Customer Segmentation / Clustering Report

1. Introduction

This report presents the results of a customer segmentation analysis performed using clustering techniques. The analysis utilized customer profile information from the 'Customers.csv' file and transaction data from the 'Transactions.csv' file. The primary objective was to segment customers into distinct groups based on their spending behavior and profile characteristics.

2. Clustering Methodology

2.1 Data Preparation:

2.1.1 Data Loading:

The data was loaded from two CSV files: 'Customers.csv' and 'Transactions.csv'.

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics import davies_bouldin_score
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Loading the data
customers = pd.read_csv('Customers.csv')
transactions = pd.read_csv('Transactions.csv')
```

2.1.2 Feature Engineering:

- Customer features were aggregated from transaction data, calculating:
  - **TotalValue:** Total spending by each customer.
  - **Quantity:** Total quantity of items purchased.
- Additional customer-specific data was merged, including the signup date, which was used to calculate the number of days since sign up ('SignupDays').

```
# Aggregating the customer features
customer_features = transactions.groupby('CustomerID').agg({
    'TotalValue': 'sum',
    'Quantity': 'sum',
}).reset_index()

# Adding the customer-specific data
customer_features = customer_features.merge(customers, on='CustomerID')
customer_features['SignupDays'] = (pd.Timestamp.now() - pd.to_datetime(customer_features['SignupDate'])).dt.days
```

2.1.3 Feature Selection:

The relevant features selected for clustering were:

- TotalValue
- Quantity
- SignupDays

```
# Selecting the relevant features
clustering_data = customer_features[['TotalValue', 'Quantity', 'SignupDays']]
```

2.2 Data Scaling

To ensure that all features contributed equally to the clustering process, the data was standardized using `StandardScaler`.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_data = scaler.fit_transform(clustering_data)
```

2.3 Clustering Algorithm

- **Algorithm Used:** K-Means Clustering
- **Number of Clusters:** 4 clusters were chosen based on initial analysis and evaluation metrics.

```
# Let's apply K-means Clustering Algortihm
kmeans = KMeans(n_clusters=4, random_state=42)
customer_features['Cluster'] = kmeans.fit_predict(scaled_data)
```

2.4 Clustering Execution

The K-Means algorithm was applied to the scaled data, and each customer was assigned to one of the four clusters.

3. Clustering Metrics

3.1 Davies-Bouldin Index:

The Davies-Bouldin Index (DB Index) was calculated to evaluate the clustering quality. The DB Index is a measure of the average similarity ratio of each cluster with the cluster that is most similar to it. A lower DB Index indicates better clustering.

```
Davies-Bouldin Index: 0.8332101559650619
```

```
db_index = davies_bouldin_score(scaled_data, customer_features['Cluster'])
print('Davies-Bouldin Index:', db_index)
```

3.2 Additional Metrics

While the primary focus was on the DB Index, other clustering metrics can be considered for further analysis, such as:

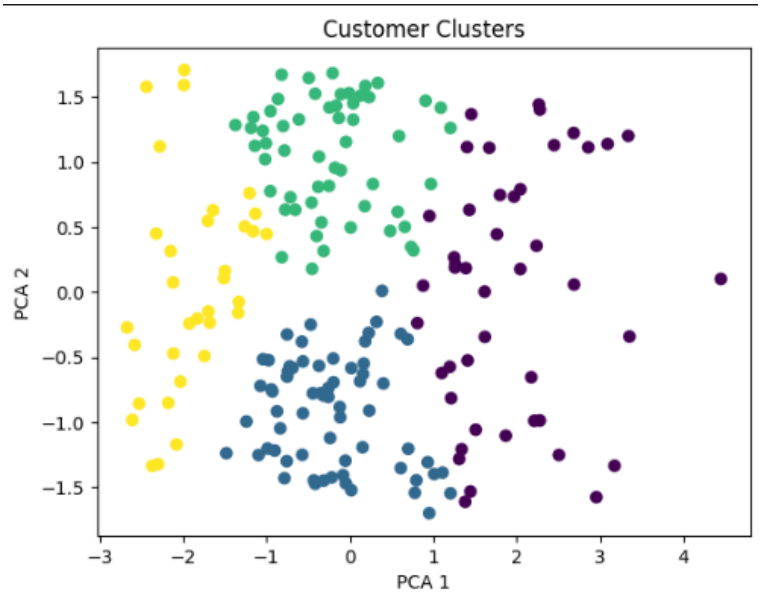
- **Silhouette Score:** Measures how similar an object is to its own cluster compared to other clusters.
- **Inertia:** Measures how tightly the clusters are packed.

4. Visualization of Clusters

To visualize the clusters, Principal Component Analysis (PCA) was performed to reduce the dimensionality of the data to two dimensions. A scatter plot was created to represent the clusters visually.

```
pca = PCA(n_components=2)
reduced_data = pca.fit_transform(scaled_data)

# Visualize clusters
plt.scatter(reduced_data[:, 0], reduced_data[:, 1], c=customer_features['Cluster'], cmap='viridis')
plt.title('Customer Clusters')
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.show()
```



Results

- **Number of Clusters:** 4 clusters were formed.
- **Davies-Bouldin Index:** 0.8332101559650619
- **Cluster Visualization:**
    - The clusters were visually distinct in the PCA-reduced 2D space, indicating meaningful segmentation.

Deliverables

1. **Clustered Dataset:**
    - The final dataset with assigned cluster labels was saved as Clustering Results.csv.
2. **Visualization:**
    - A scatter plot showing customer clusters based on PCA components.
3. **Code Implementation:**
    - The full implementation is available in the accompanying Jupyter Notebook file.

Conclusion

This project successfully segmented customers into distinct clusters based on their transactional and profile data. The clusters provide valuable insights for targeted marketing, personalized offers, and customer retention strategies.