TeamX

Customer Segmentation for E-commerce

Task 2 Documentation

March 10, 2025

1 Introduction

In this project, we aim to identify distinct customer segments in an e-commerce platform. By analyzing key behavioral metrics such as purchase frequency, average cart value, and discount usage, we can categorize customers into meaningful groups. These insights are crucial for targeted marketing, personalized recommendations, and strategic resource allocation.

Objectives:

- Develop a clustering model to group customers with similar behavioral patterns.
- Interpret the clusters and map them to known archetypes: *Bargain Hunters*, *High Spenders*, and *Window Shoppers*.
- Provide actionable recommendations for business strategies based on these segments.

2 Data Preprocessing

2.1 Dataset Overview

The dataset includes the following columns:

- `customer_id`: Unique identifier for each customer.
- `total_purchases`: Total number of purchases made by the customer.
- `avg_cart_value`: Average monetary value of items in each purchase.
- `total_time_spent`: Total time (in minutes) spent on the platform.
- `product_click`: Number of products viewed by the customer.
- `discount_counts`: Number of times a discount code was used.

2.2 Data Loading

We load the CSV file using `pandas` and inspect the first few rows:

Listing 1: Loading and previewing the dataset.

```
blueimport pandas as pd

df = pd.read_csv(purple'purplecustomer_datapurple.purplecsvpurple')
blueprint(df.head())
blueprint(df.info())
```

2.3 Handling Missing Values

An initial check (`df.isnull().sum()`) revealed missing values in `total_purchases`, `avg_cart_value`, and `product_click`. Instead of dropping rows, we use KNN imputation:

Listing 2: KNN Imputation.

```
bluefrom sklearn.impute blueimport KNNImputer

knn_imputer = KNNImputer(n_neighbors=5)
numeric_cols = [purple'purpletotal_purchasespurple', purple'
   purpleavg_cart_valuepurple',
purple                 purple'purpletotal_time_spentpurple', purple'
                     purpleproduct_clickpurple',
purple                 purple'purplediscount_countspurple']

df[numeric_cols] = knn_imputer.fit_transform(df[numeric_cols])
```

2.4 Feature Selection & Scaling

`customer_id` is excluded from clustering since it's an identifier. For distance-based algorithms like K-Means, we standardize numeric features:

Listing 3: Feature scaling with StandardScaler.

```
bluefrom sklearn.preprocessing blueimport StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(df[numeric_cols])
```

3 Exploratory Data Analysis (EDA)

3.1 Statistical Summaries

We use `df.describe()` to understand basic statistics (mean, median, quartiles) of each feature.

3.2 Visualizations

- **Histograms**: Show the distribution of each numeric feature.
- **Box Plots**: Help detect outliers in `avg_cart_value` or `total_purchases`.
- **Correlation Heatmap**: Reveals relationships among features, e.g., whether `discount_counts` correlates strongly with `avg_cart_value`.

Listing 4: Example: Correlation heatmap.

```
blueimport seaborn as sns
blueimport matplotlib.pyplot as plt

corr = df[numeric_cols].corr()
plt.figure(figsize=(8,6))
sns.heatmap(corr, annot=True, cmap=purple'purpleBluespurple')
plt.title(purple'purpleCorrelationpurple purpleHeatmappurple')
plt.show()
```

4 Clustering and Model Evaluation

4.1 K-Means Clustering

We apply K-Means with `n_clusters=3`, aligning with the known segments. We also compute a silhouette score to gauge how well-separated the clusters are.

Listing 5: K-Means clustering.

```
bluefrom sklearn.cluster blueimport KMeans
bluefrom sklearn.metrics blueimport silhouette_score

kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
cluster_labels = kmeans.fit_predict(X_scaled)

sil_score = silhouette_score(X_scaled, cluster_labels)
blueprint(purple"purpleSilhouettepurple purpleScorepurple:purple",
   sil_score)
```

4.2 Cluster Labeling

After clustering, we add a new column `cluster` to our DataFrame:

Listing 6: Assigning cluster labels to the DataFrame.

```
df[purple'purpleclusterpurple'] = cluster_labels
```

5 Cluster Interpretation & Customer Classification

5.1 Analyzing Mean Feature Values

We group by `cluster` and compute the average for each feature:

Listing 7: Cluster summary.

```
cluster_summary = df.groupby(purple'purpleclusterpurple')[numeric_cols
    ].mean()
blueprint(cluster_summary)
```

By comparing these mean values with the expected segment profiles:

- **Bargain Hunters:** High `total_purchases`, low `avg_cart_value`, moderate `product_click`, high `discount_counts`.
- **High Spenders:** Moderate `total_purchases`, high `avg_cart_value`, moderate `product_click`, low `discount_counts`.
- **Window Shoppers:** Low `total_purchases`, moderate `avg_cart_value`, high `product_click`, high `total_time_spent`, low `discount_counts`.

we map each cluster label (0, 1, 2) to a segment name and create a new column `segment`:

Listing 8: Mapping clusters to segment names.

```
cluster_mapping = {
    0: purple'purpleHighpurple purpleSpenderspurple',
    1: purple'purpleBargainpurple purpleHunterspurple',
    2: purple'purpleWindowpurple purpleShopperspurple'
}

df[purple'purplesegmentpurple'] = df[purple'purpleclusterpurple'].bluemap(
    cluster_mapping)
```

6 Visualization of Clusters and Classification

6.1 PCA Scatter Plot

To visualize high-dimensional data in 2D, we use PCA:

Listing 9: PCA for visualization.

```
bluefrom sklearn.decomposition blueimport PCA

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

plt.figure(figsize=(8,6))
bluefor cluster, label bluein cluster_mapping.items():
    cluster_points = X_pca[cluster_labels == cluster]
```

```
    plt.scatter(cluster_points[:, 0], cluster_points[:, 1], label=
       label)
plt.xlabel(purple'purplePrincipalpurple purpleComponentpurple purple1
   purple')
plt.ylabel(purple'purplePrincipalpurple purpleComponentpurple purple2
   purple')
plt.title(purple'purpleCustomerpurple purpleSegmentspurple
   purpleVisualizedpurple purpleusingpurple purplePCApurple')
plt.legend()
plt.show()
```

6.2 Centroid Table or Annotations

We can annotate centroids or create a table showing each cluster's dominant features:

Listing 10: Centroid annotation example.

```
blueimport numpy as np

bluefor cluster, label bluein cluster_mapping.items():
    centroid = np.mean(X_pca[cluster_labels == cluster], axis=0)
    plt.text(centroid[0], centroid[1], label,
             fontsize=12, weight=purple'purpleboldpurple',
             ha=purple'purplecenterpurple', va=purple'purplecenterpurple',
             bbox=bluedict(facecolor=purple'purplewhitepurple', alpha
                =0.6, edgecolor=purple'purpleblackpurple'))
```

7 Suggestions for Improvement

- **Feature Engineering:** Incorporate recency, frequency, monetary (RFM) features or additional browsing behavior metrics.
- **Alternative Clustering:** Explore hierarchical clustering, DBSCAN, or Gaussian Mixture Models to see if they yield better separation.
- **Evaluation Metrics:** Use the Calinski-Harabasz or Davies-Bouldin index alongside the silhouette score.
- **Business Integration:** Implement targeted campaigns or product recommendations for each segment to validate the real-world impact.

8 Reproducibility & Running Instructions

8.1 Pre-requisites

- **Python Environment:** Version 3.7 or later.
- **Dependencies:**

```
pip install pandas numpy scikit-learn matplotlib seaborn
```

8.2 Steps to Run the Program

1. **Download the Dataset:** Place `customer_data.csv` in your working directory.
2. **Open the Notebook or Script:** Use the provided Jupyter Notebook, structured as follows:
   - Data Loading and Preprocessing
   - Exploratory Data Analysis (EDA)
   - Clustering (K-Means) and Evaluation
   - Cluster Interpretation and Classification
   - Visualization (PCA-based plots and tables)
3. **Execute the Notebook Sequentially:** Run each cell step-by-step, verifying the outputs and charts.
4. **Review Results:** Observe clustering metrics (silhouette score) and the mapped segments in the dataset.
5. **Experiment:** Modify clustering parameters (`n_clusters`, `n_init`) or explore alternative algorithms to see if the silhouette score improves.

9 Libraries & Dependencies

- **pandas** for data manipulation.
- **NumPy** for numerical computations.
- **scikit-learn** for imputation, scaling, K-Means, and PCA.
- **matplotlib & seaborn** for data visualization and exploratory plots.

10 Conclusion

This documentation provides a comprehensive overview of the customer segmentation process. From data preprocessing (including KNN imputation) to clustering with K-Means and final segment mapping, each step ensures that insights are both data-driven and actionable. By following the reproducibility instructions, you can replicate our analysis, experiment with various clustering approaches, and ultimately derive deeper business insights to inform marketing and product strategies.