Customer Clustering Script Documentation

Title: Lookalike Model Documentation for eCommerce Transactions Dataset

Prepared by: Atul Kumar Suthar

Date: 26-jan-2025

Project: eCommerce Transactions Dataset Analysis

Overview

This script performs customer segmentation using K-Means clustering on transaction and customer data. The goal is to group customers based on their transaction behavior, enabling businesses to target specific segments more effectively.

Steps in the Script

1. Load Datasets

- The script reads two CSV files:
  - **Customers.csv**: Contains customer details.
  - **Transactions.csv**: Contains transaction data.

Code Snippet:

```
customers = pd.read_csv("Customers.csv")

transactions = pd.read_csv("Transactions.csv")
```

---

2. Merge and Aggregate Data

- Transactions are merged with customer data on CustomerID.
- Aggregates transaction metrics for each customer:
  - **TotalValue**: Sum of total transaction values.
  - **Quantity**: Total quantity purchased.

Code Snippet:

```
merged = pd.merge(transactions, customers, on="CustomerID")

customer_data = merged.groupby("CustomerID").agg({

   'TotalValue': 'sum',  # Total transaction value

   'Quantity': 'sum',    # Total quantity purchased

}).reset_index()
```

---

3. Scale Data

- Standardizes numerical features using StandardScaler to ensure all features contribute equally to clustering.

Code Snippet:

```
scaler = StandardScaler()

scaled_data = scaler.fit_transform(customer_data.iloc[:, 1:])
```

---

4. Apply K-Means Clustering

- K-Means is used to partition customers into 5 clusters.
- The Cluster label is assigned to each customer.

Code Snippet:

```
kmeans = KMeans(n_clusters=5, random_state=42)

customer_data['Cluster'] = kmeans.fit_predict(scaled_data)
```

---

5. Evaluate Clustering

- The Davies-Bouldin Index (DBI) is calculated to evaluate cluster compactness and separation. Lower values indicate better clustering.

Code Snippet:

```
db_index = davies_bouldin_score(scaled_data, customer_data['Cluster'])

print(f"Davies-Bouldin Index: {db_index:.2f}")
```

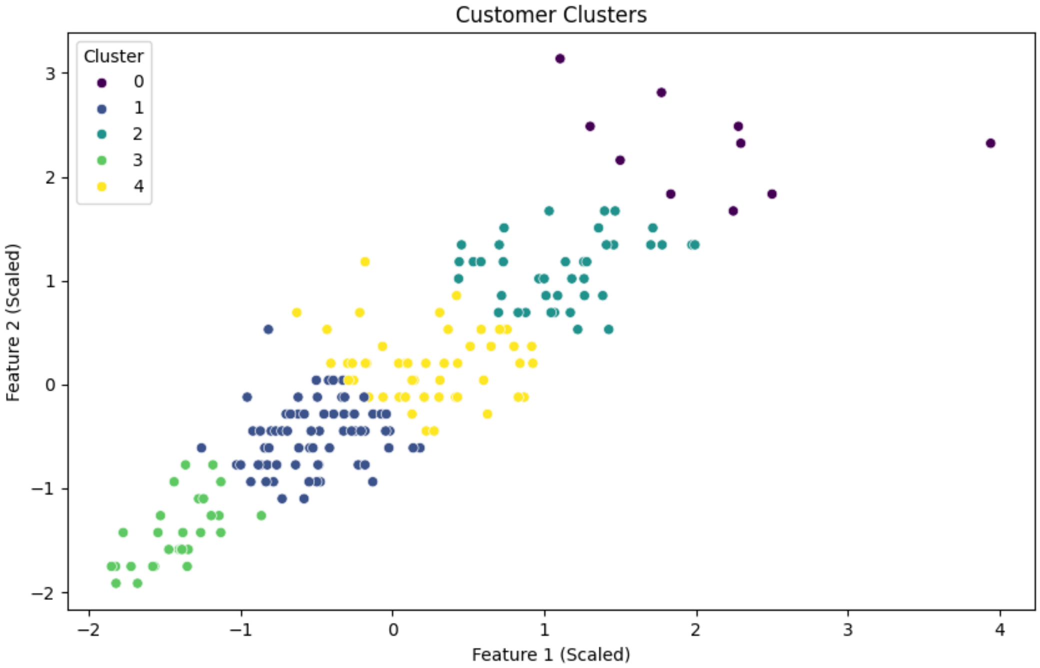
---

6. Visualize Clusters

- A scatter plot visualizes the clusters in two dimensions using the scaled features.
- Clusters are color-coded for easy interpretation.

Code Snippet:

```
plt.figure(figsize=(10, 6))
sns.scatterplot(
  x=scaled_data[:, 0], y=scaled_data[:, 1],
  hue=customer_data['Cluster'], palette="viridis"
)
plt.title("Customer Clusters")
plt.xlabel("Feature 1 (Scaled)")
plt.ylabel("Feature 2 (Scaled)")
plt.legend(title="Cluster")
plt.show()
```



---

7. Save Results

- Cluster assignments are saved to a CSV file named Customer_Clusters.csv.

Code Snippet:

```
customer_data.to_csv("Customer_Clusters.csv", index=False)

print("Cluster assignments saved to Customer_Clusters.csv.")
```

Output

- **Davies-Bouldin Index**: Printed to evaluate clustering quality.
- **Visualization**: Scatter plot of customer clusters.
- **CSV File**: Customer_Clusters.csv contains:
    - **CustomerID**: Unique customer identifier.
    - **TotalValue**: Sum of total transaction values.
    - **Quantity**: Total quantity purchased.
    - **Cluster**: Assigned cluster label.

Sample Output (Customer_Clusters.csv):

```
CustomerID,TotalValue,Quantity,Cluster

C001,1500.00,20,0

C002,1200.00,15,1

C003,1800.00,25,2
```

Features and Benefits

1. **Scalable Data Processing**: Handles large datasets efficiently with Pandas.
2. **Standardized Features**: Ensures fair clustering by normalizing features.
3. **Evaluation Metric**: Uses Davies-Bouldin Index for cluster validation.
4. **Actionable Insights**: Visualizes clusters for easy interpretation.

How to Run

1. Place Customers.csv and Transactions.csv in the same directory as the script.
2. Run the script:
3. python customer_clustering.py
4. Check the output visualization and the Customer_Clusters.csv file for results.