CUSTOMER SEGMENTATION FOR A U.S. ONLINE COSMETICS RETAILER USING PYTHON-BASED CLUSTERING

1. Introduction

A growing U.S.-based DTC cosmetics brand operating primarily online wanted to optimize its marketing strategy by identifying distinct customer groups based on behavior. Their previous campaigns used generic segmentation (e.g., age or geography) and had plateaued in performance. The client sought a data-driven segmentation approach using unsupervised learning to identify natural customer clusters that could inform personalized promotions, retention strategies, and product bundling.

2. Objective

- To apply K-Means clustering using Python to segment customers based on behavior and value metrics
- To generate meaningful buyer personas using internal purchase data
- To deliver a model pipeline and strategic report that marketing and CRM teams could use without needing to understand code
- To visualize clusters in a clear, interpretable format for business decisions

3. Data Used

The dataset consisted of **9,750 customer records** from the previous 12 months of transactions on the company's Shopify storefront. Each row represented an individual customer and included:

- Total_Spend (USD)
- Avg_Basket_Size
- Purchase_Frequency (orders per month)
- Time_Since_Last_Purchase (in days)
- Preferred_Category (Makeup, Skincare, Haircare, Wellness)

Dummy variables were created for Preferred_Category. Additional features were engineered:

- Monetary_Value = Total_Spend

- Recency = Time_Since_Last_Purchase
- Frequency = Purchase_Frequency
- AOV (Average Order Value) = Total_Spend / Purchase_Frequency

4. Methodology

4.1 Data Preprocessing

- Normalized features using Min-Max Scaling to standardize across metrics
- One-hot encoded the product category preferences
- Applied the **Elbow Method and Silhouette Score** to identify the optimal number of clusters (chosen K=4)
- Visualized clusters in 2D using PCA for simplified interpretation

4.2 Model Development

- Used KMeans from sklearn.cluster
- Implemented elbow and silhouette scoring loops to test K from 2 to 10
- Segmentation was based on scaled values of Recency, Frequency, Monetary Value, AOV, and Category indicators

4.3 Tools Used

- Jupyter Notebook and Google Colab
- Libraries: pandas, numpy, matplotlib, seaborn, sklearn, plotly
- Delivered model and visuals in both .ipynb and .pdf formats

5. Key Results

- Final model identified **4 distinct customer clusters**:
    - **Cluster 1 – High Spenders**: Low frequency, high AOV, prefer Skincare
    - **Cluster 2 – Repeat Loyalists**: High frequency, mid AOV, prefer Makeup and Haircare
    - **Cluster 3 – Bargain Hunters**: Low AOV, high frequency, responded best to promotions
    - **Cluster 4 – Lapsed Users**: Low spend, long time since last purchase

- Visualizations included:
    - Clustered 2D PCA scatter plots with hover info
    - Radar charts showing average metrics per cluster
    - Distribution charts of category preferences per cluster

6. Delivered Report Summary

- **Deliverables**:
    - Clean Python code in .ipynb with embedded comments
    - PDF summary with cluster insights, charts, and action points
    - Excel file listing customer ID mapped to cluster label
    - Playbook: "How to Use This Segmentation in Klaviyo/CRM Tools"
- **Business Insights in Report**:
    - Suggested retargeting strategies per cluster (e.g., replenish, upsell, winback)
    - Identified opportunity to launch a skincare-specific premium line for Cluster 1
    - Highlighted importance of time-based re-engagement for Cluster 4 with automation triggers

7. Business Impact

- CRM and marketing teams used the clusters to personalize email campaigns
- A/B tests on Cluster 3 showed **22% uplift in promo email conversion rates**
- Cluster 2 was targeted with loyalty rewards and contributed to **9% repeat revenue increase in Q2**
- Segmentation was integrated with Shopify Plus + Klaviyo, allowing automated journeys by cluster
- Reduced CAC by **11%** by excluding low-LTV segments from retargeting ad budgets

8. Future Scope

- Introduce lifetime value prediction per cluster
- Integrate browsing behavior from GA4 or Shopify analytics to refine clusters

- Use Hierarchical Clustering or DBSCAN to test stability of cluster definitions
- Set up a retraining loop every 90 days using updated sales data
- Deploy as a backend microservice using Flask to push live cluster labels into CRM pipelines