step4

November 10, 2025

0.1 Step 04: Uplift Modeling & Causal Validation

0.1.1 Objective

After estimating the Average Treatment Effect (ATE) using Propensity Score Matching (PSM), this step aims to **quantify the incremental impact** of the treatment on customer behavior — specifically, conversion rates and revenue generation.
While ATE provides an average effect across all users, **uplift modeling** identifies *which customers respond positively* to the treatment, providing actionable insights for targeting and personalization.

0.1.2 Methodology

We implement the **Two-Model Approach** for uplift prediction:

- **Model 1:** Predicts conversion probability for the **treatment group**
- **Model 2:** Predicts conversion probability for the **control group**
- **Uplift = P(Treatment Conversion) − P(Control Conversion)**

This approach helps isolate the **true incremental effect** of the treatment for each customer.

0.1.3 Why This Matters

Traditional machine learning models predict *likelihood of conversion*, but uplift models predict *change in likelihood due to intervention.*
This distinction is critical in data-driven business experiments such as:

- Marketing campaign optimization
- A/B test impact analysis
- Customer retention and churn prevention
- Product feature rollout evaluation

By integrating uplift modeling after causal inference, we bridge **statistical causality** and **predictive decision-making**, aligning perfectly with real-world data science applications.

0.1.4 Key Deliverables

- Uplift model training using Random Forests
- Distribution visualization of predicted uplift
- Quantification of incremental conversion and revenue lift
- Model artifacts saved for downstream deployment

0.1.5 Expected Outcome

A reliable estimate of how the treatment group differs from the control group at the **individual level**, empowering targeted interventions and smarter business decisions.

[4]:
```
import pandas as pd

# Load your post-PSM data (update the file path if needed)
matched_data = pd.read_csv("D:\\Project for job\\ab_test_matched.csv")

print("Matched data loaded successfully")
print(matched_data.shape)
print(matched_data.head())
```

```
Matched data loaded successfully
(5881, 9)
   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

   Treatment  treatment  propensity_score
0          0          0          0.014692
1          1          1          0.996333
2          0          0          0.004432
3          1          1          0.996380
4          0          0          0.004396
```

[8]:
```
#Prepare the Data for Uplift Modeling
from sklearn.model_selection import train_test_split

# We'll predict uplift in 'Converted' (conversion probability)
uplift_data = matched_data.copy()
```

```
# Ensure the right column names
uplift_data = uplift_data.rename(columns={'treatment': 'TreatmentFlag'})

# Features and target
features = ['Revenue', 'NumTransactions', 'TotalUnits', 'propensity_score']
target = 'Converted'
treatment_col = 'TreatmentFlag'

X_train, X_test, y_train, y_test, t_train, t_test = train_test_split(
    uplift_data[features],
    uplift_data[target],
    uplift_data[treatment_col],
    test_size=0.3,
    random_state=42
)
print("Done")
```

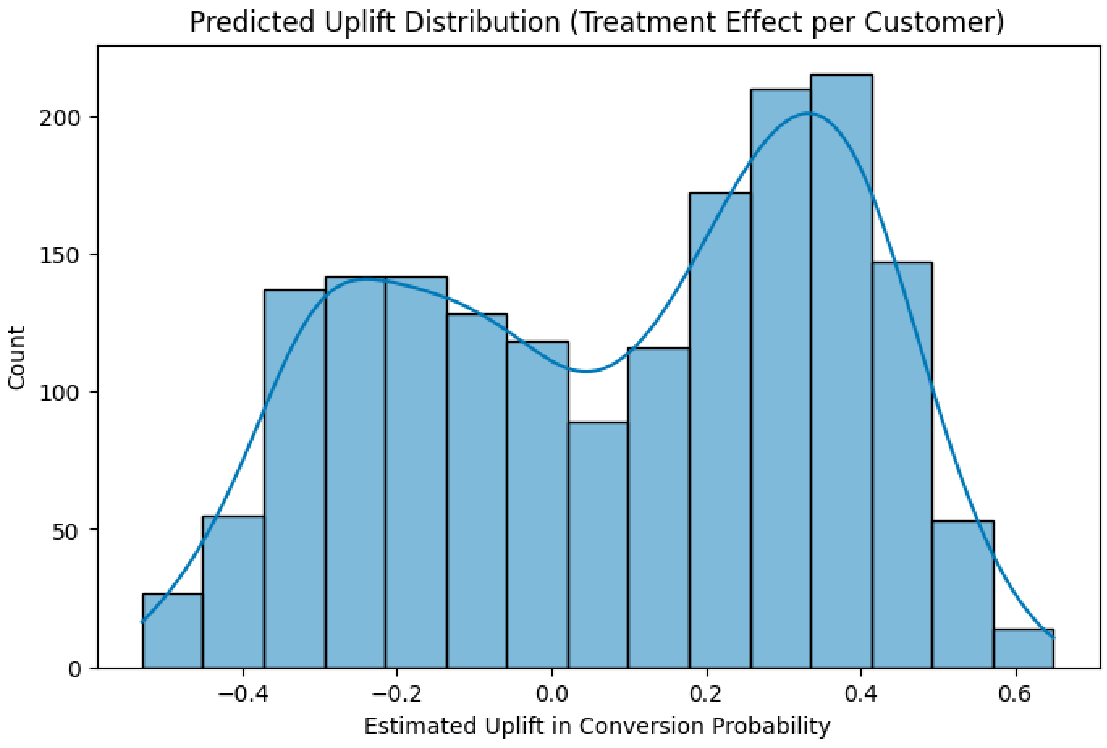
```
Done
```

[9]:
```
#Train Uplift Model (Two-Model Approach)
from sklearn.ensemble import RandomForestClassifier

# Train separate models
rf_treat = RandomForestClassifier(n_estimators=100, random_state=42)
rf_ctrl = RandomForestClassifier(n_estimators=100, random_state=42)

rf_treat.fit(X_train[t_train==1], y_train[t_train==1])
rf_ctrl.fit(X_train[t_train==0], y_train[t_train==0])

# Predict probabilities
p_treat = rf_treat.predict_proba(X_test)[:,1]
p_ctrl = rf_ctrl.predict_proba(X_test)[:,1]

uplift = p_treat - p_ctrl
uplift_df = pd.DataFrame({
    'Uplift': uplift,
    'Treatment': t_test,
    'Converted': y_test
})
uplift_df.head()
```

[9]:
```
      Uplift  Treatment  Converted
5371   -0.28          1          1
5299    0.27          0          0
199     0.36          0          0
3268   -0.04          1          0
```

```
3504   -0.27          1          0
```

[12]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Visualization
plt.figure(figsize=(8,5))
sns.histplot(uplift_df['Uplift'], kde=True, color='#0077b6')
plt.title("Predicted Uplift Distribution (Treatment Effect per Customer)")
plt.xlabel("Estimated Uplift in Conversion Probability")
plt.ylabel("Count")
plt.show()
```



[13]:
```
avg_uplift = uplift_df['Uplift'].mean()
treatment_effect = uplift_df[uplift_df['Treatment']==1]['Converted'].mean() -␣
  ↪uplift_df[uplift_df['Treatment']==0]['Converted'].mean()

print(f"Average Predicted Uplift: {avg_uplift:.4f}")
print(f"Observed Conversion Lift (Treatment vs Control): {treatment_effect:.
  ↪4f}")
```

```
Average Predicted Uplift: 0.0824
Observed Conversion Lift (Treatment vs Control): 0.0141
```

[14]:
```python
import joblib

joblib.dump(rf_treat, "D:/Project for job/models/uplift_treatment_model.pkl")
joblib.dump(rf_ctrl, "D:/Project for job/models/uplift_control_model.pkl")

print("Uplift models saved successfully")
```

```
Uplift models saved successfully
```

[ ]:
```

```