Customer Segmentation Case Study

E-Commerce Cluster Analysis Using Hierarchical and K-Means Methods

Juliho Castillo Colmenares

Tec de Monterrey

Business Context

Business Problem: An e-commerce company seeks to understand their customer base by discovering natural segments based on purchasing behavior, engagement patterns, and browsing habits.

Key Questions:

- How many distinct customer segments exist in our data?
- What behavioral patterns characterize each segment?
- How can we tailor marketing strategies to each discovered segment?

Dataset Overview

Dataset: 2,000 customers with 7 behavioral features

Variables:

- monthly_purchases
- avg_basket_size
- total_spend
- session_duration
- email_clicks
- product_views
- return_rate

Important: No predefined labels (unsupervised learning)

Exploratory Data Analysis

Why EDA Matters

Before attempting to discover customer segments, we must first understand the characteristics and relationships within our data. Unlike supervised learning where we have predefined labels, cluster analysis is exploratory in nature, making this preliminary investigation even more critical.

EDA: Approach

What We Do:

- Visualize distributions using histograms
- Identify skewness, outliers, and typical value ranges
- Compute correlation matrix to understand relationships
- Assess variable redundancy

Why: Understanding data structure before algorithmic analysis helps anticipate which features might drive segmentation.

EDA: Distribution Analysis

```python
# Load customer data
df = pd.read_csv("customer_data.csv")

# Summary statistics
print(df.describe())

# Distribution plots for each variable
fig, axes = plt.subplots(3, 3, figsize=(15, 12))
for idx, col in enumerate(df.columns):
    axes[row, col_idx].hist(df[col], bins=30)
```

EDA: Correlation Analysis

```python
# Correlation matrix
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True,
            fmt='.2f', cmap='coolwarm')

# Identify strongest correlations
# Example: monthly_purchases <-> total_spend: 0.94
#          avg_basket_size <-> total_spend: 0.89
```

EDA: Key Findings

Outcome:

- Balanced dataset with diverse customer behaviors
- Right-skewed distributions (some extreme high spenders)
- Strong positive correlations between purchase-related variables
- Email clicks correlate moderately with purchase frequency

Implication: Multiple distinct behavioral patterns suggest natural customer segments exist.

Data Standardization

Why Standardization is Critical

Cluster analysis algorithms rely on distance metrics to measure similarity. However, our behavioral variables are measured in different units and scales:

- Purchases (counts)
- Spending (dollars: 58 to 7,891)
- Return rate (proportions: 0.0 to 0.5)

The Problem Without Standardization

Without standardization: Variables with larger numeric ranges dominate distance calculations.

Example:

- Customer A: 1 purchase, 100 dollars
- Customer B: 2 purchases, 200 dollars

Distance dominated by dollar difference (100) rather than purchase difference (1).

This leads to biased clustering that reflects scale differences, not true behavioral patterns.

Standardization: Approach

Z-score Standardization:

$$z_i = \frac{x_i - \mu}{\sigma}$$

Properties:

- Transforms to mean = 0, standard deviation = 1
- Preserves distribution shape
- Ensures equal contribution to distance calculations
- Value of 2.0 means “2 std deviations above mean”

Standardization: Implementation

```python
from sklearn.preprocessing import StandardScaler

# Standardize features
scaler = StandardScaler()
X_standardized = scaler.fit_transform(df)

# Convert back to DataFrame
df_standardized = pd.DataFrame(
    X_standardized,
    columns=df.columns
)
```

```python
print(df_standardized.describe())
# All means approx 0, all std approx 1
```

Standardization: Outcome

Result: All variables successfully transformed to mean approximately 0 and standard deviation approximately 1.

Impact: Clustering algorithms now treat all behavioral dimensions equally when computing distances between customers.

Example: Total spend (originally 58-7,891 dollars) and return rate (originally 0.0-0.5) now contribute equally to customer similarity.

Hierarchical Clustering

The Fundamental Question

How many customer segments should we look for?

Unlike supervised classification where the number of classes is predetermined, clustering requires us to discover the appropriate number of groups.

Hierarchical clustering provides an elegant solution: Build a complete hierarchy of nested clusters without specifying k in advance.

Hierarchical Clustering: Advantage

Key Benefit: Creates a tree-like structure (dendrogram) showing how customers progressively merge into larger groups.

Result: We can identify the most natural number of segments based on where large jumps in distance occur.

No need to predefine the number of clusters

Hierarchical Clustering: Linkage Methods

How do we measure distance between clusters?

- **Single Linkage:** Minimum distance between any two points
- **Complete Linkage:** Maximum distance between any two points
- **Average Linkage:** Mean distance between all pairs
- **Ward's Method:** Minimizes within-cluster variance

Recommendation: Ward's method typically produces the most balanced clusters for customer segmentation.

Hierarchical Clustering: Implementation

```
from scipy.cluster.hierarchy import linkage, dendrogram

# Compute linkage matrices for different methods
linkage_methods = ['single', 'complete',
                   'average', 'ward']
linkage_matrices = {}

for method in linkage_methods:
    linkage_matrices[method] = linkage(
        X_standardized,
        method=method
    )
```

Dendrogram Interpretation

```
# Create dendrogram for Ward's method
plt.figure(figsize=(14, 7))
dendrogram(linkage_matrices['ward'],
           no_labels=True,
           color_threshold=50)
plt.axhline(y=50, color='r', linestyle='--',
            label='Potential cut (4 clusters)')
```

How to Read:

- Horizontal axis: Customers
- Vertical axis: Distance at which clusters merge
- Large vertical gaps suggest natural cluster boundaries

Hierarchical Clustering: Outcome

Finding: Four dendrograms reveal distinct clustering structures depending on linkage method.

Observations:

- Single linkage: Chaining effects with unbalanced clusters
- Complete linkage: Too many small clusters
- Average linkage: Compromise between extremes
- Ward's method: Clear hierarchical structure, balanced sizes

Decision: Ward's dendrogram suggests 4 clusters as a natural choice.

Extracting Clusters from Dendrogram

```
from scipy.cluster.hierarchy import fcluster

# Extract 4 clusters using Ward's method
n_clusters_hier = 4
hierarchical_labels = fcluster(
    linkage_matrices['ward'],
    n_clusters_hier,
    criterion='maxclust'
)

# Calculate silhouette score
silhouette_hier = silhouette_score(
```

```
    X_standardized,
    hierarchical_labels
)
# Result: 0.458 (moderate cluster quality)
```

Hierarchical Clustering: Results

Cluster Sizes:

- Cluster 0: 440 customers (22.0%)
- Cluster 1: 300 customers (15.0%)
- Cluster 2: 560 customers (28.0%)
- Cluster 3: 700 customers (35.0%)

Silhouette Score: 0.458

Interpretation: Customers reasonably well-separated into clusters with balanced sizes. Score above 0.4 is acceptable for customer segmentation where behavioral boundaries are often fuzzy.

K-Means Clustering

Why K-Means?

While hierarchical clustering provided valuable insights through dendrogram visualization, it has computational limitations:

- Complexity: O(n squared) or worse
- Impractical for very large datasets

K-means offers a scalable alternative:

- Works well with larger customer bases
- Computational complexity: O(n times k times p times iterations)

K-Means: The Challenge

Requirement: Must specify the number of clusters (k) in advance.

Solution: Elbow method provides a data-driven approach.

Goal: Identify the point where adding more clusters provides diminishing returns in terms of improved fit.

Elbow Method: Approach

K-means minimizes within-cluster sum of squared distances (inertia):

$$\mathrm{WCSS} = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

Procedure:

- Run clustering for k = 2 to 10
- Calculate inertia for each k
- Plot inertia vs. k
- Look for the “elbow” point

Elbow Method: Implementation

```python
from sklearn.cluster import KMeans

# Test different k values
inertias = []
silhouette_scores = []
K_range = range(2, 11)

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_standardized)
    inertias.append(kmeans.inertia_)
    silhouette_scores.append(
```

```
        silhouette_score(X_standardized, kmeans.labels_)
    )
```

Elbow Method: Results

Plot Analysis:

- Elbow curve shows diminishing returns after k=4
- Silhouette scores peak at k=4
- Both metrics converge on k=4

Decision: Both hierarchical and elbow analyses suggest k=4

K-Means: Final Clustering

```
# Apply k-means with optimal k
optimal_k = 4
kmeans_final = KMeans(
    n_clusters=optimal_k,
    random_state=42,
    n_init=10
)
kmeans_labels = kmeans_final.fit_predict(
    X_standardized
)

# Silhouette score: 0.458 (same as hierarchical)
```

K-Means: Convergence Validation

Key Finding: K-means achieves identical silhouette score to hierarchical clustering (0.458).

Significance: Convergence between two fundamentally different algorithms provides strong evidence that four customer segments represent genuine structure in the data.

Confidence: Clusters reflect real behavioral patterns rather than algorithmic artifacts.

Recommendation: Use k-means for production deployment due to computational efficiency.

Cluster Interpretation and Profiling

From Statistics to Business Value

Successfully identifying clusters is only the first step. The real business value comes from understanding what distinguishes each segment and translating these differences into actionable marketing strategies.

Goal: Transform abstract cluster labels into concrete customer personas.

Interpretation: Approach

```python
# Add cluster labels to original data
df_with_clusters = df.copy()
df_with_clusters['Cluster_KMeans'] = kmeans_labels

# Calculate cluster means (original units)
cluster_profiles = df_with_clusters.groupby(
    'Cluster_KMeans'
)[df.columns].mean()

# Create heatmap
sns.heatmap(cluster_profiles.T, annot=True)
```

Why original units? Makes profiles interpretable to business stakeholders.

Cluster Characterization: Method

```python
# Compare each cluster to overall means
overall_means = df.mean()

for cluster_id in range(optimal_k):
    cluster_mean = cluster_profiles.loc[cluster_id]

    # Calculate percentage differences
    differences = (
        (cluster_mean - overall_means) /
        overall_means * 100
    )
```

```python
# Identify distinctive features (>10% difference)
high_features = differences.nlargest(3)
low_features = differences.nsmallest(3)
```

Cluster 0: Engaged but Selective Shoppers

Size: 307 customers (15.3%)

Distinctive High Features:

- avg_basket_size: +72.0% vs average
- return_rate: +56.2% vs average
- email_clicks: +51.9% vs average

Distinctive Low Features:

- session_duration: −69.6% vs average
- product_views: −58.0% vs average
- monthly_purchases: −42.1% vs average

Cluster 1: Low-Value Browsers

Size: 707 customers (35.4%)

Distinctive High Features:

- session_duration: +19.0% vs average
- return_rate: +15.2% vs average

Distinctive Low Features:

- total_spend: −83.8% vs average
- email_clicks: −78.3% vs average
- monthly_purchases: −76.2% vs average

Characterization: Spend time browsing but make few purchases.

Cluster 2: Premium High-Value Customers

Size: 432 customers (21.6%)

Distinctive High Features:

- total_spend: +168.3% vs average
- avg_basket_size: +127.5% vs average
- monthly_purchases: +124.1% vs average

Distinctive Low Features:

- return_rate: −24.1% vs average

Characterization: The premium segment with highest spending and purchase frequency.

Cluster 3: Frequent Small-Basket Shoppers

Size: 554 customers (27.7%)

Distinctive High Features:

- monthly_purchases: +23.8% vs average
- product_views: +12.4% vs average

Distinctive Low Features:

- avg_basket_size: −48.8% vs average
- total_spend: −45.1% vs average
- return_rate: −31.8% vs average

Characterization: Regular shoppers with lower average order values.

Cluster Profiling: Outcome

Four Distinct Segments Identified:

Each segment has distinctive characteristics differing by more than 10% from overall average on multiple dimensions.

Confirmation: These represent meaningfully different customer types requiring differentiated marketing strategies.

Next Step: Develop targeted campaigns for each segment.

Cluster Validation with Silhouette Analysis

Beyond Average Scores

While we have identified interpretable customer segments, we should validate the quality of our clustering solution.

Questions:

- Are customers well-matched to their assigned clusters?
- Are some clusters poorly defined?
- Are there misclassifications?

Silhouette Analysis: Approach

Silhouette Coefficient for each customer:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

where:

- $a(i)$ = average distance to points in same cluster
- $b(i)$ = average distance to points in nearest neighboring cluster

Range: −1 to +1

- Near +1: Well-matched to cluster
- Near 0: On border between clusters

- Negative: Possible misclassification

Silhouette Plot: Implementation

```python
from sklearn.metrics import silhouette_samples

# Calculate silhouette values for each customer
silhouette_vals = silhouette_samples(
    X_standardized,
    kmeans_labels
)

# Create visualization
for i in range(optimal_k):
    cluster_vals = silhouette_vals[kmeans_labels == i]
    cluster_vals.sort()
```

```
plt.fill_betweenx(y_range, 0, cluster_vals)
```

Silhouette Analysis: Interpretation

Key Observations:

- Width of each section represents cluster size
- All clusters extend beyond average score (0.458)
- Most customers have positive coefficients
- Few customers near zero (boundary cases)
- Very few negative coefficients (rare misclassifications)

Conclusion: Generally good cluster quality across all four segments. Segmentation is sound.

Visualization in 2D Space

The Challenge

We have worked with seven-dimensional customer data, which is impossible to visualize directly.

Problem: Humans can only perceive 2-3 spatial dimensions effectively.

Solution: Principal Component Analysis (PCA) projects the 7D space onto 2D while preserving as much information as possible.

PCA: Approach

What PCA Does:

- Transforms 7 original variables into uncorrelated components
- Orders components by variance explained
- PC1 captures maximum variance
- PC2 captures maximum remaining variance orthogonal to PC1

Result: 2D visualization retaining most important structure.

PCA Projection: Implementation

```
from sklearn.decomposition import PCA

# Apply PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_standardized)

# Variance explained
print(f"PC1: {pca.explained_variance_ratio_[0]:.3f}")
print(f"PC2: {pca.explained_variance_ratio_[1]:.3f}")

# Create scatter plot
```

```
plt.scatter(X_pca[:, 0], X_pca[:, 1],
            c=kmeans_labels, cmap='tab10')
```

PCA Visualization: Outcome

Results:

- 2D projection captures moderate proportion of variance (40-60%)
- Reasonably good cluster separation visible
- Some overlap consistent with silhouette analysis
- K-means centroids clearly separated
- Both methods show similar spatial patterns

Interpretation: While 2D view sacrifices some information, it confirms clusters are spatially distinct and well-positioned.

Business Recommendations

From Analysis to Action

Having completed technical analysis and validation, we now translate statistical findings into actionable business strategies.

Goal: Create customer personas and design specific marketing tactics for each segment aligned with their behaviors and needs.

Segment 1: Engaged but Selective Shoppers

Profile: High basket sizes, high returns, engaged with emails but low browsing.

Marketing Strategy:

- Launch VIP loyalty program with exclusive perks
- Offer early access to new products
- Provide free expedited shipping
- Personalized product recommendations
- Improve product information to reduce returns

Segment 2: Low-Value Browsers

Profile: High browsing time but low purchases and engagement.

Marketing Strategy:

- Implement abandoned cart recovery campaigns
- Use retargeting ads to re-engage visitors
- Offer limited-time discounts to incentivize first purchase
- Improve product information and customer reviews
- Create urgency with flash sales

Segment 3: Premium High-Value Customers

Profile: Highest spending, purchase frequency, and basket sizes.

Marketing Strategy:

- Exclusive VIP treatment and recognition
- Premium customer service channel
- Early access to new collections
- Referral program incentives
- Maintain satisfaction to prevent churn

Segment 4: Frequent Small-Basket Shoppers

Profile: Regular purchases with lower average order values.

Marketing Strategy:

- Send targeted discount codes and bundle offers
- Promote free shipping thresholds to increase basket size
- Highlight clearance and sale items
- Create value packs and multi-buy promotions
- Build purchase frequency through regular engagement

Key Findings: Summary

Methodology:

- Analyzed 2,000 customers across 7 behavioral variables
- Standardized data to ensure equal feature weighting
- Applied both hierarchical (Ward's) and k-means clustering
- Used elbow method and silhouette analysis

Result: Both methods converged on 4 distinct customer segments with silhouette score of 0.458.

Key Findings: Segments

Four Distinct Customer Segments:

1. **Engaged Selective (15%)**: High basket, high return
2. **Low-Value Browsers (35%)**: Browse but don't buy
3. **Premium High-Value (22%)**: Highest spenders
4. **Frequent Small-Basket (28%)**: Regular small orders

Each requires tailored marketing strategies.

Methodological Learnings

Hierarchical Clustering:

- Provides hierarchy view
- No need to predefine k
- Best for smaller datasets

K-Means:

- More scalable and faster
- Requires specifying k
- Better for production deployment

Both methods converged: Strong evidence of genuine structure.

Business Value

Enables:

- Targeted marketing campaigns by segment
- Optimized resource allocation to high-value customers
- Opportunities for customer retention and growth
- Data-driven customer understanding

ROI: Improved marketing efficiency and customer lifetime value.

Next Steps

Implementation:

1. Validate cluster assignments with domain experts
2. Implement targeted campaigns for each segment
3. Monitor segment-specific KPIs (conversion, AOV, retention)
4. Re-run clustering periodically to detect behavioral shifts
5. Consider additional features (geographic, demographic) for refinement

Validation in Practice

Important Note: In real-world unsupervised learning, true labels do not exist.

Cluster validation relies on:

- Domain expertise
- Business metrics
- Silhouette analysis
- Stability across different methods

This synthetic dataset included true labels only for educational validation purposes.

Python Code Resources

Complete implementation available in:

- `customer_clustering_analysis.ipynb`
- `fetch_customer_data.py` (data generation)
- `CUSTOMER_DATA_DICTIONARY.md` (variable descriptions)

Key Libraries:

- scikit-learn (KMeans, StandardScaler)
- scipy (hierarchical clustering)
- pandas, numpy (data manipulation)
- matplotlib, seaborn (visualization)

Questions?

Thank you for your attention!

Juliho Castillo Colmenares
julihocc@tec

Office: Tec de Monterrey CCM Office 1540
Office Hours: Monday-Friday, 9:00 AM - 5:00 PM