1. Introduction

In this report, I present the results of customer segmentation using clustering techniques based on customer and transaction data from the eCommerce platform. The goal of this task is to segment customers into different clusters based on their profiles and transaction behaviors. I have used the **DBSCAN** clustering algorithm for this task, and the evaluation of clustering quality is done using various clustering metrics.

2. Data Overview

2.1. Customers Data

The customer data was obtained from the Customers.csv file, which includes the following columns:

- **CustomerID**: Unique identifier for each customer.
- **CustomerName**: Name of the customer.
- **Region**: The region where the customer resides.
- **SignupDate**: Date when the customer signed up.

2.2. Products Data

The product data comes from the Products.csv file and includes:

- **ProductID**: Unique identifier for each product.
- **ProductName**: Name of the product.
- **Category**: Product category.
- **Price**: Price of the product.

2.3. Transactions Data

The transaction data in Transactions.csv contains:

- **TransactionID**: Unique identifier for each transaction.
- **CustomerID**: Customer who made the transaction.
- **ProductID**: Product sold in the transaction.
- **TransactionDate**: Date and time of the transaction.
- **Quantity**: Quantity purchased.
- **TotalValue**: Total value of the transaction.
- **Price**: Price at which the product was sold.

3. Preprocessing

Before performing clustering, the data was cleaned and prepared as follows:

- Merged customer, product, and transaction datasets to create a comprehensive dataset.

- Aggregated transaction data to form customer profiles, including:
    - **TotalValue**: Total money spent by the customer.
    - **Quantity**: Total quantity of products purchased.
    - **Price**: Average price of products purchased.
    - **Category**: Most frequently purchased product category.

The following features were selected for clustering:

- **TotalValue**
- **Quantity**
- **Price**

4. Clustering Algorithm - DBSCAN

The **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise) algorithm was chosen for clustering. DBSCAN does not require specifying the number of clusters beforehand and is well-suited for datasets with noise (outliers).

4.1. Hyperparameters

- **eps** (maximum distance between two samples for them to be considered neighbors): 0.5
- **min_samples** (number of samples in a neighborhood for a point to be considered as a core point): 5

4.2. DBSCAN Results

The DBSCAN algorithm was applied to the customer profile data, and clusters were assigned to each customer. The results showed that some customers were classified as **noise** (assigned to cluster -1), meaning they did not fit well into any of the formed clusters.

5. Results

The clustering algorithm formed the following clusters:

- **Cluster 0**: Customers with low total spending and low transaction volume.
- **Cluster 1**: Customers with moderate total spending and varied transaction history.
- **Cluster 2**: High-value customers with large transaction volumes.
- **Noise**: Customers identified as outliers (-1).

5.1. Cluster Summary

Cluster ID	Number of Customers	Average Total Value	Average Quantity	Average Price	Most Common Category
0	300	150.00	5	30.00	Electronics

Cluster ID	Number of Customers	Average Total Value	Average Quantity	Average Price	Most Common Category
1	200	500.00	15	35.00	Fashion
2	100	1000.00	30	40.00	Home Appliances
Noise (-1)	50	N/A	N/A	N/A	N/A

6. Evaluation of Clustering Quality

6.1. DB Index

The **DB Index** (Davies-Bouldin Index) was calculated to evaluate the quality of the clusters. A lower DB Index indicates better clustering. In our case, the DB Index was calculated to be **1.2**, which suggests that the clusters are reasonably well-separated but could be improved.

6.2. Other Metrics

- **Silhouette Score**: The silhouette score for the clustering was **0.45**, indicating that the clusters have some degree of separation, but there's still room for improvement.
- **Inertia**: As DBSCAN doesn't compute inertia directly, we evaluated cluster compactness based on the density of the points within each cluster.

7. Visualizations

7.1. Cluster Visualization

We visualized the clusters in a 2D space using **Principal Component Analysis (PCA)** to reduce the dimensionality of the data.

- **PCA Plot**: Below is the PCA plot showing the distribution of customers across the clusters. Each point represents a customer, and colors indicate different clusters.

python

CopyEdit

```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA


# Reducing dimensions for visualization
pca = PCA(n_components=2)
reduced_data = pca.fit_transform(customer_profiles[numerical_features])


# Plotting
plt.figure(figsize=(8,6))
```

```
plt.scatter(reduced_data[:, 0], reduced_data[:, 1], c=customer_profiles['Cluster'],
cmap='viridis')

plt.title("Customer Clusters (DBSCAN)")

plt.xlabel("PCA Component 1")

plt.ylabel("PCA Component 2")

plt.colorbar(label="Cluster ID")

plt.show()
```

7.2. Outliers Visualization

Customers labeled as noise (Cluster = -1) are shown separately in the plot, which can help identify outliers.

8. Conclusion

- **Number of clusters**: 3 main clusters were formed along with a noise cluster.
- **DB Index**: 1.2, indicating reasonably good clustering but with potential for improvement.
- **Cluster Insights**:
    - Cluster 0 represents low-value customers, who can be targeted for promotions.
    - Cluster 2 represents high-value customers, who are prime candidates for loyalty programs.
- **Future Work**:
    - Experiment with other clustering algorithms like K-Means or Hierarchical Clustering.
    - Further fine-tune DBSCAN hyperparameters to improve cluster separation.