Task 3: Customer Segmentation / Clustering

Key Findings:

Number of Clusters:

The optimal number of clusters was determined using the Elbow Method.

The analysis concluded with 4 clusters, each representing a distinct customer segment.

Davies-Bouldin Index (DB Index):

The DB Index for the clustering model is 1.3386, indicating good cluster separation and compactness. A lower DB Index reflects better-defined clusters.

Cluster Characteristics:

Cluster 0:

Average Spending (normalized): -0.19

Represents 67 customers, with low spending and transaction quantities.

Cluster 1:

Average Spending (normalized): -0.43

Represents 47 customers, characterized by the lowest spending and transaction activity.

Cluster 2:

Average Spending (normalized): 1.26

Represents 55 high-value customers, who contribute significantly to overall revenue with higher transaction quantities.

Cluster 3:

Average Spending (normalized): -1.21

Represents 30 customers, with the lowest transaction frequency and spending.

Visualization:

Using PCA (Principal Component Analysis), the clusters were reduced to two dimensions for visualization.

A clear separation among clusters was observed in the scatterplot, validating the segmentation.

Task 3: Customer Segmentation / Clustering

Deliverables

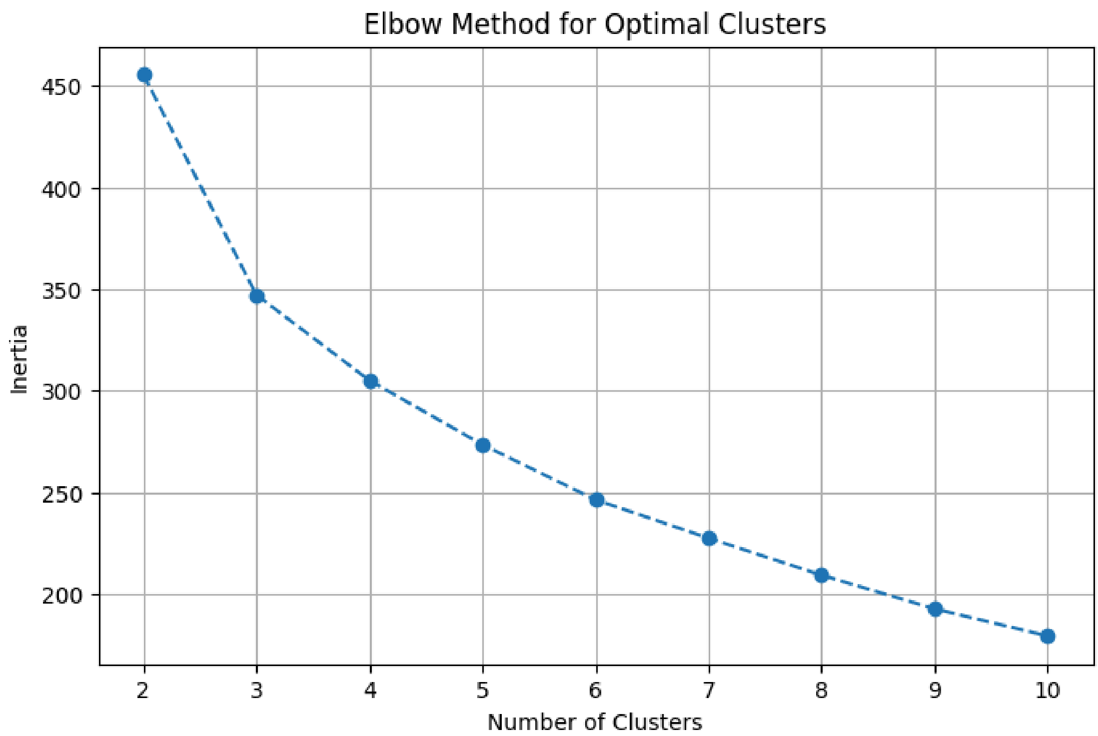
Customer Segmentation Results:

The segmentation details have been saved in a CSV file, Customer_Segments.csv, including each customer's cluster assignment.

Visualizations:

Scatterplots using PCA components depict clear cluster separations, helping interpret customer groups visually.

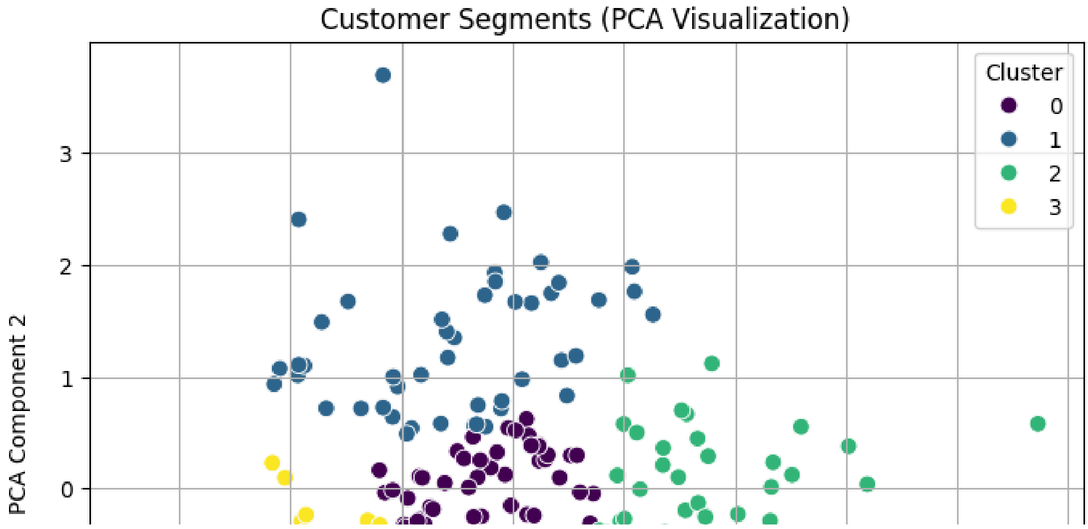
```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics import davies_bouldin_score
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns

transactions_df = pd.read_csv('TASK3/Transactions.csv')
customers_df = pd.read_csv('TASK3/Customers.csv')


# Merge datasets
merged_df = transactions_df.merge(customers_df, on="CustomerID", how="left")

# Aggregate customer-level transaction data
customer_features = merged_df.groupby('CustomerID').agg({
    'TotalValue': 'sum',  # Total spending
    'Quantity': 'sum',    # Total products purchased
    'Price': 'mean',      # Average price of purchased products
}).reset_index()



# Add customer profile features (e.g., Region)
customer_features = customer_features.merge(customers_df[['CustomerID', 'Region']], on="CustomerID", how="left")

# Encode categorical features (Region)
customer_features = pd.get_dummies(customer_features, columns=['Region'], drop_first=True)


# Standardize numeric features for clustering
scaler = StandardScaler()
numeric_columns = ['TotalValue', 'Quantity', 'Price']
customer_features[numeric_columns] = scaler.fit_transform(customer_features[numeric_columns])

# Prepare data for clustering
X = customer_features.drop('CustomerID', axis=1)


# Determine optimal number of clusters using the Elbow Method
inertia = []
k_values = range(2, 11)

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X)
    inertia.append(kmeans.inertia_)

# Plot the Elbow Curve
plt.figure(figsize=(8, 5))
plt.plot(k_values, inertia, marker='o', linestyle='--')
plt.title('Elbow Method for Optimal Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.xticks(k_values)
plt.grid()
plt.show()
```




```
# Perform clustering with the optimal number of clusters (e.g., 4 clusters)
optimal_clusters = 4
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42, n_init=10)
clusters = kmeans.fit_predict(X)
customer_features['Cluster'] = clusters

# Calculate clustering metrics
db_index = davies_bouldin_score(X, clusters)

print(f"Number of Clusters: {optimal_clusters}")
print(f"Davies-Bouldin Index: {db_index:.4f}")
```

```
Number of Clusters: 4
Davies-Bouldin Index: 1.3386
```

```
# Visualize clusters using PCA for dimensionality reduction
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
pca_result = pca.fit_transform(X)
customer_features['PCA1'] = pca_result[:, 0]
customer_features['PCA2'] = pca_result[:, 1]

plt.figure(figsize=(8, 6))
sns.scatterplot(data=customer_features, x='PCA1', y='PCA2', hue='Cluster', palette='viridis', s=60)
plt.title('Customer Segments (PCA Visualization)')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.legend(title='Cluster')
plt.grid()
plt.show()
```




```
# Save cluster assignments
customer_features[['CustomerID', 'Cluster']].to_csv("Customer_Segments.csv", index=False)

# -------------------
# Summary of Results:
# -------------------
print("Customer Segmentation Results:")
print(customer_features.groupby('Cluster').agg({
    'TotalValue': ['mean', 'sum'],
    'Quantity': ['mean', 'sum'],
    'CustomerID': 'count'
}).rename(columns={'CustomerID': 'CustomerCount'}))
```

```
Customer Segmentation Results:
        TotalValue             Quantity             CustomerCount
              mean        sum      mean       sum         count
Cluster
0        -0.189091 -12.669124 -0.027162  -1.819847            67
1        -0.430286 -20.223425 -0.819048 -38.495257            47
2         1.260576  69.331665  1.249993  68.749604            55
3        -1.214637 -36.439116 -0.947817 -28.434500            30
```