Decision Making: Learning Utility Functions

C. Piedallu B. Heurtault J. Mathet

February 7, 2025

Section 1

Overview

Project Focus

- Learning utility functions from pairwise preferences using MIP
- Extending UTA method for multiple preference clusters
- Developing heuristic approaches for larger problems

Section 2

Mathematical Formulation

Mixed Integer Programming Model

Objective Function:

$$\min \sum_{k=1}^{K} \sum_{j=1}^{P} \sigma^k(j)$$

Key Decision Variables:

- $u_i^k(x_i^l)$: Utility values
- $c_k^{(j)}$: Cluster assignments

Key Constraints

Normalization:

$$\sum_{i=1}^{n} u_i^k(x_i^L) = 1 \quad \forall k \in [1, K]$$

Monotonicity:

$$u_i^k(x_i^{l+1}) - u_i^k(x_i^l) \geq \epsilon \quad \forall l \in [0, L]$$

Preferences:

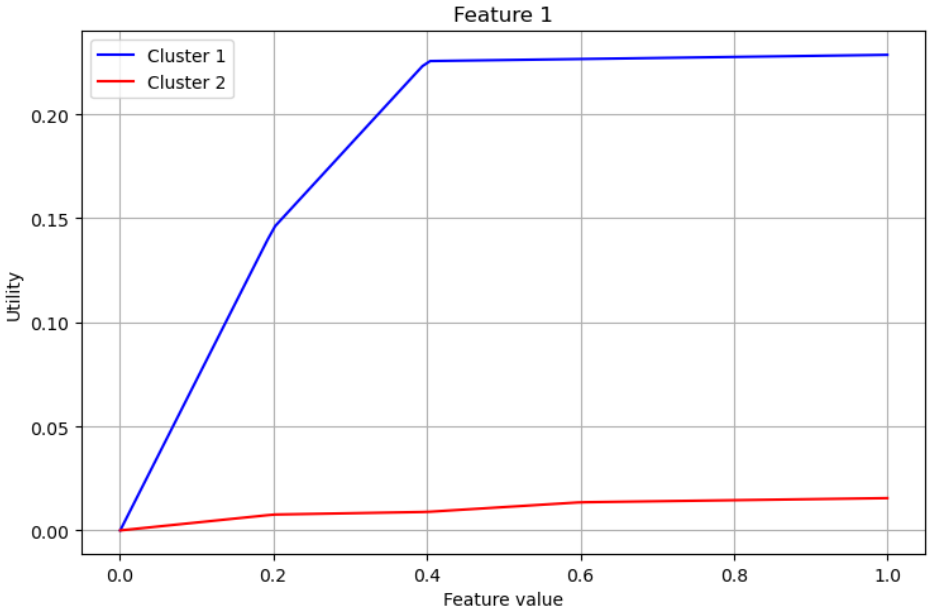
$$M(c_k^{(j)} - 1) \leq \sum_{i=1}^{n} [u_i^k(x_i^{(j)}) - u_i^k(y_i^{(j)})] + \sigma^k(j) \leq M c_k^{(j)}$$

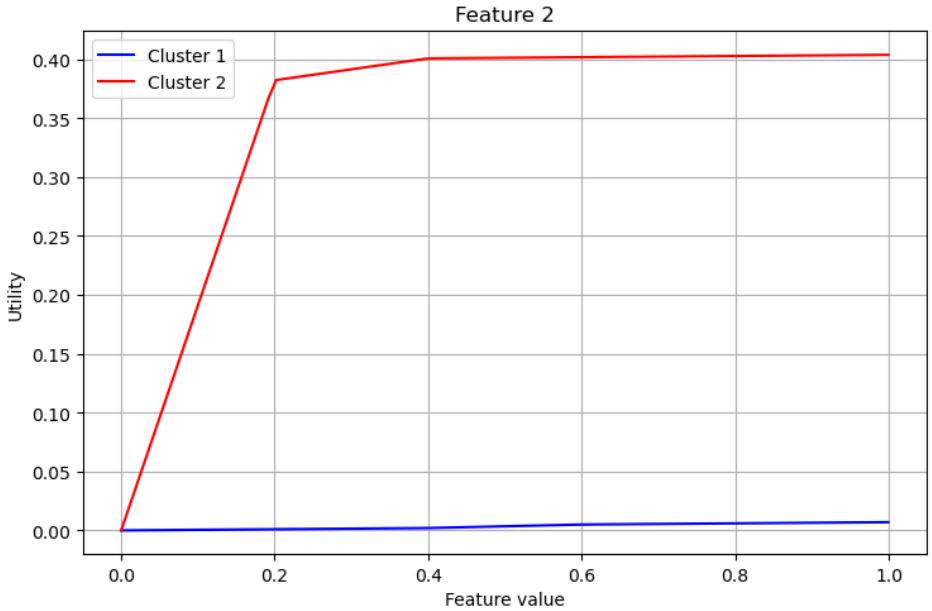
$$\sum_{k=1}^{K} c_k^{(j)} \geq 1 \quad \forall j$$

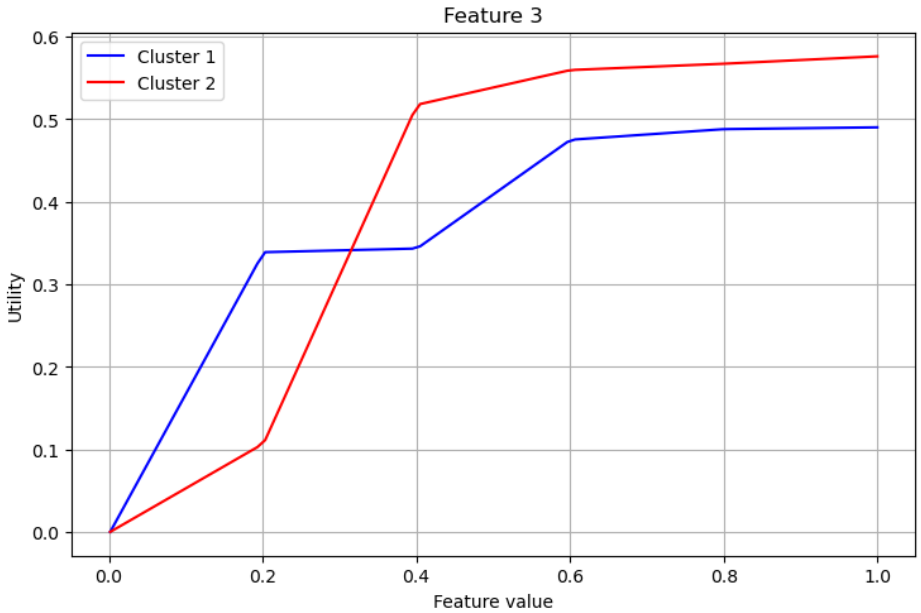
Section 3

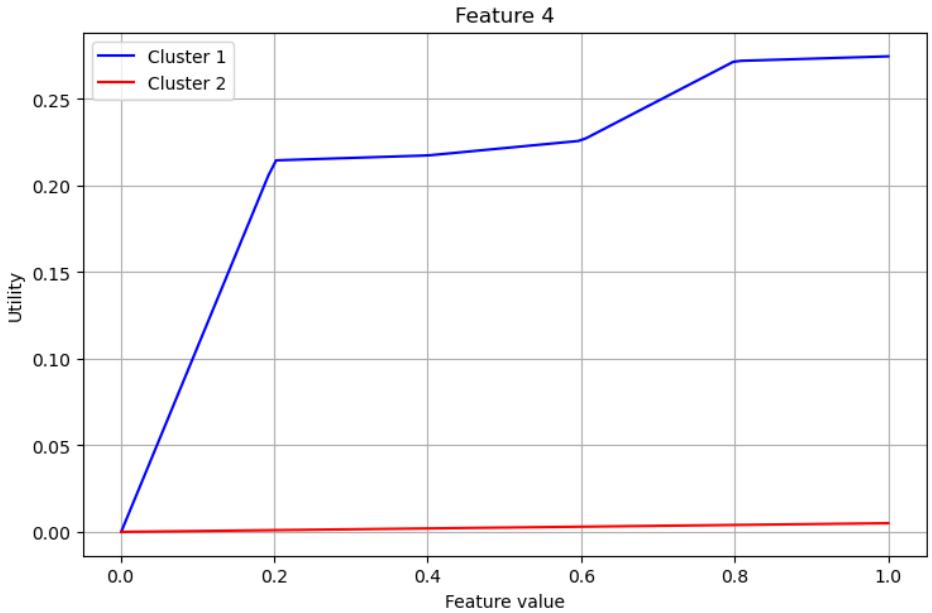
Implementation Results

Learned Utility Functions









Section 4

Heuristic Approach

Key Ideas

1. **Randomly assigns initial clusters** to each preference pair.
2. **Computes utility functions** based on piecewise linear functions.
3. **Refines clusters iteratively** by minimizing the objective function.
4. **Stops when cluster assignments stabilize**, indicating convergence.

Advantages

- Linear scaling with dataset size
- Independent of commercial solvers
- Efficient for large-scale problems

How to transform data

```python
import pandas as pd
import numpy as np

# Define only numerical features that influence preferences
numerical_features = ["price", "range", "acc", "speed", "pollution", "size", "space", "cost", "station"]

preference_pairs = []

for _, row in df.iterrows():
    chosen_col = row["choice"]

    chosen_index = int(chosen_col.replace("choice", "")) - 1  # Convert "choice1" → index 0

    # Extract only numerical features for the chosen car
    X = row[[f"{col}{chosen_index + 1}" for col in numerical_features]].values

    # Iterate over rejected alternatives
    for alt_index in range(6):  # Assuming 6 alternatives per customer
        if alt_index != chosen_index:
            # Extract only numerical features for the rejected alternative
            Y = row[[f"{col}{alt_index + 1}" for col in numerical_features]].values
            preference_pairs.append((X, Y))

print(X)  # Check chosen car numerical features
print(Y)  # Check rejected car numerical features
```

Figure 2: Transformation of cars dataset

Results

```
Extracted 23270 preference pairs.
[[4.1753448 250 4.0 ... 0.7 4 0.1]
 [4.1753448 250 4.0 ... 0.7 4 0.1]
 [4.1753448 250 4.0 ... 0.7 4 0.1]
 ...
 [5.1388859 75 2.5 ... 1.0 2 0.3]
 [5.1388859 75 2.5 ... 1.0 2 0.3]
 [5.1388859 75 2.5 ... 1.0 2 0.3]] [[4.1753448 250 4.0 ... 0.7 4 0.1]
 [4.8177056 400 6.0 ... 1.0 6 0.3]
 [4.8177056 400 6.0 ... 1.0 6 0.3]
 ...
 [5.1388859 75 2.5 ... 1.0 2 0.3]
 [4.1753448 350 4.0 ... 1 6 1.0]
 [4.1753448 350 4.0 ... 1 6 1.0]]
```

Figure 3: Extracted pairs

Learned Utility Functions

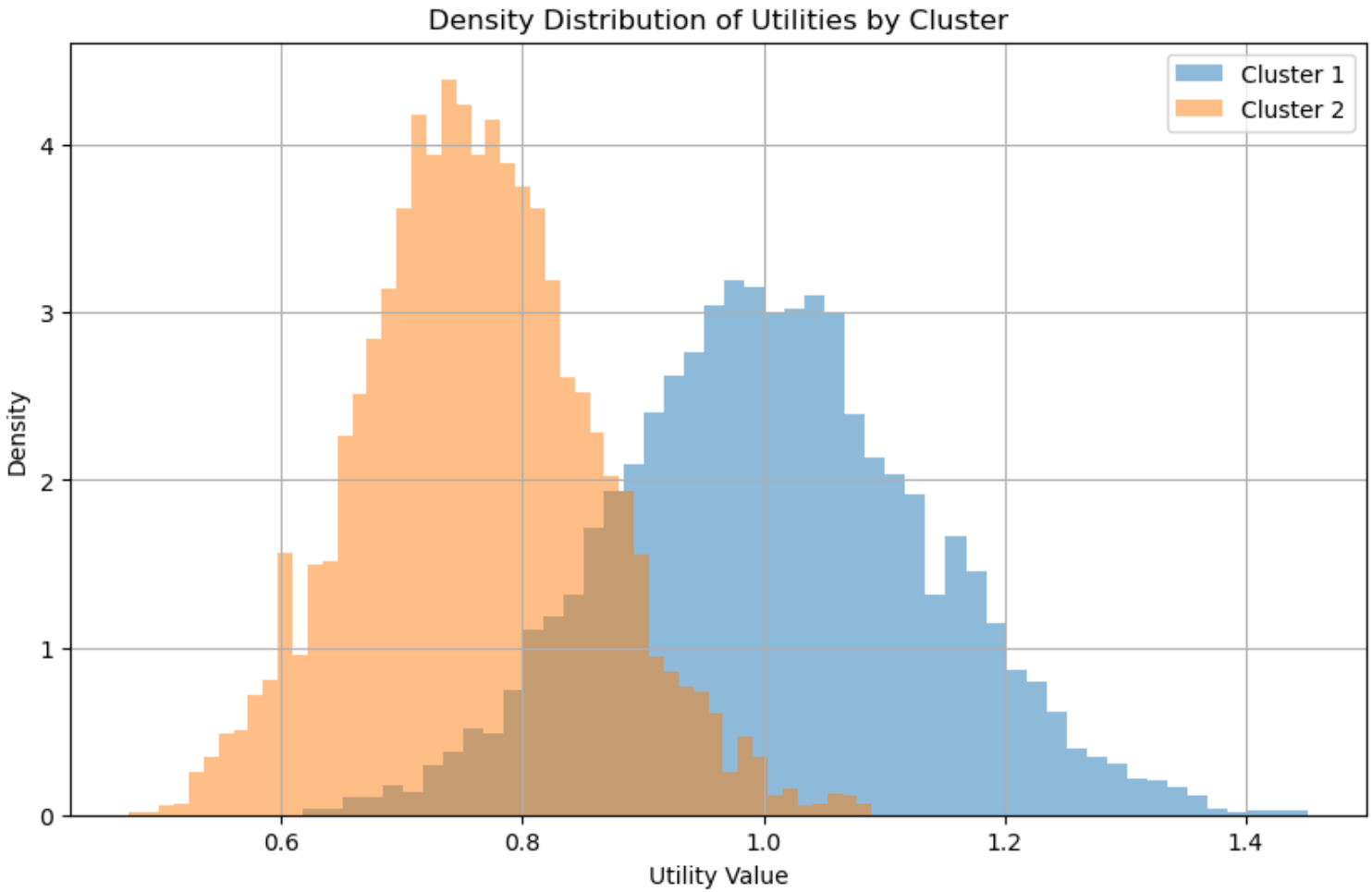


Figure 4: Utility Functions Density

Learned Utility Functions

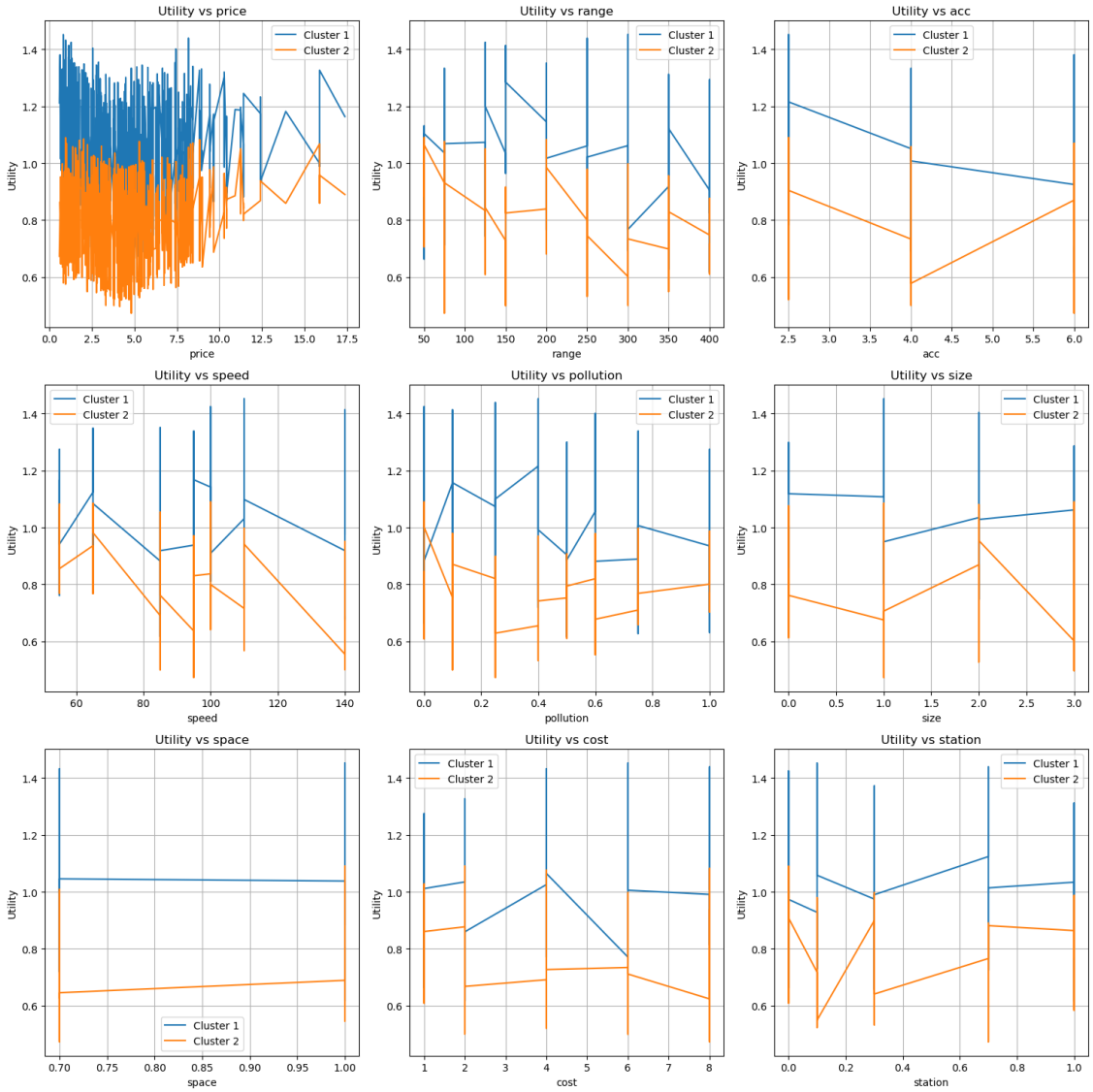


Figure 5: Utility Functions

Contact Information

For more details, contact:

- clovis.piedallu@student-cs.fr
- basile.heurtault@student-cs.fr
- jeremy.mathet@student-cs.fr