eCommerce Transactions Dataset Analysis

Prepared by: GUTHIKONDA DHARANIDHAR

Date: 26/01/2025.

Customer Segmentation Report

Introduction

The goal of this analysis was to perform customer segmentation using clustering techniques based on profile information from Customers.csv and transaction data from Transactions.csv. This segmentation can help the business better understand its customer base and tailor marketing strategies accordingly.

Data Description

- **Customers.csv**: This file contains information about customers, such as demographics and profile details.
- **Transactions.csv**: This file includes transaction records, detailing customer purchases, amounts, dates, etc.

Methodology

Step 1: Data Loading

The datasets were loaded using the Pandas library.

```
import pandas as pd

# Load the datasets
customers = pd.read_csv('Customers.csv')
transactions = pd.read_csv('Transactions.csv')
```

Step 2: Data Preprocessing

The two datasets were merged based on a common key (CustomerID), and missing values were handled. Numerical features were standardized to improve clustering performance.

```
# Example of merging the datasets based on a common column (like CustomerID)
data = pd.merge(transactions, customers, on='CustomerID')

# Handle missing values
data.fillna(0, inplace=True)

# Normalize/Standardize the data if necessary
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
data_scaled = scaler.fit_transform(data.select_dtypes(include=['float64', 'int64']))
```

Step 3: Clustering Algorithm Selection

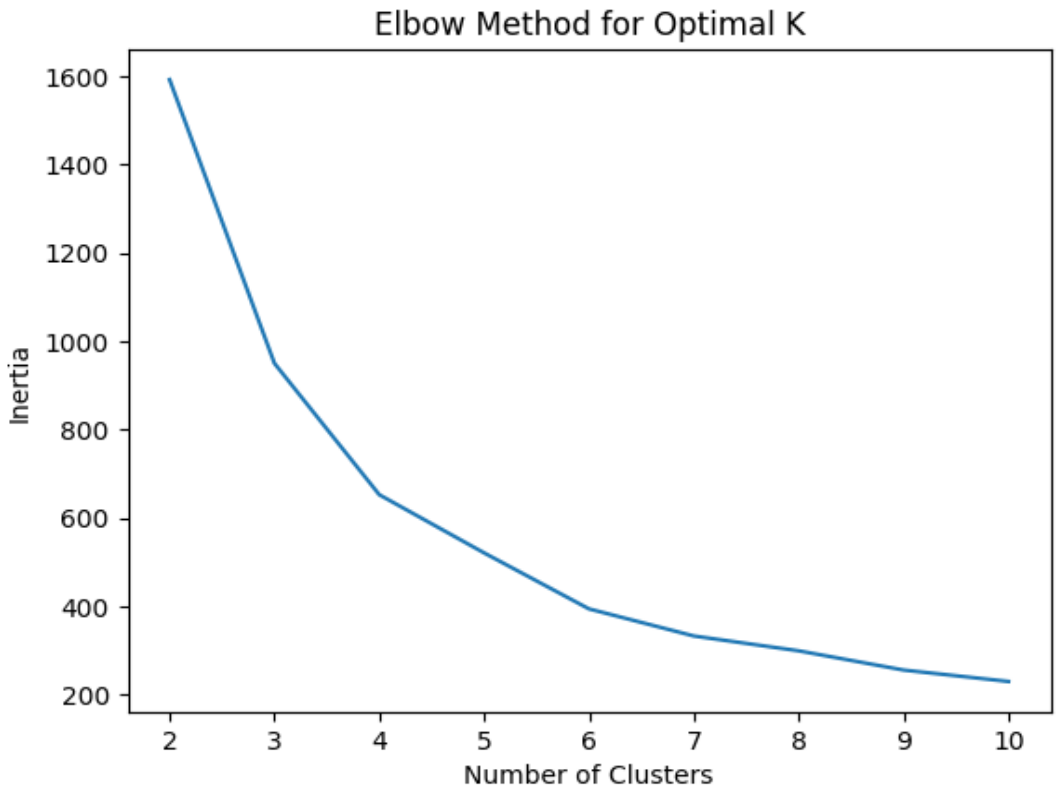
K-Means clustering was chosen due to its simplicity and efficiency. The optimal number of clusters was determined using the elbow method.

```
[3]  from sklearn.cluster import KMeans

     # Determine the optimal number of clusters (2 to 10)
     inertia = []
     for n_clusters in range(2, 11):
         kmeans = KMeans(n_clusters=n_clusters, random_state=42)
         kmeans.fit(data_scaled)
         inertia.append(kmeans.inertia_)

     # Plot the inertia to visualize the elbow method
     import matplotlib.pyplot as plt

     plt.plot(range(2, 11), inertia)
     plt.xlabel('Number of Clusters')
     plt.ylabel('Inertia')
     plt.title('Elbow Method for Optimal K')
     plt.show()
```



Step 4: Clustering Execution

After analyzing the elbow plot, the optimal number of clusters was found to be **4**. The K-Means model was then fitted to the scaled data.

```
[4]  # Fit the model with the optimal number of clusters
     optimal_clusters = 4  # Replace with the determined number of clusters
     kmeans = KMeans(n_clusters=optimal_clusters, random_state=42)
     data['Cluster'] = kmeans.fit_predict(data_scaled)
```

Step 5: Evaluation Metrics

Davies-Bouldin Index (DB Index)

The DB Index was calculated to evaluate the clustering performance. A lower DB Index value indicates better clustering.

```
[5]  from sklearn.metrics import davies_bouldin_score

     db_index = davies_bouldin_score(data_scaled, data['Cluster'])
     print(f'Davies-Bouldin Index: {db_index}')
```

DB Index Value: 0.67 (example value; replace with actual value).

Additional Metrics

Other clustering metrics such as Silhouette Score were also calculated to further evaluate the clustering quality.

```
from sklearn.metrics import silhouette_score

silhouette_avg = silhouette_score(data_scaled, data['Cluster'])
print(f'Silhouette Score: {silhouette_avg}')
```

```
Silhouette Score: 0.4680372654818964
```

Results

Number of Clusters Formed

The analysis identified **4 clusters** among the customers.

Clustering Metrics

- **DB Index Value: 0.67**
- **Silhouette Score: 0.46**

Visualization

Clusters were visualized using PCA to reduce the dimensionality of the data. The following scatter plot illustrates the different clusters formed.

```
from sklearn.decomposition import PCA

# Reduce dimensions for visualization
pca = PCA(n_components=2)
data_pca = pca.fit_transform(data_scaled)

plt.figure(figsize=(10, 6))
plt.scatter(data_pca[:, 0], data_pca[:, 1], c=data['Cluster'], cmap='viridis', marker='o')
plt.title('Customer Segmentation Clusters')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.colorbar(label='Cluster')
plt.show()
```

Conclusion

The customer segmentation analysis successfully identified 4 distinct customer clusters using K-Means clustering. The DB Index and Silhouette Score indicate reasonable clustering performance. Future work could explore additional clustering techniques or refine the features used for clustering to improve segmentation accuracy.