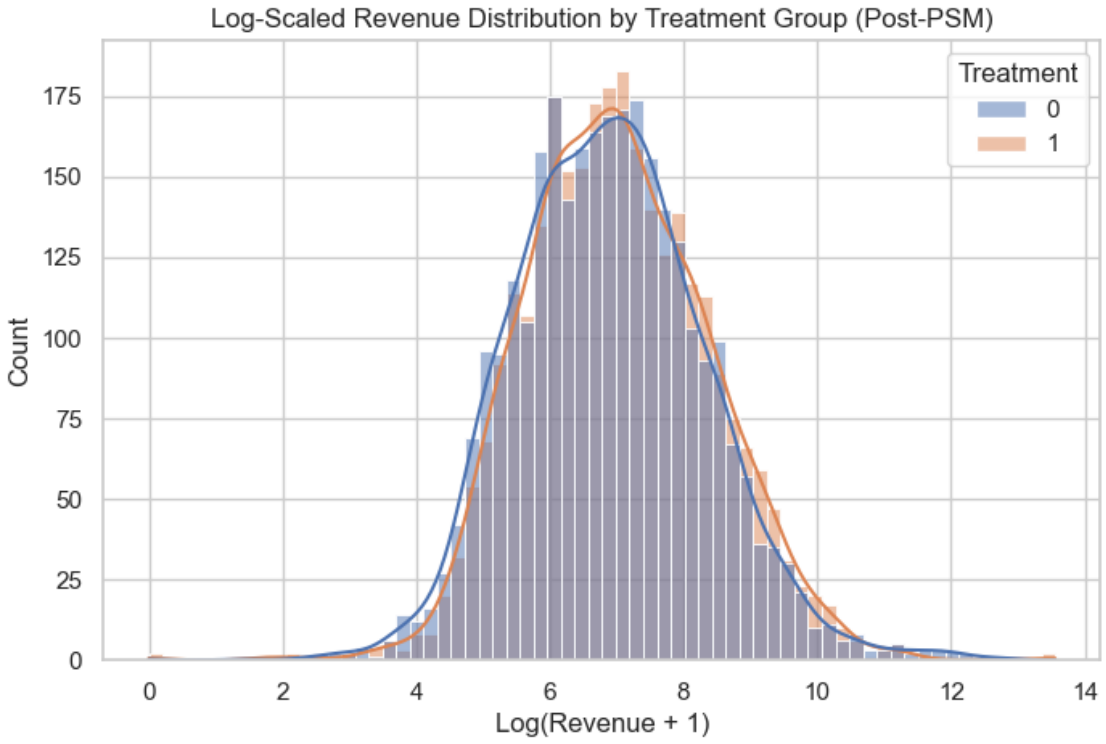
[18]:
```
#Refute the estimate
refute = model.refute_estimate(
    identified_estimand,
    estimate,
    method_name="placebo_treatment_refuter"
)
print(refute)
```

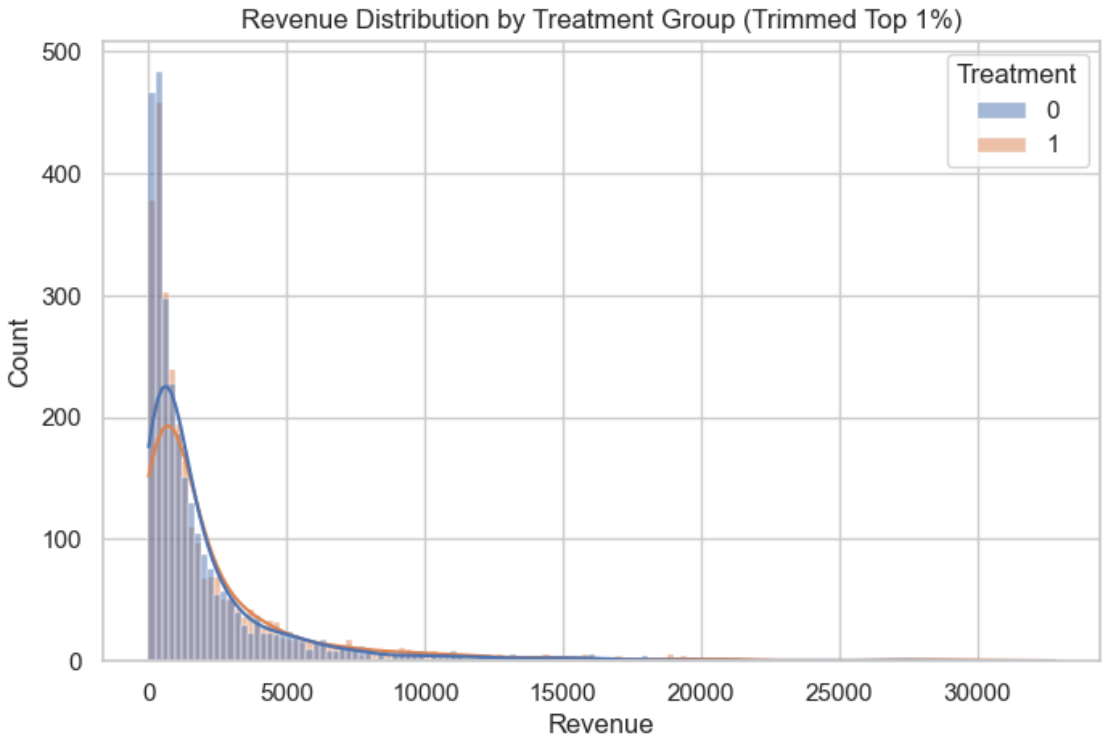
```
Refute: Use a Placebo Treatment
Estimated effect:0.009352150994728798
New effect:0.00021765005951368806
p value:0.8999999999999999
```

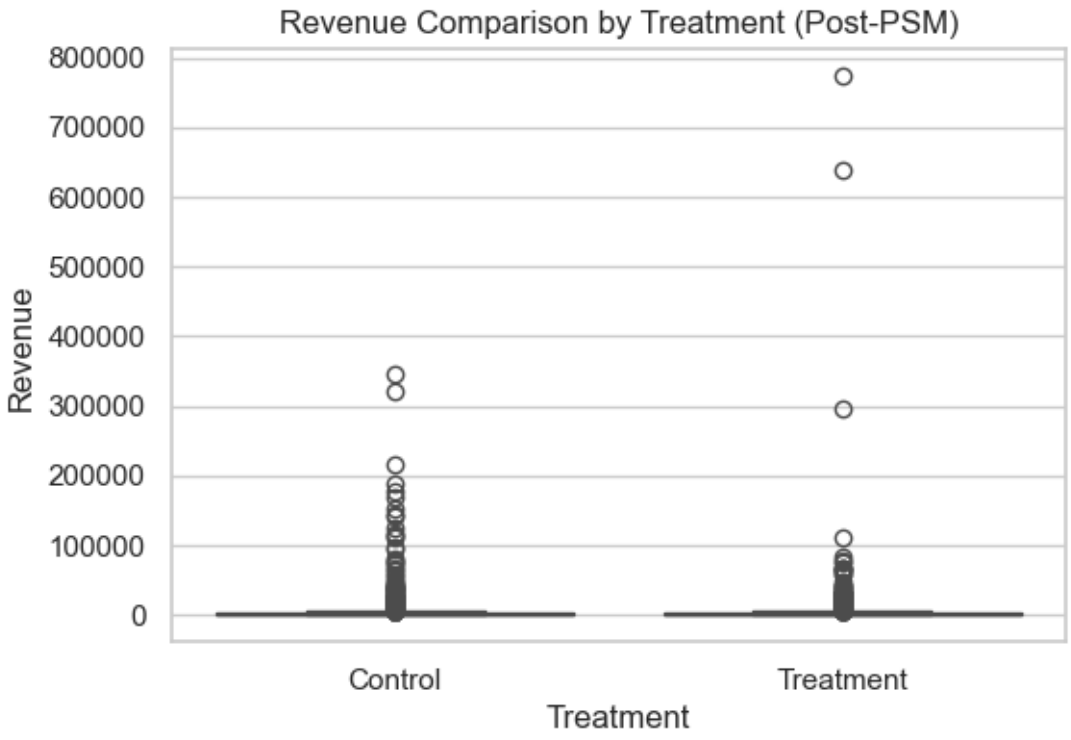
[20]:
```
#visualizing
import matplotlib.pyplot as plt
sns.histplot(data=data, x='Revenue', hue='Treatment', kde=True)
plt.title("Revenue Distribution by Treatment Group (Post-PSM)")
plt.show()
plt.figure(figsize=(8,5))
sns.histplot(
    data=data,
    x=np.log1p(data['Revenue']),  # log(Revenue + 1)
    hue='Treatment',
    kde=True
)
plt.title("Log-Scaled Revenue Distribution by Treatment Group (Post-PSM)")
plt.xlabel("Log(Revenue + 1)")
plt.ylabel("Count")
plt.show()
plt.figure(figsize=(8,5))
sns.histplot(
    data=data[data['Revenue'] < data['Revenue'].quantile(0.99)],
    x='Revenue',
    hue='Treatment',
    kde=True
)
plt.title("Revenue Distribution by Treatment Group (Trimmed Top 1%)")
plt.show()
plt.figure(figsize=(6,4))
```

```
sns.boxplot(data=data, x='Treatment', y='Revenue')
plt.title("Revenue Comparison by Treatment (Post-PSM)")
plt.xticks([0, 1], ['Control', 'Treatment'])
plt.show()
```




Log-Scaled Revenue Distribution by Treatment Group (Post-PSM)
Treatment
0
1
Count
0
25
50
75
100
125
150
175
0
2
4
6
8
10
12
14
Log(Revenue + 1)


Revenue Distribution by Treatment Group (Trimmed Top 1%)
Treatment
0
1
Count
0
100
200
300
400
500
0
5000
10000
15000
20000
25000
30000
Revenue



[23]:
```
original_data = pd.read_csv("D:\\Project for job\\ab_test_simulated.csv")
# AFTER PSM dataset (your current matched data after causal modeling)
matched_data = data.copy()   # 'data' is the one used after PSM in your model
```

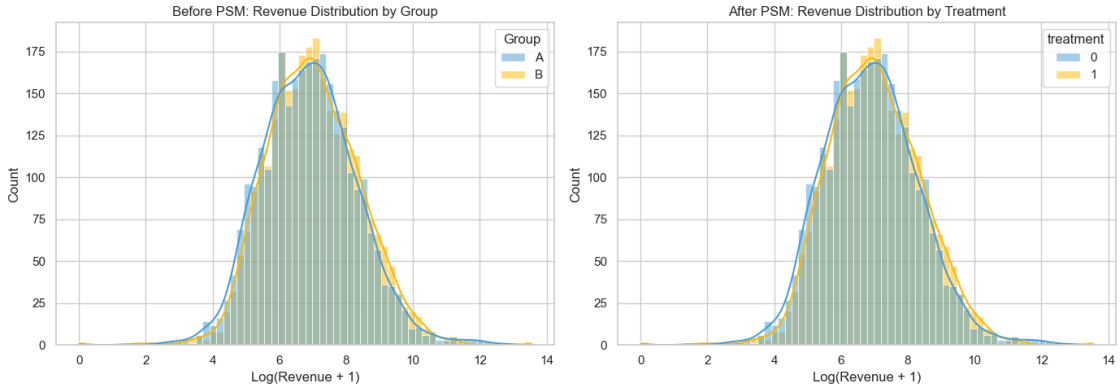
[26]:
```
print(data.columns.tolist())
```

```
['Customer ID', 'Revenue', 'NumTransactions', 'TotalUnits', 'Converted',
'Group', 'Treatment', 'treatment', 'propensity_score']
```

[28]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd

# Load original data (before matching)
original_data = pd.read_csv("D:\\Project for job\\ab_test_simulated.csv")

# matched_data is your current 'data' (after PSM)
matched_data = data.copy()

plt.figure(figsize=(14, 5))
fig, axes = plt.subplots(1, 2, figsize=(14, 5))
```

```
#   BEFORE PSM
sns.histplot(
    data=original_data,
    x=np.log1p(original_data['Revenue']),
    hue='Group',
    kde=True,
    ax=axes[0],
    palette=['#4B9CD3', '#FFB703']
)
axes[0].set_title("Before PSM: Revenue Distribution by Group")
axes[0].set_xlabel("Log(Revenue + 1)")
axes[0].set_ylabel("Count")

#   AFTER PSM
sns.histplot(
    data=matched_data,
    x=np.log1p(matched_data['Revenue']),
    hue='treatment',
    kde=True,
    ax=axes[1],
    palette=['#4B9CD3', '#FFB703']
)
axes[1].set_title("After PSM: Revenue Distribution by Treatment")
axes[1].set_xlabel("Log(Revenue + 1)")
axes[1].set_ylabel("Count")

plt.tight_layout()
plt.show()
```

```
<Figure size 1400x500 with 0 Axes>
```



[31]:
```
#Save the matched dataset for future steps
matched_data.to_csv("D:\\Project for job\\ab_test_matched.csv", index=False)
```

```
print("Matched dataset saved successfully at: D:/Project for job/data/processed/
  ↪ab_test_matched.csv")
```

```
Matched dataset saved successfully at: D:/Project for
job/data/processed/ab_test_matched.csv
```