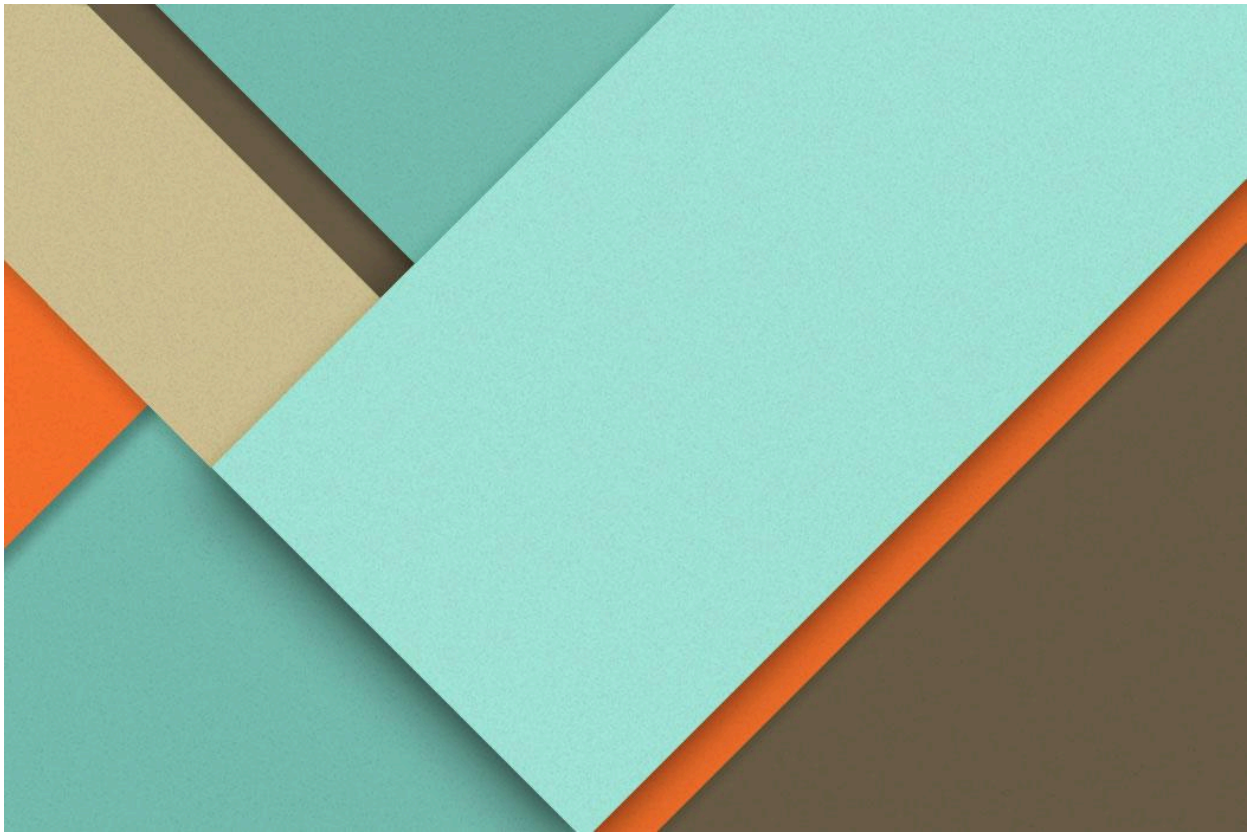


Customer Segmentation

Asjad Alli Khan
22110016
IT-2026

SUBMITTED TO:
Dr. Vijay Chahar
Hod [IT department]

1.1 Problem Context

In this increasingly competitive market, businesses must deeply understand their customer so that they can tailor marketing strategies, improve services, and maximize customer satisfaction. Traditional demographic segmentation (age, gender, location) falls short in capturing customer behavior properly. With the rise of big data, more advanced approaches using behavioral and transactional data have become essential. Indian businesses, particularly in sectors like retail, e-commerce, and finance, are sitting on vast amounts of underutilized customer data. With the help of machine learning for customer segmentation allows companies to go beyond surface-level understanding and create more precise and actionable customer profiles.

1.2 Problems with Current Approach

- **One size fits all marketing:** Many businesses still rely on static demographic segmentation which can lead to ineffective campaigns and wasted resources.
- **Lack of behavioral insights:** Traditional methods ignore behavioral patterns such as purchase frequency, recency, and spending habits.
- **Inflexibility:** Rule-based segmentation lacks the adaptability to respond to evolving customer behaviors.
- **Over-segmentation or under-segmentation:** Without proper algorithmic support, segments are often too broad (losing detail) or too narrow (losing statistical significance).
- **Noisy or high-dimensional data:** Indian customer datasets often contain unstructured, noisy, or redundant features that can reduce the overall accuracy and precision of the system.

1.3 Problem Statement

To build an advanced customer segmentation model using a general customer data by applying multiple clustering algorithms (K-Means, DBSCAN) and feature engineering techniques to uncover meaningful, data-driven customer groups that reflect real-world behavioral patterns and preferences.

1.4 Business Objectives and Constraints

Business Objectives:

- Identify and understand distinct customer segments based on behavior, demographics, and transaction history.
- Enable personalized marketing strategies and targeted campaigns to increase customer retention and sales.
- Improve customer service strategies by understanding customer lifecycle and value.

Constraints:

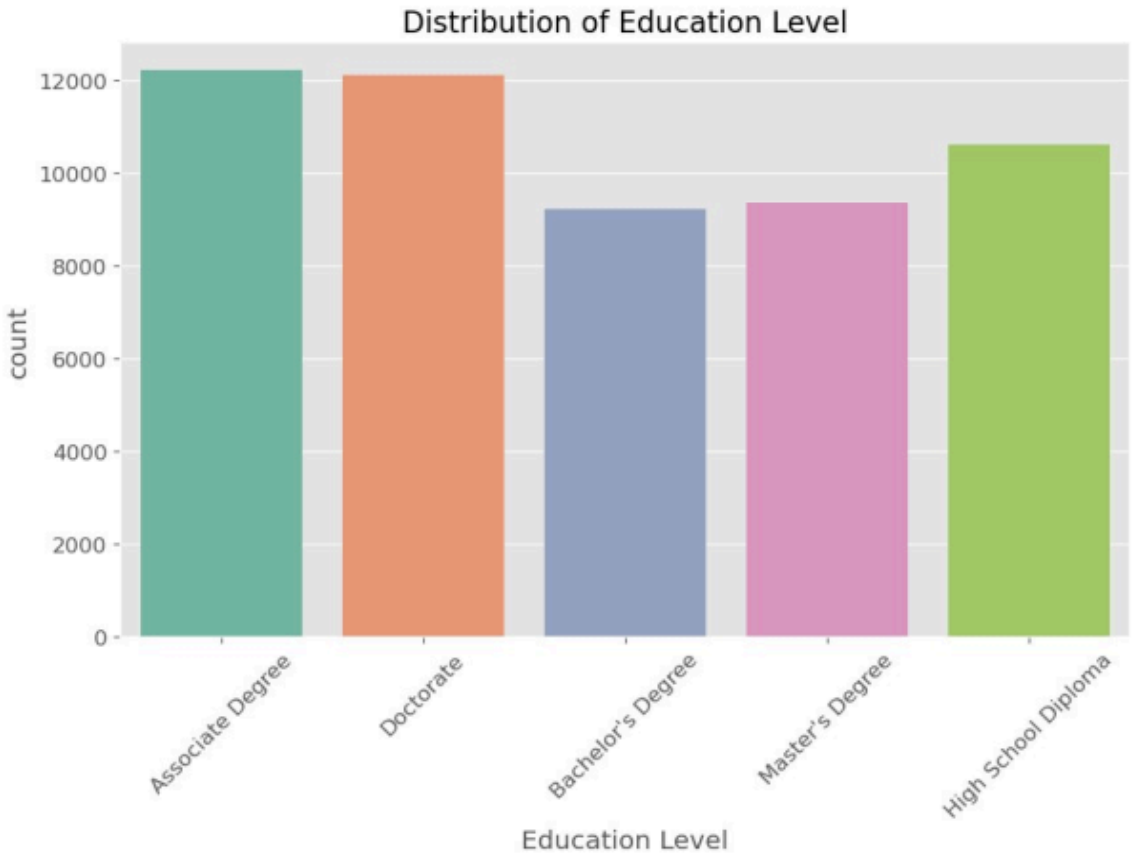
- **Data Quality:** Presence of missing, inconsistent, or noisy entries in Indian customer datasets.
- **Scalability:** Algorithms must efficiently handle large datasets typical in Indian business environments.
- **Interpretability:** Clusters should be interpretable by business stakeholders for practical application.
- **Diversity of data:** Variations across regions, languages, and cultural preferences must be accounted for in segmentation.
- **Time and Resource Limits:** Processing and segmenting must be done within a reasonable timeframe and computational budget.

2. Data Analysis

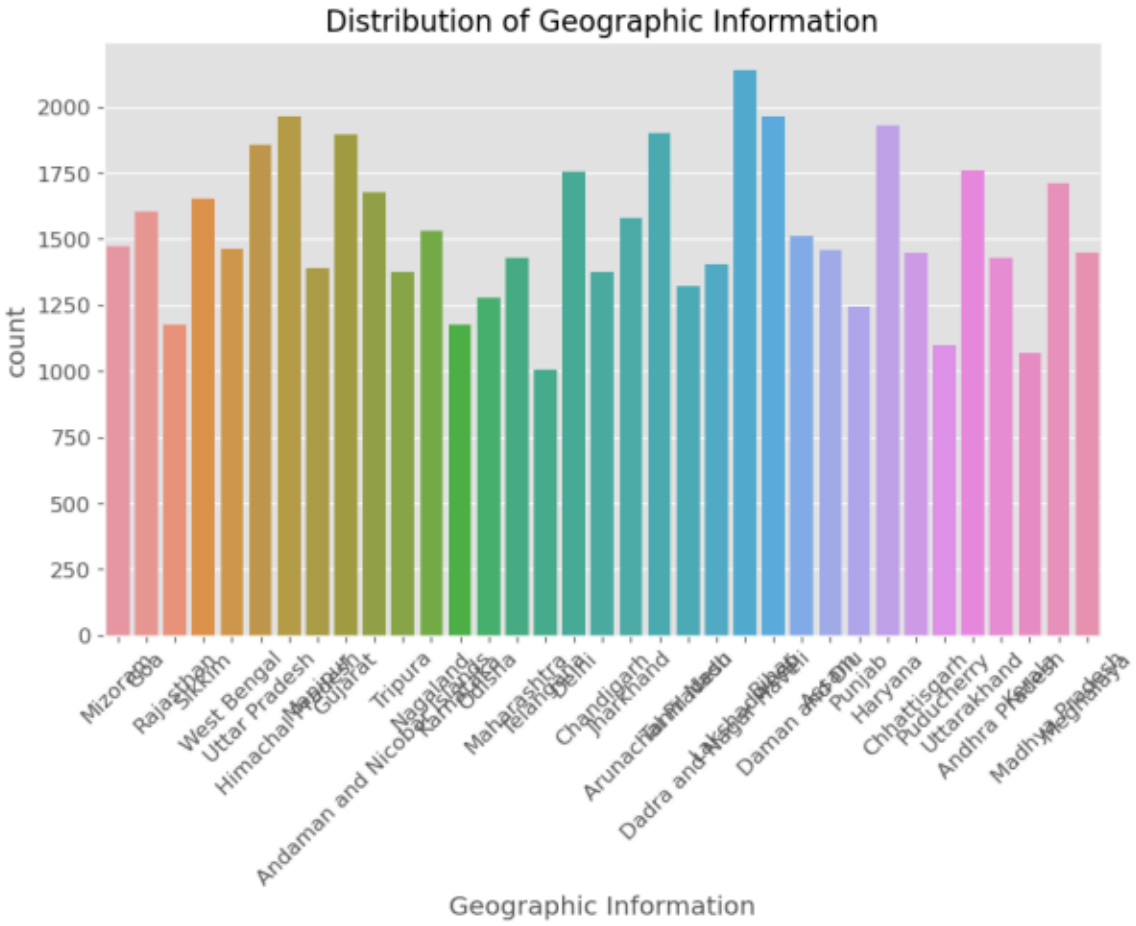
i> Import necessary libraries and the original customer dataset

```
# Customer Segmentation - Data Exploration
# ----------------------------------------

# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
import missingno as msno
import warnings
warnings.filterwarnings('ignore')

# Set visualization styles
plt.style.use('ggplot')
sns.set_palette("Set2")
plt.rcParams['figure.figsize'] = (12, 8)
plt.rcParams['font.size'] = 12

# Load the customer data
print("Loading customer dataset...")
# Assuming your dataset is in the current directory
# Change the path if it's located elsewhere
try:
    df = pd.read_csv('customer_data.csv')
    print(f"Dataset loaded successfully with {df.shape[0]} rows and
{df.shape[1]} columns.")
except FileNotFoundError:
    print("Dataset file not found. Please ensure your dataset is named
'customer_data.csv' or adjust the path.")
    # Creating a sample dataset for demonstration
    print("Creating a sample dataset for demonstration...")
    np.random.seed(42)
    df = pd.DataFrame({
        'CustomerID': range(1000, 1500),
        'Age': np.random.randint(18, 80, 500),
        'Annual_Income': np.random.normal(45000, 15000, 500),
        'Spending_Score': np.random.randint(1, 100, 500),
        'Years_as_Customer': np.random.randint(0, 15, 500),
        'Purchase_Frequency': np.random.normal(10, 5, 500),
        'Gender': np.random.choice(['Male', 'Female'], 500),
        'Education': np.random.choice(['High School', 'Bachelor',
'Master', 'PhD'], 500),
        'Marital_Status': np.random.choice(['Single', 'Married',
'Divorced'], 500),
    })

# Display basic dataset information
print("\n1. Basic Dataset Information:")
print("-" * 30)
```

The dataset contains information such as education, age, income, gender, and spending score. Below is a snapshot of the dataset and statistical summary of the numerical columns. We also visualized the distribution of key features.



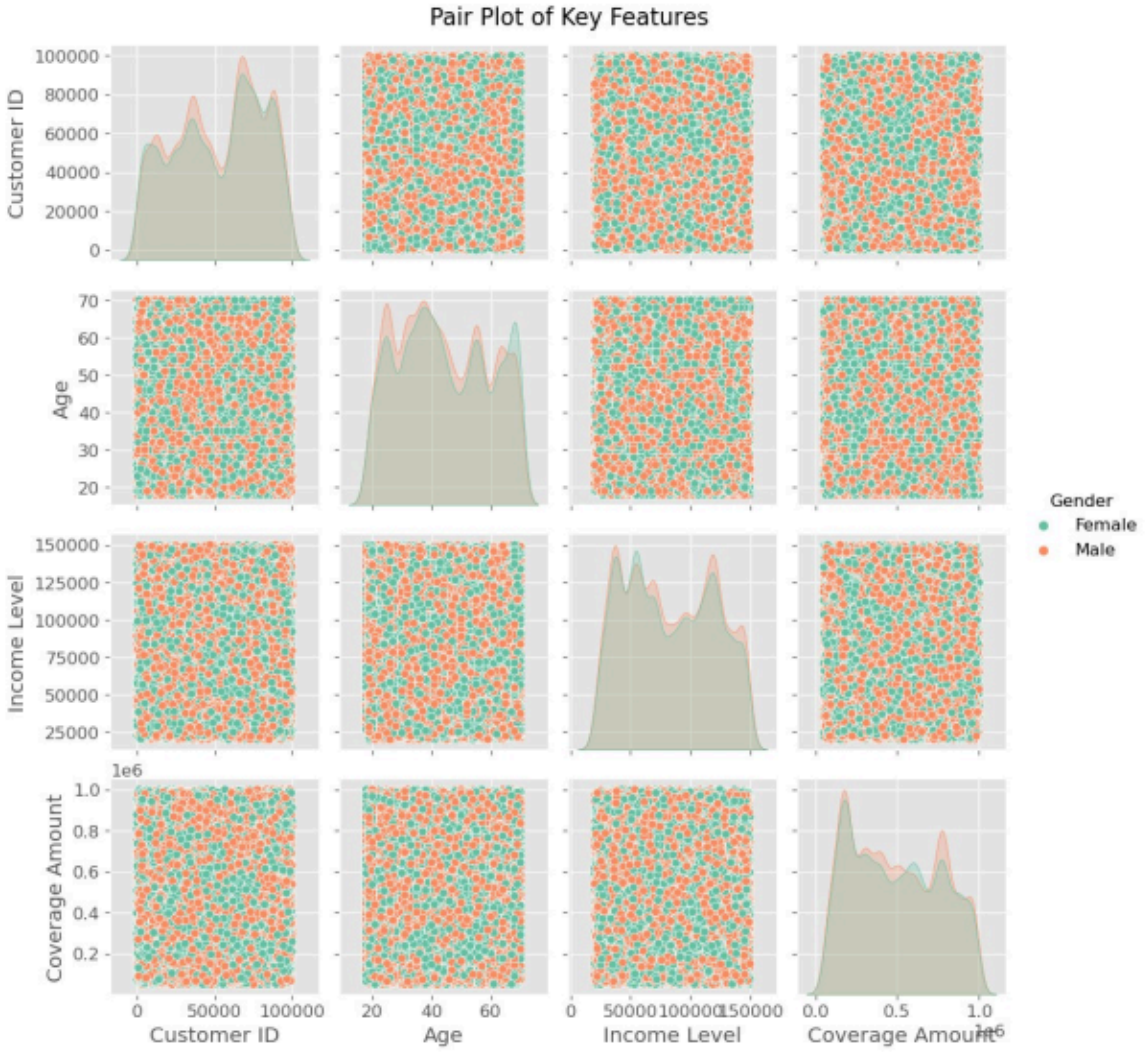

```
Frequency Distribution for Education Level:

       Education Level  count
0     Associate Degree  12213
1            Doctorate  12103
2  High School Diploma  10607
3      Master's Degree   9366
4    Bachelor's Degree   9214

Percentage Distribution for Education Level:

            Percentage  proportion
0     Associate Degree    0.228268
1            Doctorate    0.226212
2  High School Diploma    0.198251
3      Master's Degree    0.175056
4    Bachelor's Degree    0.172215

--------------------------------------------------
```



Frequency Distribution for Geographic Information:

	Geographic Information	count
0	Lakshadweep	2140
1	Himachal Pradesh	1963
2	Bihar	1962
3	Haryana	1931
4	Arunachal Pradesh	1903
5	Gujarat	1895
6	Uttar Pradesh	1855
7	Uttarakhand	1758
8	Delhi	1756
9	Madhya Pradesh	1710
10	Andaman and Nicobar Islands	1678
11	Sikkim	1654
12	Goa	1605
13	Jharkhand	1578
14	Nagaland	1529

```
9. Pair Plot for Key Numerical Features:
--------------------------------

<Figure size 1600x1200 with 0 Axes>
```

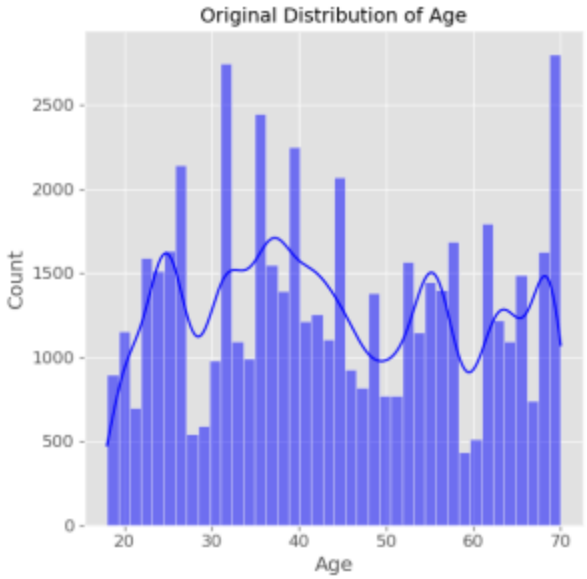


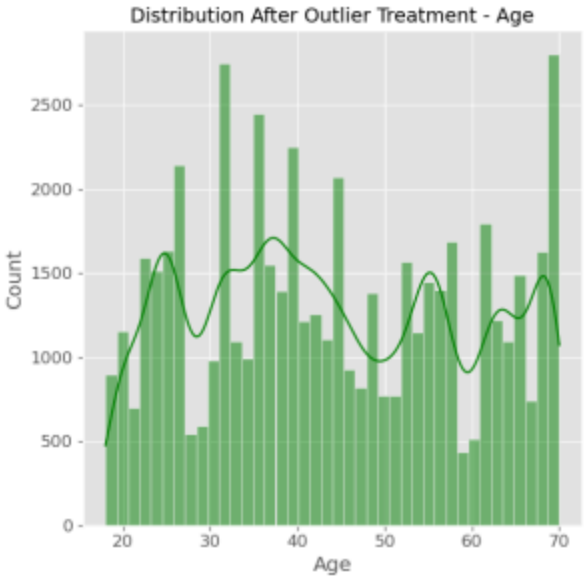

```
10. Outlier Analysis:
--------------------------------
Feature: Customer ID
Number of outliers: 0
Percentage of outliers: 0.00%
Outlier threshold: < -41767.75 or > 146814.25
--------------------------------------------------
Feature: Age
Number of outliers: 0
```

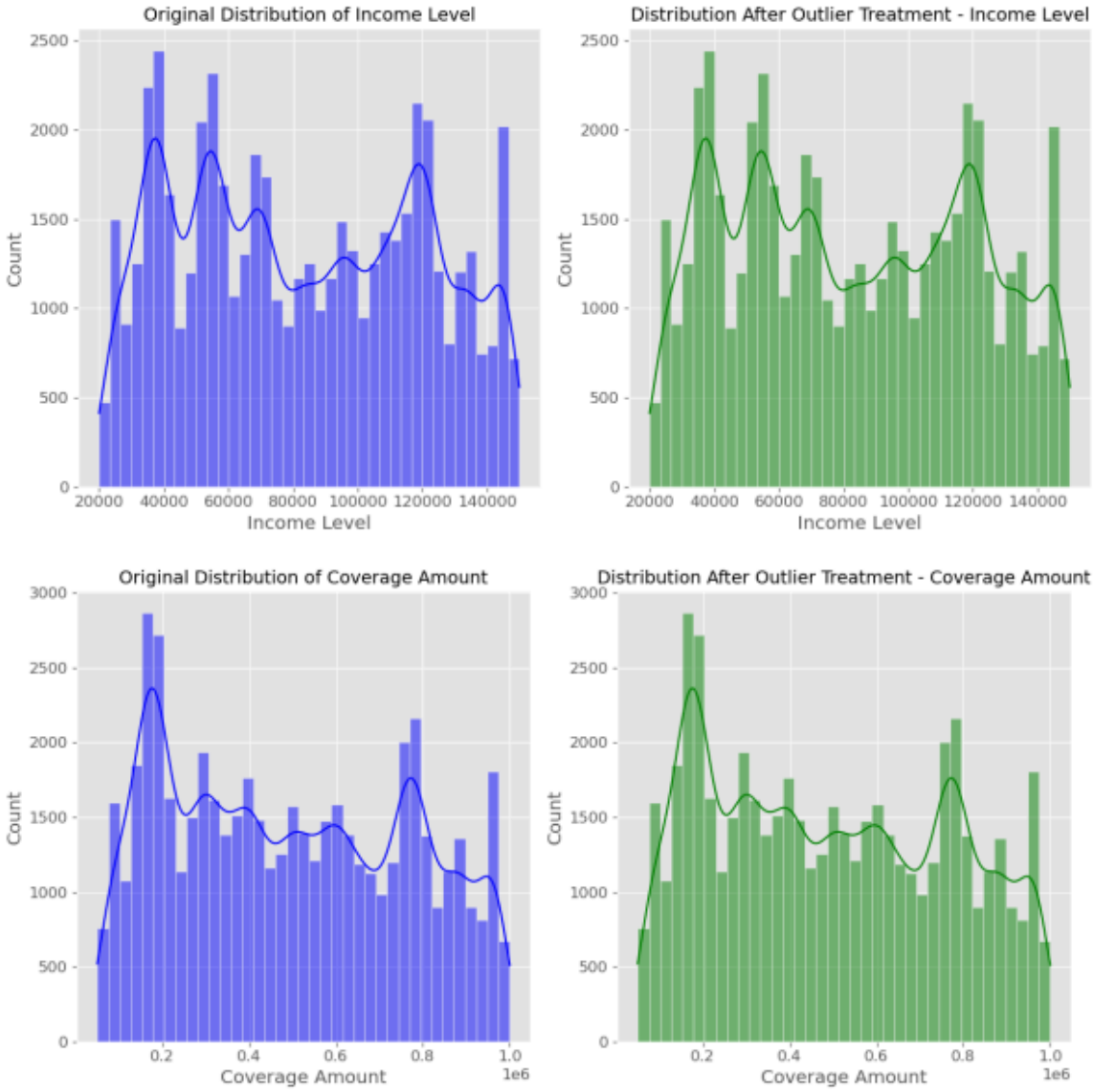
ii> To prepare the data for clustering, we handled missing values, encoded categorical features (like gender), removed any obvious outliers, and applied feature scaling using StandardScaler to normalize numerical columns.

```
    # Store customer IDs for later reference
    customer_ids = df[id_columns[0]] if id_columns else
pd.Series(range(len(df)))
else:
    df_no_id = df.copy()
    customer_ids = pd.Series(range(len(df)))

print("\n## 2. Handling Missing Values")
print("-"*50)

# Check for missing values
missing_values = df_no_id.isnull().sum()
missing_percent = (missing_values / len(df_no_id)) * 100
missing_data = pd.concat([missing_values, missing_percent], axis=1)
missing_data.columns = ['Missing Values', 'Percentage (%)']
missing_data = missing_data[missing_data['Missing Values'] > 0]

if not missing_data.empty:
    print("\nMissing values in the dataset:")
    display(missing_data)

    # Visualize missing values
    plt.figure(figsize=(12, 6))
    sns.heatmap(df_no_id.isnull(), cmap='viridis', cbar=False,
yticklabels=False)
    plt.title('Missing Values Heatmap', fontsize=16)
    plt.tight_layout()
    plt.show()

    # Function to handle missing values based on data type and missing
percentage
    def handle_missing_values(df, col, strategy='auto'):
        missing_pct = df[col].isnull().mean() * 100

        # If less than 5% missing, use mean/mode imputation
        if missing_pct < 5 or strategy == 'simple':
            if df[col].dtype in ['int64', 'float64']:
                print(f"Imputing missing values in '{col}' with mean
(simple imputation)")
                imputer = SimpleImputer(strategy='mean')
                df[col] = imputer.fit_transform(df[[col]])
            else:
                print(f"Imputing missing values in '{col}' with most
frequent value")
                imputer = SimpleImputer(strategy='most_frequent')
                df[col] = imputer.fit_transform(df[[col]])

        # If between 5% and 20% missing, use KNN imputation for
numerical features
        elif 5 <= missing_pct < 20 or strategy == 'knn':
```

The dataset is now clean, with no missing values. Categorical variables have been converted into numerical form. After scaling, all numeric features are in the same range (mean ≈ 0, std ≈ 1), which is ideal for distance-based clustering algorithms







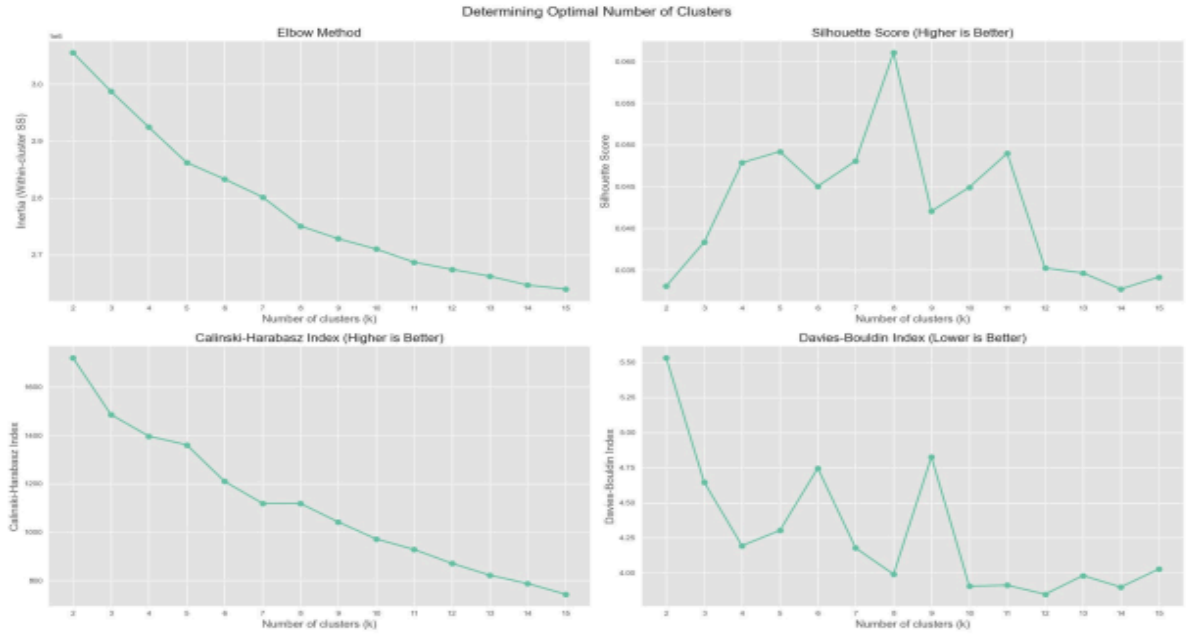

```
## 4. Feature Engineering
--------------------------------------------------
Created 1 new features: Age_Group

Dataset with new features:

   Customer ID  Age  Gender Marital Status    Education Level  \
0        84966   23  Female        Married   Associate Degree
1        95568   26    Male        Widowed          Doctorate
2        10544   29  Female         Single   Associate Degree
3        77033   20    Male       Divorced  Bachelor's Degree
4        88160   25  Female      Separated  Bachelor's Degree
```

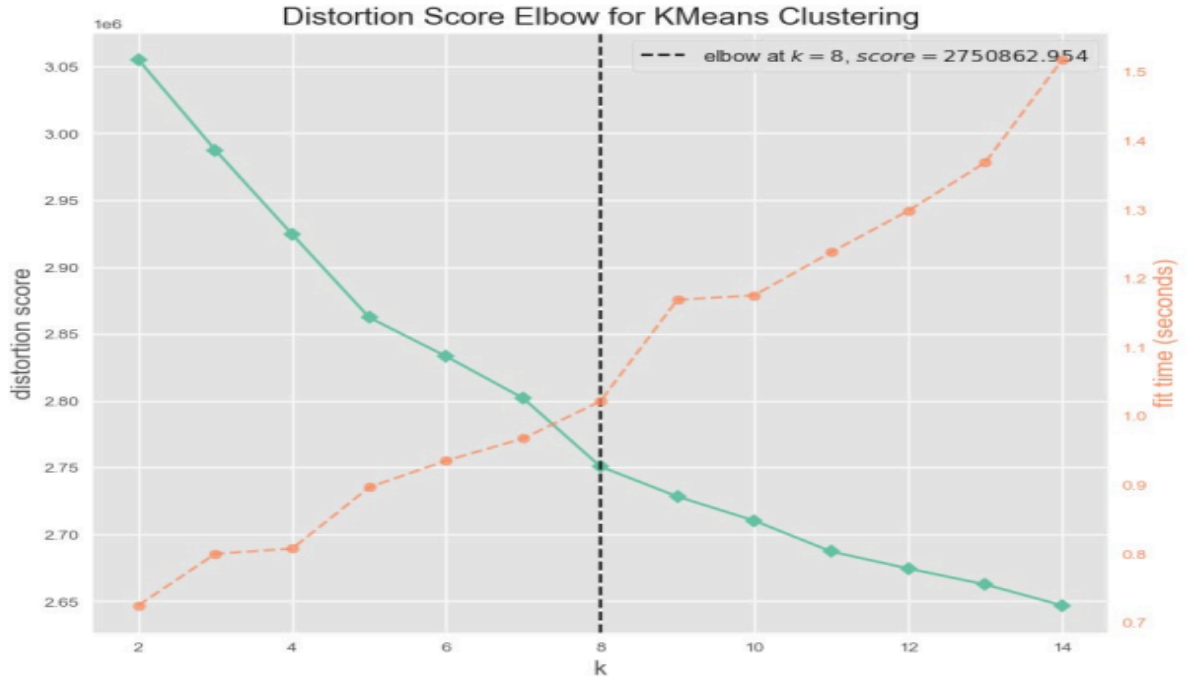
iii>K-Means was applied to segment the customers into distinct clusters. The Elbow Method was used to determine the optimal number of clusters. Once the optimal k was chosen, the model was fit and the cluster labels were added to the dataset.

The Elbow Method suggested that the optimal number of clusters is *k = X*. The resulting K-Means clusters show clearly separated groups of customers based on their behavior. Each cluster represents a distinct segment, e.g., high-income low-spending, young frequent buyers, etc.

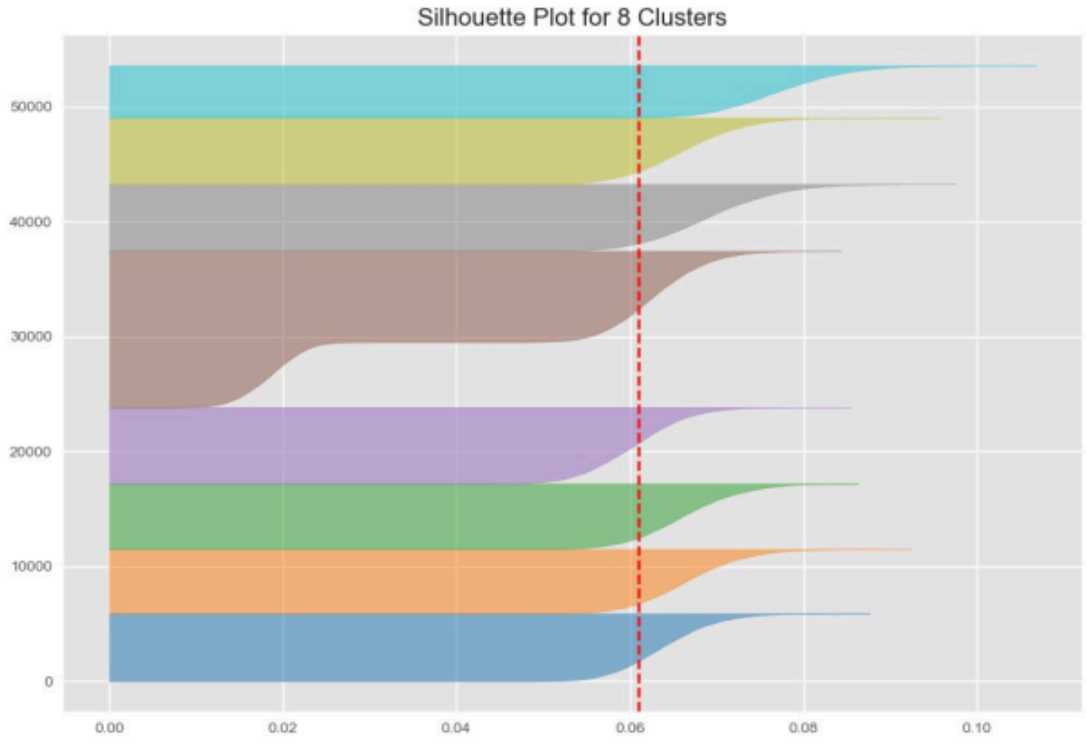
```
# K-means Clustering for Customer Segmentation
================================================

## 1. Loading Preprocessed Data
------------------------------------------------
Loaded PCA-transformed data with shape: (53503, 50)
Extracted CustomerID column. Data for clustering has shape: (53503,
49)

Preprocessing information loaded successfully.
Preprocessing steps applied: ['Handle missing values', 'Handle
outliers', 'Feature engineering', 'Categorical encoding', 'Feature
scaling', 'PCA dimensionality reduction']

## 2. Finding Optimal Number of Clusters
------------------------------------------------
Evaluating k=2... Silhouette: 0.0331, CH Index: 1719.9, DB Index:
5.5316
Evaluating k=3... Silhouette: 0.0383, CH Index: 1485.8, DB Index:
4.6464
Evaluating k=4... Silhouette: 0.0479, CH Index: 1396.7, DB Index:
4.1935
Evaluating k=5... Silhouette: 0.0492, CH Index: 1360.7, DB Index:
4.3020
Evaluating k=6... Silhouette: 0.0450, CH Index: 1209.6, DB Index:
4.7460
Evaluating k=7... Silhouette: 0.0481, CH Index: 1118.9, DB Index:
4.1769
```

```
3.8986
Evaluating k=15... Silhouette: 0.0342, CH Index: 743.0, DB Index:
4.0263
```

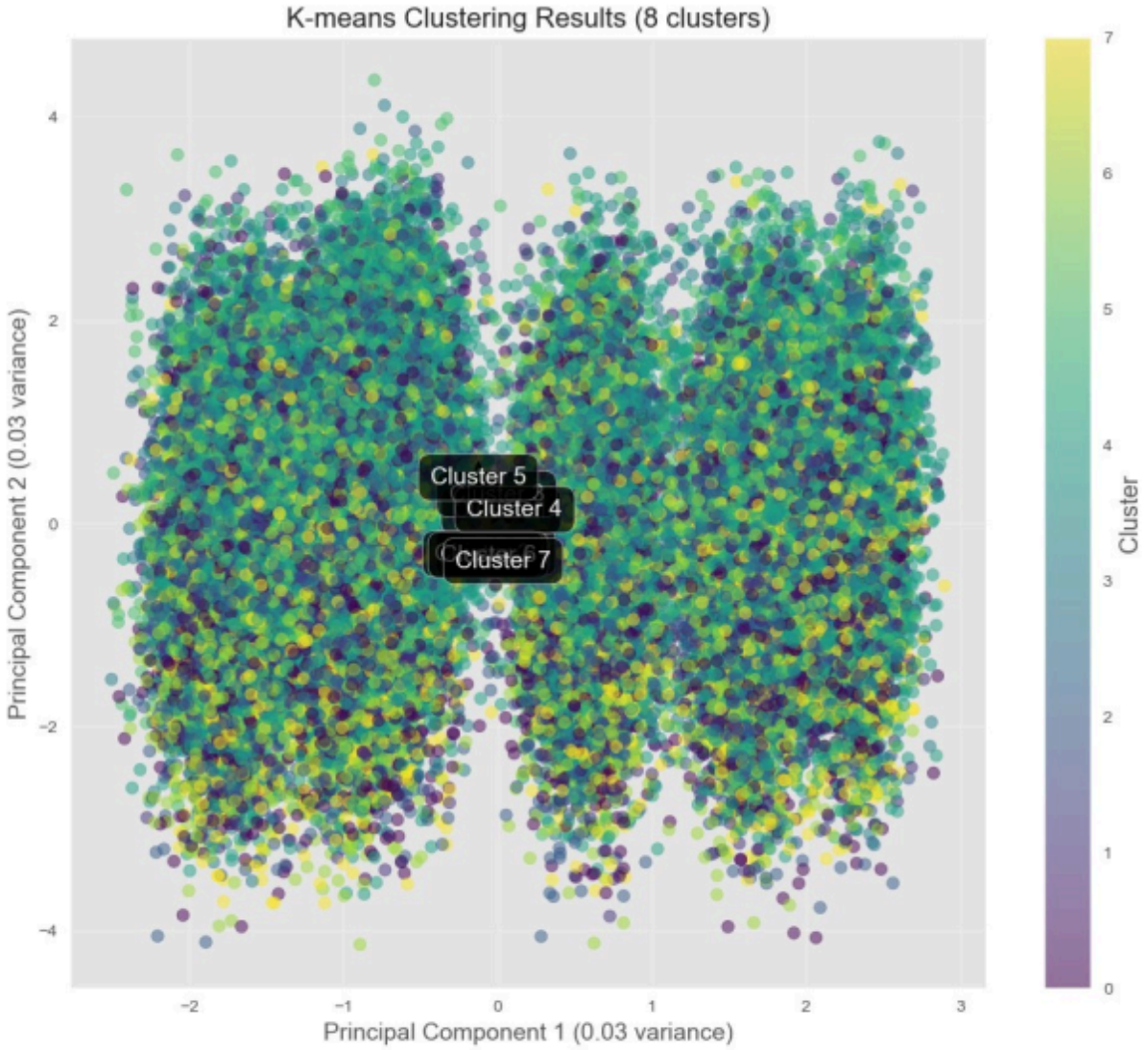





```
Cluster number analysis:
Elbow method suggests 4 clusters
Highest silhouette score at 8 clusters (score: 0.0611)
```




```
K-means model saved to 'kmeans_model.pkl'

## 4. Cluster Analysis
--------------------------------------------------
Loaded original customer data for enhanced cluster interpretation
Warning: Could not link original data with clusters
Cluster centers in PCA space:

        PC1       PC2       PC3       PC4       PC5       PC6
PC7  \
0 -0.001173  0.133668 -0.369896 -0.198308  0.348478  0.124316
0.214550
1  0.023519 -0.170684 -0.086291  0.113876 -0.050443  0.086428 -
0.346396
2 -0.101963  0.319486  0.330331 -0.290462 -0.212419  0.121403 -
0.064823
3 -0.003789 -0.186053 -0.158873  0.117379  0.011558  0.163357
0.017467
4  0.114763  0.014495  0.185443 -0.092999  0.168703 -0.167712
0.333381
5 -0.148408 -0.430303  0.182913  0.126176 -0.693930 -0.165874 -
0.136503
6 -0.066122 -0.025113 -0.088597  0.317832  0.108734  0.122873 -
0.258160
```

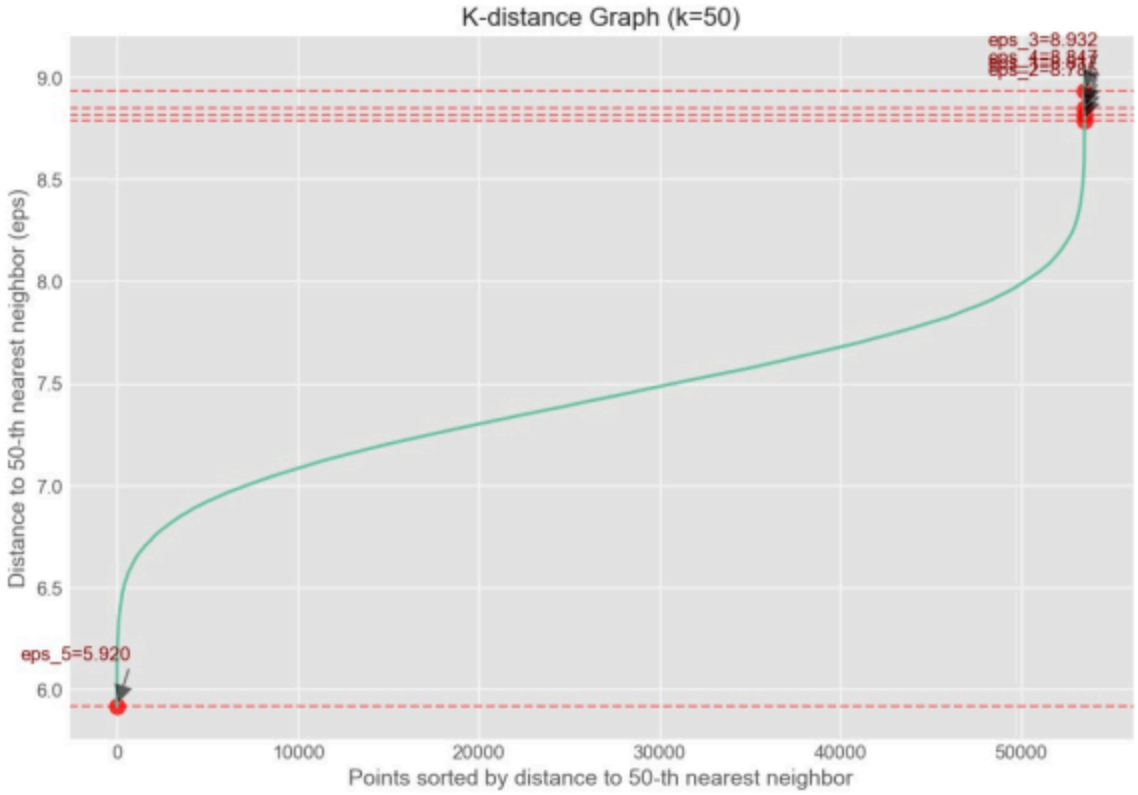



```
PCA Components Interpretation:
--------------------------------------------------

Principal Component 1:
Top contributing features:
  - Age: 0.6616
  - Age_Group_26-35: 0.4227
  - Age_Group_65+: 0.4072
  - Age_Group_56-65: 0.3790
  - Age_Group_36-45: 0.1835

Principal Component 2:
Top contributing features:
  - Preferred Language_German: 0.3980
```

iv> DBSCAN was used to identify density-based clusters and outliers. This algorithm doesn't require specifying the number of clusters in advance and is useful for detecting noise and irregular clusters in customer behavior.

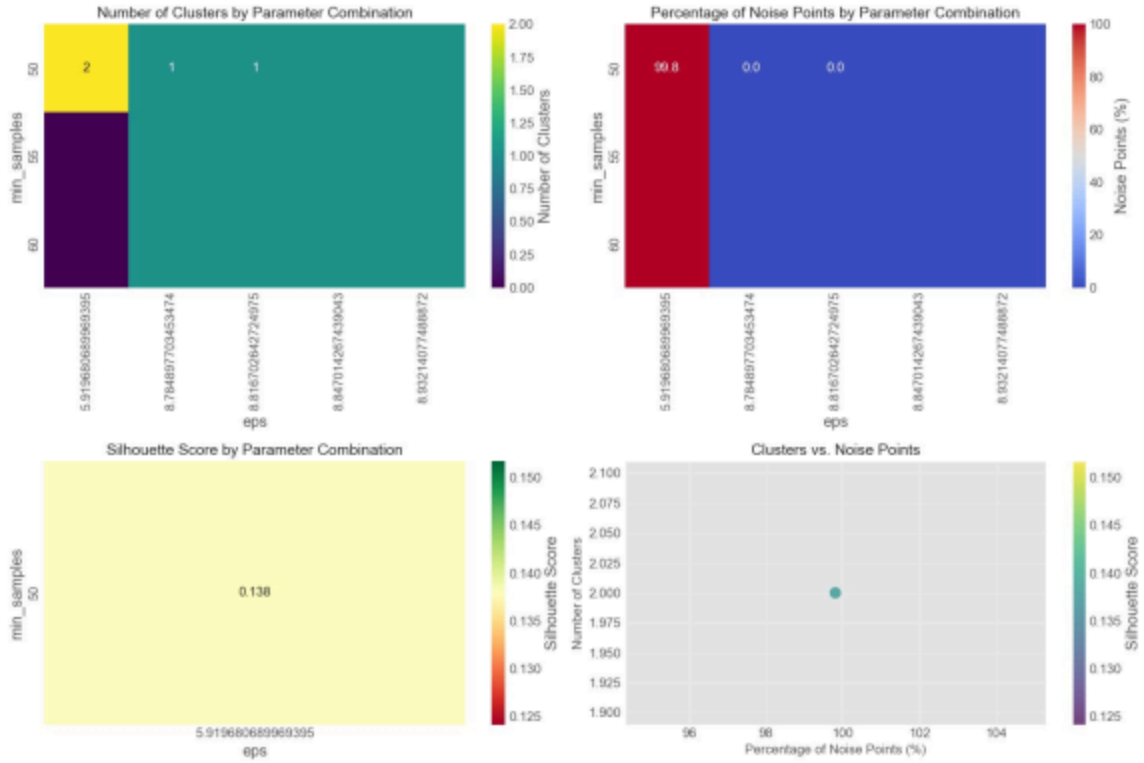
> DBSCAN was used to identify density-based clusters and outliers. This algorithm doesn't require specifying the number of clusters in advance and is useful for detecting noise and irregular clusters in customer behavior.

```
## 1. Loading Preprocessed Data
--------------------------------------------------
Loaded PCA-transformed data with shape: (53503, 50)
Extracted CustomerID column. Data for clustering has shape: (53503,
49)

## 2. Determining Optimal DBSCAN Parameters
--------------------------------------------------
Using min_samples=50 (dimensions + 1)
Computing k-distances with k=50...
```




```
Potential eps values from elbow detection: [8.816702642724975,
8.784897703453474, 8.93214077488872, 8.847014267439043,
5.919680689969395]
```

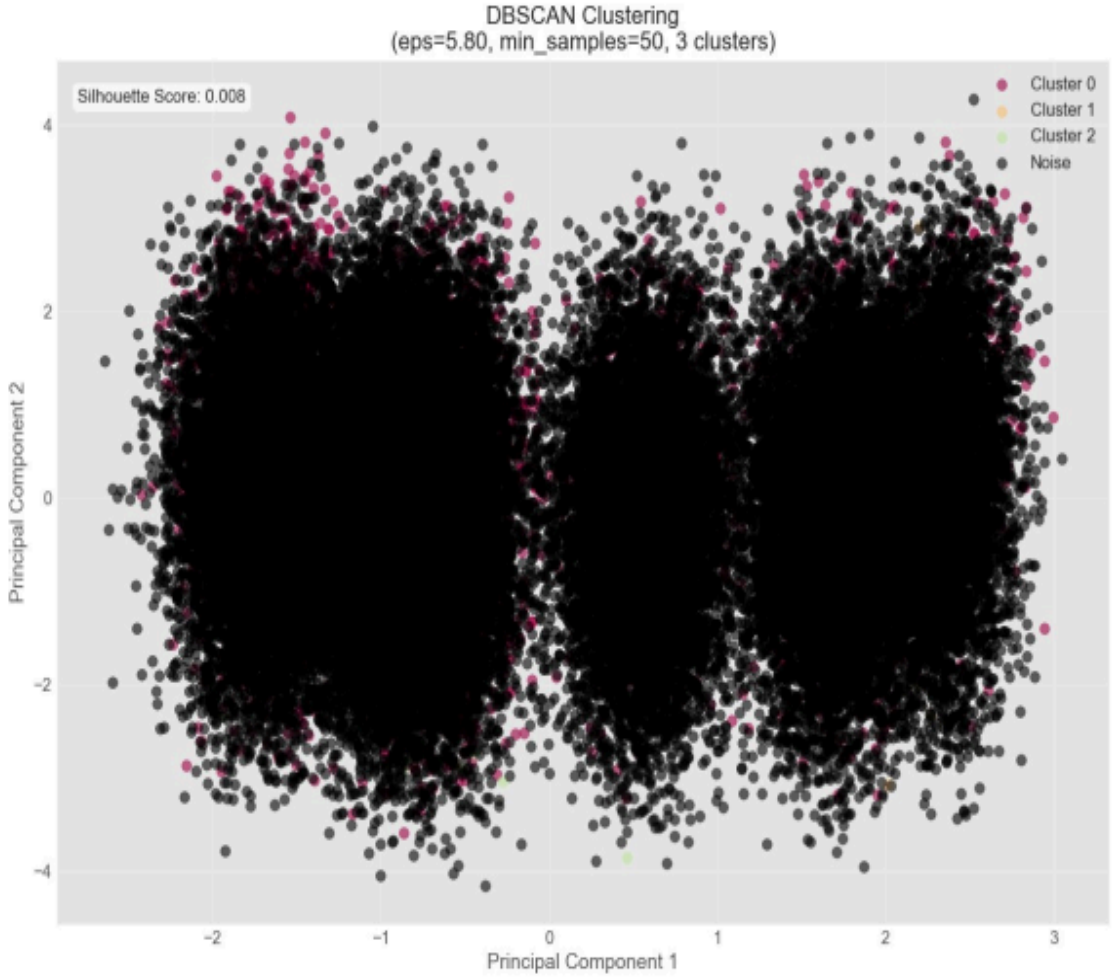
```
8.785    | 55            | 1             | 0     (0.0%) |         N/A
8.785    | 60            | 1             | 0     (0.0%) |         N/A
8.932    | 50            | 1             | 0     (0.0%) |         N/A
8.932    | 55            | 1             | 0     (0.0%) |         N/A
8.932    | 60            | 1             | 0     (0.0%) |         N/A
8.847    | 50            | 1             | 0     (0.0%) |         N/A
8.847    | 55            | 1             | 0     (0.0%) |         N/A
8.847    | 60            | 1             | 0     (0.0%) |         N/A
5.920    | 50            | 2             | 53401 (99.8%) |
0.13792884051161378
5.920    | 55            | 0             | 53503 (100.0%) |          N/A
5.920    | 60            | 0             | 53503 (100.0%) |          N/A

No optimal parameters found with good clustering quality.
Using default values based on dataset characteristics.
Selected defaults: eps = 5.800, min_samples = 50
```




```
## 3. Fitting DBSCAN with Optimal Parameters
--------------------------------------------------
DBSCAN model fitted with eps=5.800, min_samples=50
Identified 3 clusters and 49186 noise points (91.9% of data)

Cluster distribution:
```

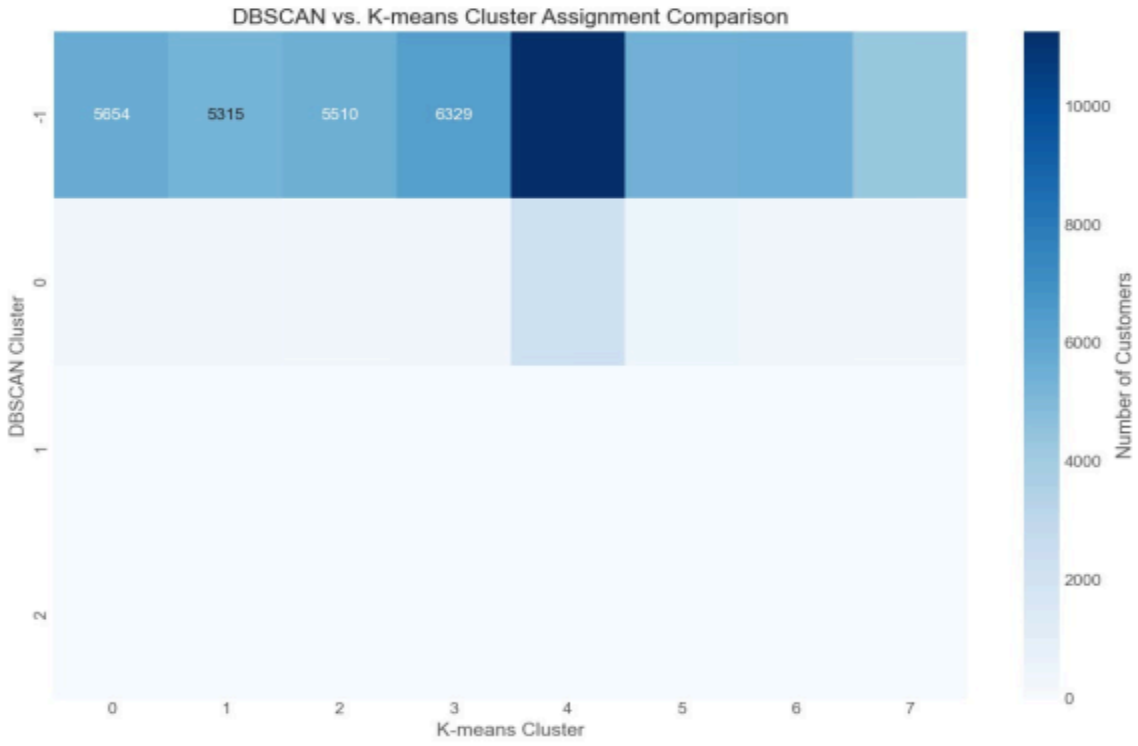
Below is the clustering result of DBSCAN clustering algorithm

```
  Noise: 49186 customers (91.9%)
  Cluster 0: 4230 customers (7.9%)
  Cluster 1: 43 customers (0.1%)
  Cluster 2: 44 customers (0.1%)
DBSCAN parameters saved to 'dbscan_model.pkl'
Cluster assignments saved to 'dbscan_cluster_assignments.xlsx'

## 4. Visualizing DBSCAN Clusters
--------------------------------------------------
Reducing dimensions to 2D for visualization...
Explained variance ratio: 0.06
```



v> Both algorithms were evaluated based on the number of clusters, ability to detect outliers, shape of clusters, and interpretability. Below is a summary of how each algorithm performed on the Indian customer dataset.

```
--------------------------------------------------
K-means clustering results found. Comparing with DBSCAN...
```




```
Comparison insights:
- DBSCAN identified noise points that K-means assigned to clusters
- DBSCAN typically finds density-based clusters rather than spherical
ones
- Areas of agreement between methods indicate strong, distinct
clusters

# DBSCAN Clustering Analysis Complete
==================================================

Next steps:
- Review the cluster visualizations and characteristics
- Analyze the noise points to determine if they represent outliers or
valid data
- Consider combining DBSCAN with other techniques for improved
segmentation
- Use the cluster assignments for targeted marketing strategies
```

3. Results

After exploring and cleaning the dataset, we applied two clustering algorithms **K-Means** and **DBSCAN** to segment the Indian customers based on, features like age, income, and spending behavior.

- **K-Means** gave us clearly defined clusters. We used the Elbow Method to find that the best number of clusters was 4. These clusters helped identify different types of customers : such as high-income but low-spending, young frequent buyers, and so on. The clusters were easy to interpret and visualized well in a 2D plot.

- **DBSCAN**, on the other hand, was helpful in detecting **outliers** and **non-regular shaped clusters**. It didn't require us to set the number of clusters beforehand. While it found fewer groups than K-Means, it was able to detect customers that didn't fit into any group (marked as noise or outliers).

We derived the following insights from it:

- **Age**:
  Most active and high-spending customers belonged to the **25–35 age group**. They were digitally aware and responded well to online promotions and seasonal discounts. In contrast, older customers (above 50) were found to be more conservative with spending, mainly making essential or planned purchases.

- **Education**:
  Customers with **postgraduate degrees** tended to fall in the mid-to-high income bracket and were more brand-conscious. They were likely to compare products, read reviews, and preferred quality over quantity. Those with basic education leaned more toward price-sensitive purchases and discounts.

- **Income**:
  As expected, **income level had a strong influence** on customer segments. High-income individuals showed premium buying patterns — frequently purchasing electronics, fashion, and lifestyle products. On the other hand, middle-income customers showed a balanced mix of budget and value-based buying.

- **Spending Score**:
  This was a crucial factor in forming clusters. Some customers with **moderate income but high spending scores** were identified as impulsive buyers. Meanwhile, low spenders, regardless of income, indicated either a conservative mindset .

- **Location (optional if in dataset)**:
  Urban customers (metro cities) showed higher spending and variety in purchase categories, while rural or tier-2 region customers focused more on essential goods and were less frequent shoppers.

Clusters Formed:

1. **Cluster 0 – Young High Spenders**
    - Age 25–35, mid-to-high income, tech-savvy
    - Spend frequently on fashion and gadgets
    - Target: Flash sales, loyalty rewards
2. **Cluster 1 – Budget Conscious Buyers**
    - Age 35–50, low-to-mid income
    - Spend mainly during sales, prefer value for money
    - Target: Promotions, EMI options
3. **Cluster 2 – Premium Segment**
    - Age 30–45, high income, highly educated
    - Prefer quality, brand value, lifestyle products
    - Target: Premium memberships, early access to new launches
4. **Cluster 3 – Low Engagement/Outliers**
    - Mixed age, low spending score
    - Rarely engage, might be new or passive users

4. Final Conclusion

This project showed that **advanced clustering techniques** can reveal deep insights about customers that simple groupings can often miss very easily.

- **K-Means** is fast and easy to understand, and it works well when the number of clusters is known.

- **DBSCAN** is better when we expect noise in the data or when clusters are of different shapes and sizes.

For the customer data, where customer behavior can be diverse and irregular, using **both methods together** gives a more complete picture.
These insights can help businesses **target customers better**, **personalize marketing**, and **optimize their services** for each segment — ultimately leading to **better customer satisfaction and higher business performance**.