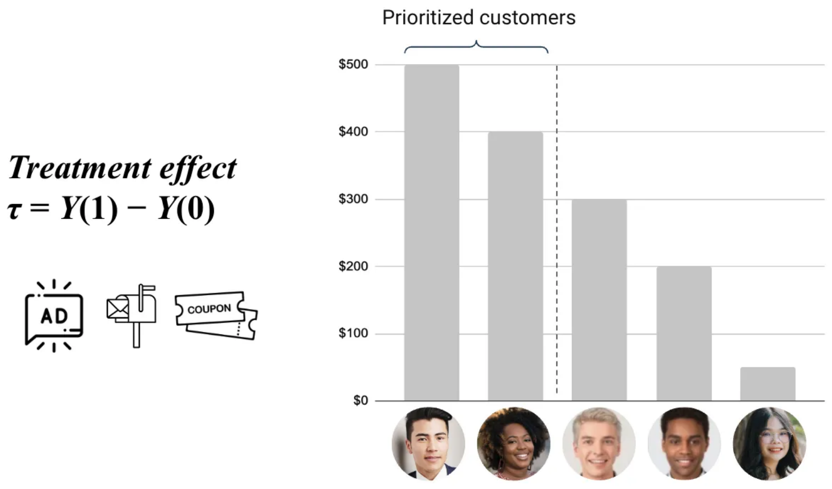
Uplift Modelling: Advanced Customer Targeting with Causal ML

In recent years, algorithms combining causal inference and machine learning have been a hot topic. In this article, we will introduce **Uplift Modelling,** a powerful technique for advanced customer targeting. We will explore its objectives, underlying concepts and key algorithms followed by a case study using CausalML library.

1. Who Should We Target To Maximize Sales?

Uplift modelling helps identify customers who are more likely to make a purchase when targeted by a specific treatment - such as a marketing offer - compared to when they are not targeted. This approach is particularly valuable in cost-intensive marketing campaigns, where efficient allocation of resources is critical.



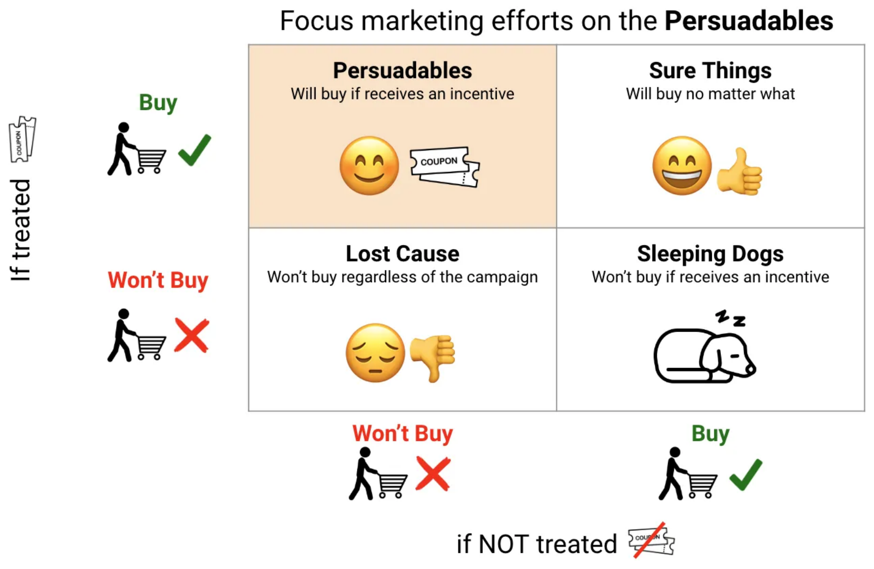
Now, let's take a closer look at the customers to be targeted.

2. Understanding The Customers To Target

To apply Uplift Modeling, we can segment customers into four types:

1. **Persuadable:** Customers who are likely to buy a product only if they receive treatment. These are primary targets for uplift modeling.
2. **Sure Things:** Customers who will purchase regardless of whether they are targeted.
3. **Lost Causes:** Customers who will not buy the product, even if they are targeted.
4. **Sleeping Dogs:** There are rare customers who are less likely to purchase if they receive treatment.

The key point is that marketing resources, such as coupons, should not be wasted on Sure Things, Lost Causes, or Sleeping Dogs, as these groups are unlikely to yield a positive return on investment (ROI).


Focus marketing efforts on the Persuadables
Persuadables
Will buy if receives an incentive
COUPON
Sure Things
Will buy no matter what
Lost Cause
Won't buy regardless of the campaign
Sleeping Dogs
Won't buy if receives an incentive
If treated
COUPON
Buy
Won't Buy
Won't Buy
Buy
if NOT treated
COUPON

3. Uplift Tree Algorithm

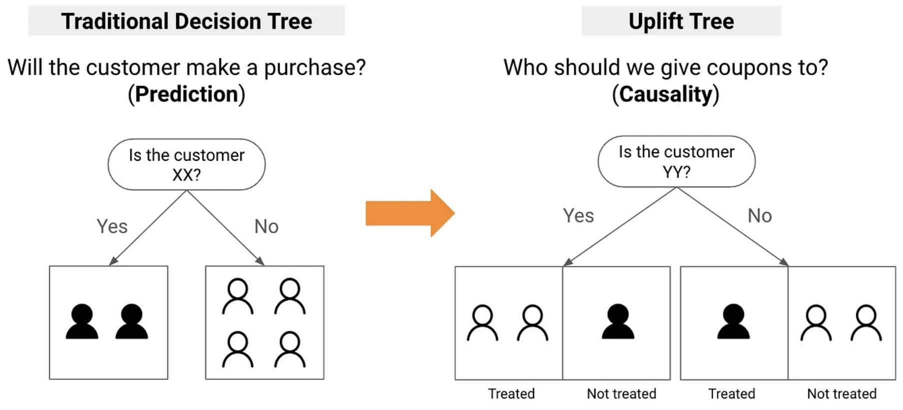
There are primarily two ways to conduct uplift modeling.

1. **Meta Learners:** This method, involves sorting customers by the estimated treatment effect in descending order to prioritize whom to target.
2. **Tree-Based Methods:** This approach uses decision trees specifically designed for uplift modeling, such as `Uplift Trees, and Uplift Random Forests` . The advantage of these models is their ability to handle multiple treatment groups simultaneously. For example, instead of testing a single discount, you can compare the effectiveness of different discounts - like 5%, 10% or 15% - to determine which is the most effective for each customer segment. This flexibility make tree-based methods highly practical for real-world data.

3.1 From Traditional Decision Trees to Uplift Tree:

To understand the concept of a **uplift tree,** let's compare it with a traditional decision tree. A traditional decision tree aims to predict outcomes, such as whether a customer will make a purchase. In contrast, an uplift tree answers causal questions like, "Who should we target with coupons".

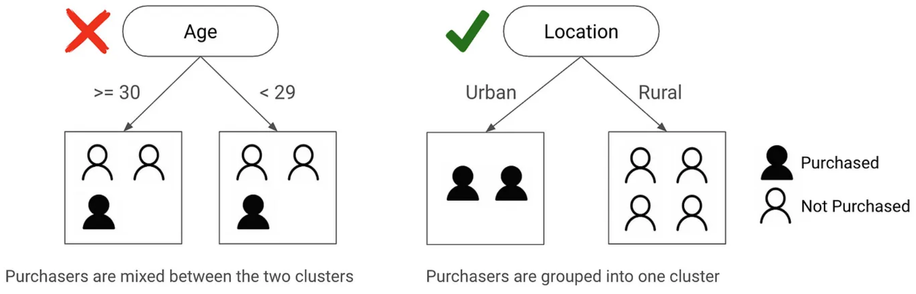
The uplift tree differentiates between treated and untreated customers, allowing us to identify the potential uplift generated by providing coupon. Now, let's deep dive into the details of each method.



3.2 Normal Decision Trees:

Depending on how the nodes for trees are chosen, there are multiple ways to construct a decision tree. But how do we choose the best nodes to construct a more effective decision tree.

Consider the two trees in the graph below: the tree on the left splits by "Age," while the tree on the right splits by "Location." The black icons indicate customers who made a purchase, while the white icons represent those who did not.


Age
>= 30
< 29
Location
Urban
Rural
Purchased
Not Purchased
Purchasers are mixed between the two clusters
Purchasers are grouped into one cluster

Which tree provides a better split?

The answer is the tree on the right. In the left tree, purchasers and non-purchasers are mixed within both clusters. Conversely, in the right tree, purchasers and non-purchasers are clearly separated. But how can we quantify this judgment?

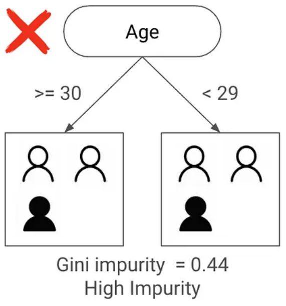
For splitting criteria, `"Gini Impurity"` can be used.

To evaluate how effective each split is, we use a metric called "Gini Impurity". This measure helps us quantify the purity of nodes in a decision tree. Gini impurity is defined by the following formula:

$$GiniImpurity = 1—(p_1^2 + p_0^2)$$

where p_1 is the probability of purchase, and p_0 is the probability of no purchase.

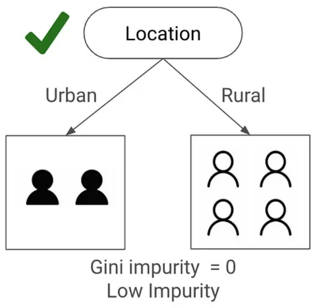
For example, in the first decision tree shown below, the Gini Impurity is calculated to be 0.44.



Weighted Gini Impurity for the split:

$$
\begin{aligned}
&= \frac{3}{6} \cdot Gini_{\text{left}} + \frac{3}{6} \cdot Gini_{\text{right}} \\
&= \frac{3}{6} \cdot \left(1 - \left(\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2\right)\right) + \frac{3}{6} \cdot \left(1 - \left(\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2\right)\right) \\
&= \frac{4}{9} \\
&\approx 0.44
\end{aligned}
$$

In contrast, the Gini Impurity for the second decision tree is 0, which indicates a more efficient tree structure. This difference suggests that using "Location" as the decision node results in a better classification, as it more effectively separates purchasers from non-purchasers.

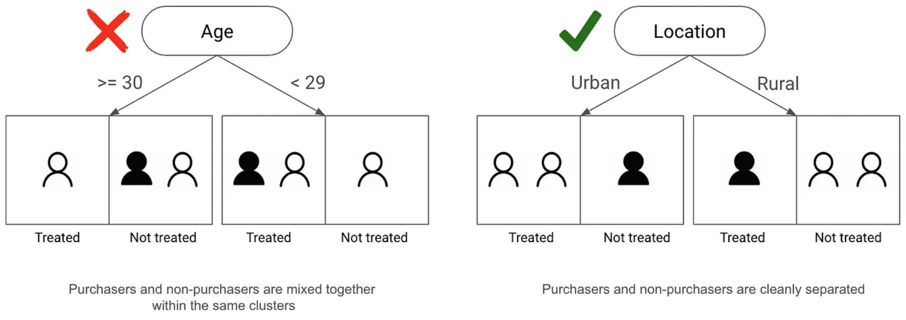


$$\begin{aligned}&\text{Weighted Gini Impurity for the split:}\\ &= \frac{2}{6}\cdot \text{Gini}_{\text{left}} + \frac{4}{6}\cdot \text{Gini}_{\text{right}}\\ &= \frac{2}{6}\cdot\left(1-\left(\left(\frac{2}{2}\right)^2+\left(\frac{0}{2}\right)^2\right)\right)+\frac{4}{6}\cdot\left(1-\left(\left(\frac{0}{4}\right)^2+\left(\frac{4}{4}\right)^2\right)\right)\\ &= \frac{2}{6}\cdot 0+\frac{4}{6}\cdot 0\\ &= 0\end{aligned}$$

3.3 Uplift Trees:

Now, let's explore uplift trees. The primary objective of an uplift tree is to determine, "Who should we give coupons to?" In other words, the goal is to maximize the difference in purchase rates between customers who receive a treatment (like a coupon) and those who do not. To achieve this, we analyze the treated and non-treated groups separately.

Consider the two uplift trees shown below. As before, the tree on the right offers a better split because it more effectively separates purchasers from non-purchasers.


Age
>= 30
< 29
Treated
Not treated
Treated
Not treated
Purchasers and non-purchasers are mixed together within the same clusters
Location
Urban
Rural
Treated
Not treated
Treated
Not treated
Purchasers and non-purchasers are cleanly separated

But how can we quantify this difference? To do this, we turn to metrics from information theory.

For split criteria, divergence metrics can be used

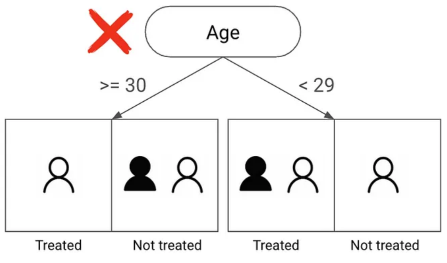
To measure the quality of the split, we can use divergence metrics such as Kullback-Leibler (KL) Divergence or Euclidean Distance. Here, we'll use the Squared Euclidean Distance. It's defined by the following formula:

Squared Euclidean Distance = $(P(0) — Q(0))^2 + P(1) — Q(1))^2$

In this formula:

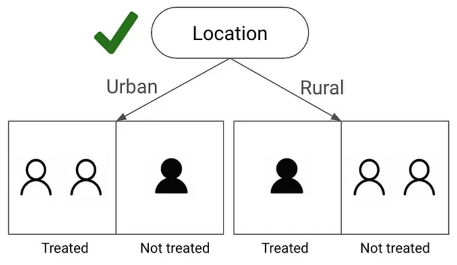
- P(1) and P(0) represent the probabilities of purchase and no purchase in the treatment group, respectively.
- Q(1) and Q(0) represent the probabilities of purchase and no purchase in the control group, respectively.

According to this metric, the first uplift tree shown below has a divergence of 1.



$$\begin{aligned}&\text{Squared Euclidean distance of split}\\ &= \text{Distance of left} + \text{Distance of right}\\ &= (P_{\text{left}}(1)-Q_{\text{left}}(1))^2+(P_{\text{left}}(0)-Q_{\text{left}}(0))^2\\ &\quad+(P_{\text{right}}(1)-Q_{\text{right}}(1))^2+(P_{\text{right}}(0)-Q_{\text{right}}(0))^2\\ &= (0-\frac{1}{2})^2+(1-\frac{1}{2})^2\\ &\quad+(\frac{1}{2}-0)^2+(\frac{1}{2}-1)^2\\ &= 1\end{aligned}$$

On the other hand, the divergence for the second decision tree is 0. This indicates that the second tree achieves a larger divergence, meaning it maximizes the uplift more effectively.



$$\begin{aligned}
&\text{Squared Euclidean distance of split} \\
&= \text{Distance of left} + \text{Distance of right} \\
&= (P_{\text{left}}(1) - Q_{\text{left}}(1))^2 + (P_{\text{left}}(0) - Q_{\text{left}}(0))^2 \\
&\quad + (P_{\text{right}}(1) - Q_{\text{right}}(1))^2 + (P_{\text{right}}(0) - Q_{\text{right}}(0))^2 \\
&= (0-1)^2 + (1-0)^2 \\
&\quad + (1-0)^2 + (0-1)^2 \\
&= 4
\end{aligned}$$

4. Sample Code

There are two popular libraries for Causal Machine Learning. For uplift modeling specifically, I suggest using **CausalML**, which offers a range of useful functions tailored to various use cases in this field.

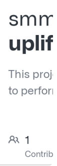
Library	Features	GitHub
CausalML	• Focus on Uplift modeling and Meta Learners • Designed as a standalone tool • Uber	uber/causalml (4.8k star)
EconML	• Covers a wide range of algorithms, strong in economics • Part of a bigger DoWhy ecosystem • Microsoft Research	py-why/EconML (3.6k star)

You can access the full code here:

GitHub - smmaisam/uplift_modelling_using_causalML: This project leverages CausalML library from Uber to perform Uplift Modelling.

This project leverages CausalML library from Uber to perform Uplift Modelling. - smmaisam/uplift_modelling_using_causalML

https://github.com/smmaisam/uplift_modelling_using_causalML


smm
uplif
This proj
to perfor
1
Contrib

4.1 Dataset:

This sample code demonstrates how to perform uplift modeling using the Criteo dataset, a well-known dataset from the advertising industry. The goal of this case study is to prioritize customers for digital ad targeting.

The dataset contains user data with 11 features, a treatment label, and a conversion label, representing whether a user saw an advertisement and whether they converted (made a purchase) afterward.

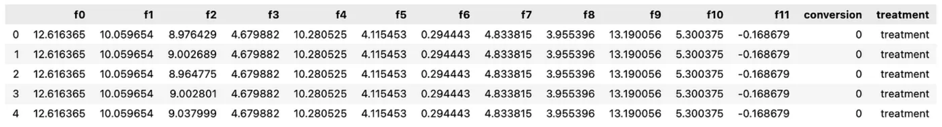
4.2 Importing Necessary Libraries:

To get started, you will need to install two libraries: `CausalML` for uplift modeling and `scikit-uplift` to download the Criteo dataset.

```
pip install causalml scikit-uplift

# Import necessary libraries
import numpy as np
import pandas as pd
from causalml.inference.tree import UpliftRandomForestClassifier
from causalml.metrics import plot_gain, plot_qini
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
import importlib
from IPython.display import display

# Ensure the required libraries are installed
print(importlib.metadata.version('causalml'))

from sklift.datasets import fetch_criteo
```

The raw data includes 11 user features, a treatment label, and a conversion label:

```
X, y, treatment = fetch_criteo(target_col='conversion', treatment_col='treatment', return_X_y_
t=True)

df = X.copy()
df['conversion'] = y.astype('int64')
df['treatment'] = treatment.astype('object').replace({0: 'control', 1: 'treatment'})
# df['user_id'] = df.index

df.head()
```



	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	conversion	treatment
0	12.616365	10.059654	8.976429	4.679882	10.280525	4.115453	0.294443	4.833815	3.955396	13.190056	5.300375	-0.168679	0	treatment
1	12.616365	10.059654	9.002689	4.679882	10.280525	4.115453	0.294443	4.833815	3.955396	13.190056	5.300375	-0.168679	0	treatment
2	12.616365	10.059654	8.964775	4.679882	10.280525	4.115453	0.294443	4.833815	3.955396	13.190056	5.300375	-0.168679	0	treatment
3	12.616365	10.059654	9.002801	4.679882	10.280525	4.115453	0.294443	4.833815	3.955396	13.190056	5.300375	-0.168679	0	treatment
4	12.616365	10.059654	9.037999	4.679882	10.280525	4.115453	0.294443	4.833815	3.955396	13.190056	5.300375	-0.168679	0	treatment

4.3 Dataset Overview:

The Criteo dataset consists of over 13 million rows and is significantly imbalanced: approximately 85% of users were exposed to advertisements (treated), while 15% were not (control group).

```
print('Total number of samples: {}'.format(len(df)))
```

> **Total number of samples: 13979592**

```
df['treatment'].value_counts(normalize = True)
```

> **treatment 0.85**
> **control 0.15**

```
df['conversion'].value_counts(normalize = True)
```

> **0: 0.997083**
> **1: 0.002917**

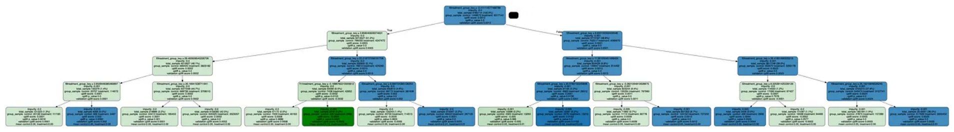
4.4 Training The Uplift Model:

After splitting the data into training and testing sets, we use the Uplift Random Forest to train the model. The `CausalML` library provides an interface similar to other machine learning libraries like `scikit-learn`.

```
# Split data into training and testing sets
X_train, X_test, y_train, y_test, treatment_train, treatment_test = train_test_split(
    df.drop(columns = ['conversion', 'treatment']), df['conversion'], df['treatment'],
    test_size=0.3, random_state=42)

# Train Uplift Random Forest model
uplift_rf = UpliftRandomForestClassifier(control_name='control')
uplift_rf.fit(X_train.values, treatment=treatment_train.values, y=y_train.values)

# Predict uplift using the trained model
y_pred = uplift_rf.predict(X_test)
```

`CausalML` provides tools for visualizing the structure of the uplift tree:

```
# Plot uplift tree
from IPython.display import Image
graph = uplift_tree_plot(uplift_tree.fitted_uplift_tree, X_train.columns)
Image(graph.create_png())
```



4.5 Feature Importance:

We can also examine the feature importance to understand which variables significantly impact the model’s predictions.

```
# Plotting the feature importance of the uplift tree
pd.Series(uplift_tree.feature_importances_, index = X_train.columns).sort_values().plot(kind
='barh', figsize=(12,8))
```

4.6 Evaluation:

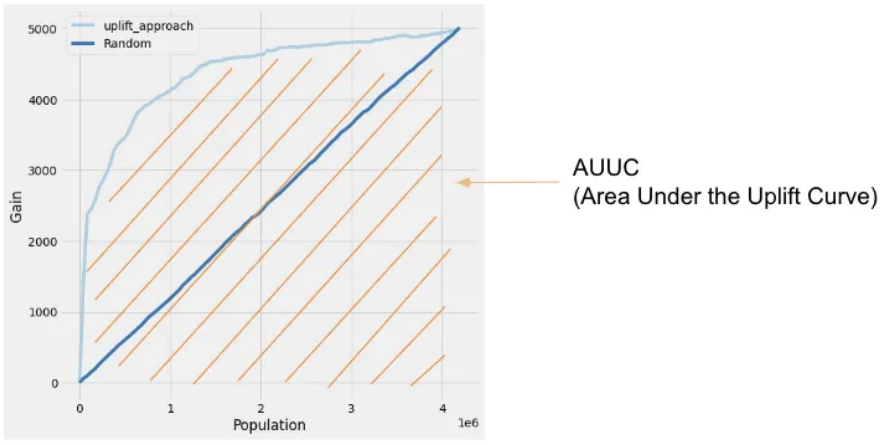
To prepare for evaluation, we create a DataFrame with columns for `is_treated`, `conversion`, and `uplift` for each user. Positive uplift indicates the advertisement is beneficial, while negative uplift suggests it is better not to show the advertisement.

```
uplift_results = pd.DataFrame(y_pred, columns=uplift_rf.classes_[1:])

best_treatment = np.where(uplift_results['treatment'] < 0, 'control', 'treatment')

auuc_metrics = (uplift_results.assign(is_treated=(treatment_test.values != 'control').astype(int
                              conversion=pd.concat([X_test, y_test, treatment_test], axis=1)['co
                              uplift=uplift_results.max(axis=1))
                .drop(columns=list(uplift_rf.classes_[1:])))
```

4.7 Uplift Curve: Total Cumulative Gain:

The Uplift Curve illustrates the Total Cumulative Gain, showing how effectively the model targets users likely to convert upon seeing the ads. The area between the uplift model's curve and the random targeting line is known as the AUUC (Area Under the Uplift Curve).

```
# Plot the uplift curve
plot_gain(auuc_metrics, outcome_col='conversion', treatment_col='is_treated')
```



4.8 AUUC (Area Under the Uplift Curve) Score:

The AUUC score evaluates uplift modeling performance, similar to AUC (Area Under the ROC Curve). A score close to 1 indicates strong performance, while a score near 0.5 suggests no better performance than random guessing. In this case, the AUUC score is 0.84, indicating good performance.

```
from causalml.metrics import auuc_score
score = auuc_score(auuc_metrics, outcome_col='conversion', treatment_col='is_treated')
print(score)
```

> **Uplift: 0.844375**
> **Random: 0.506221**

4.9 Extracting Targeted Users:

Finally, we extract a list of customers to target based on their uplift scores by calculating the decile for each customer. This allows us to prioritize whom to target according to the budget.

```
# Sort the uplift_results DataFrame by 'uplift' in descending order
uplift_results['user_id'] = X_test.index
uplift_results.rename(columns={'treatment': 'uplift'}, inplace=True)
uplift_results_sorted = uplift_results.sort_values(by='uplift', ascending=False).reset_index(dro

# Print the customer list
print("Customer List Sorted by Uplift:")
display(uplift_results_sorted[['user_id', 'uplift']].head(10))
```

By ranking customers and segmenting them into deciles, you can identify the most effective targets for your campaign and optimize your marketing spend accordingly.

```
# Calculate Rank and Decile Label
uplift_results_sorted['rank'] = uplift_results_sorted['uplift'].rank(method='first', ascending=F
uplift_results_sorted['decile'] = pd.qcut(uplift_results_sorted['rank'], 10, labels=False)

decile_labels = [
    "top 10%", "top 10%-20%", "top 20%-30%", "top 30%-40%", "top 40%-50%",
    "top 50%-60%", "top 60%-70%", "top 70%-80%", "top 80%-90%", "bottom 10%"
]

column_order = ['user_id', 'uplift', 'rank', 'decile', 'decile_label']
uplift_results_sorted['decile_label'] = uplift_results_sorted['decile'].map(lambda x: decile_lab
display(uplift_results_sorted[column_order].head(10))


uplift_results_sorted[column_order].to_csv('customer_list_sorted_by_uplift.csv', index=False)

print("Customer list has been saved to 'customer_list_sorted_by_uplift.csv'")
```

user_id	uplift	rank	decile	decile_label
0	5522331	0.095828	1.0	top 10%
1	405008	0.095828	2.0	top 10%
2	5345387	0.095828	3.0	top 10%
3	1846106	0.082823	4.0	top 10%
4	3453051	0.082823	5.0	top 10%
5	5422016	0.082228	6.0	top 10%
6	5310353	0.079992	7.0	top 10%
7	1924941	0.079992	8.0	top 10%
8	3523409	0.079992	9.0	top 10%
9	3906973	0.079992	10.0	top 10%

5. Conclusion

Here are some key takeaways.

- **Uplift Modeling** helps identify customers who are more likely to respond positively to targeted treatments, optimizing marketing efforts and improving ROI.
- **Tree-based methods** are specially designed decision trees for uplift modeling that use divergence metrics.
- **CausalML** is a powerful library tailored for uplift modeling, offering tools for implementation, visualization, and performance measurement.