Assignment 2

Akhigbe Paulinus - 0820802

Introduction

Cluster analysis is a statistical method used to group similar objects or data points into clusters based on their characteristics or features. The goal is to ensure that items within the same cluster are more similar to each other than to those in other clusters. This technique is widely used in fields such as data mining, machine learning, and pattern recognition, helping researchers and analysts uncover hidden patterns, relationships, and structures within large datasets.

The significance of cluster analysis lies in its ability to simplify complex data by organizing it into meaningful groups, enabling better insights and decision-making. For example, in marketing, cluster analysis can help identify distinct customer segments, allowing companies to tailor their products and services to meet specific needs. In biology, it can group genes or species based on similarities, aiding in the understanding of evolutionary relationships. Overall, cluster analysis is a powerful tool for exploring data, identifying trends, and informing strategies across various domains.

Dataset Description

The Wholesale Customers Dataset will be used to analyze purchasing patterns of various wholesale customers. This dataset includes information on the annual spending of different customers on various product categories. It provides insights into customer segmentation based on spending behavior across multiple product types. Here's a summary of the key attributes:

Attributes:

Channel: Specifies the type of customer channel (e.g., Horeca - Hotel/Restaurant/Café or Retail). This attribute acts as a categorical label but is usually not involved in clustering.

Region: Indicates the geographical region where the customer is located (e.g., Lisbon, Oporto, or other regions).

Fresh: Annual spending on fresh produce (e.g., fruits, vegetables).

Milk: Annual spending on milk products.

Grocery: Annual spending on grocery items.

Frozen: Annual spending on frozen products.

Detergents_Paper: Annual spending on cleaning products such as detergents and paper.

Delicatessen: Annual spending on delicatessen items (specialty foods, high-quality prepared foods).

Class Labels:

The dataset doesn't have pre-defined class labels as it's primarily designed for unsupervised learning (e.g., clustering). However, the Channel and Region columns can provide useful context for interpreting clusters or validating the segmentation by customer type or location.

Descriptive statistics for the dataset

AS shown in fig 1, the summary statistics, including mean, standard deviation, minimum, and maximum values, reveal the data distribution and variability within each category. For instance, 'Detergents_Paper' has a large range, indicating varied spending levels. Additionally, there are no missing values in any column, confirming that the dataset is complete and ready for further analysis.

```
First few rows of the dataset:
   Channel  Region  Fresh  Milk  Grocery  Frozen  Detergents_Paper  Delicassen
0        2       3  12669  9656     7561     214              2674        1338
1        2       3   7057  9810     9568    1762              3293        1776
2        2       3   6353  8808     7684    2405              3516        7844
3        1       3  13265  1196     4221    6404               507        1788
4        2       3  22615  5410     7198    3915              1777        5185

Descriptive Statistics:
          Channel      Region  ...  Detergents_Paper    Delicassen
count  440.000000  440.000000  ...        440.000000    440.000000
mean     1.322727    2.543182  ...       2881.493182   1524.870455
std      0.468052    0.774272  ...       4767.854448   2820.105937
min      1.000000    1.000000  ...          3.000000      3.000000
25%      1.000000    2.000000  ...        256.750000    408.250000
50%      1.000000    3.000000  ...        816.500000    965.500000
75%      2.000000    3.000000  ...       3922.000000   1820.250000
max      2.000000    3.000000  ...      40827.000000  47943.000000

[8 rows x 8 columns]

Missing Values:
Channel             0
Region              0
Fresh               0
Milk                0
Grocery             0
Frozen              0
Detergents_Paper    0
Delicassen          0
dtype: int64
```

Figure 1: Fig. 1 : Descriptive statistics

Methodology

Preprocessing Steps

Before applying cluster analysis, the data underwent several preprocessing steps to ensure quality and enhance the accuracy of clustering:

Data Cleaning:

I began by checking for missing values in the dataset. No missing values were found, indicating that the data was complete. Following the descriptive analysis, the dataset was confirmed to be in good shape and ready for further analysis.

Normalization/Standardization:

To prevent features with larger ranges from dominating the clustering results, numerical features were normalized to a standard scale, typically using standardscaler.

Feature Selection:

Redundant or irrelevant features were removed to reduce noise and computational complexity. Principal Component Analysis (PCA) was applied if dimensionality reduction was required, retaining components that explained the majority of the variance.

Algorithms Chosen

For this cluster analysis, three primary algorithms were selected based on the data characteristics and clustering objectives:

K-Means Clustering:

This popular algorithm was chosen for its simplicity and efficiency in handling large datasets. K-Means partitions data into K clusters by minimizing the within-cluster variance. The optimal number of clusters was determined using the Elbow Method and Silhouette Score.

Hierarchical Clustering:

This algorithm was used to provide a dendrogram, which visually represents the hierarchical relationships between data points. Agglomerative clustering was chosen with Ward's linkage method, which minimizes the variance within clusters. This approach was helpful for identifying the underlying structure of the data without specifying the number of clusters in advance.

DBSCAN Clustering:

The DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm was applied to identify clusters based on the density of data points, allowing for the detection of irregularly shaped clusters and outliers. Unlike K-means or hierarchical clustering, DBSCAN does not require a predefined number of clusters. Instead, it uses two key parameters—eps (maximum distance between points in a cluster) and min_samples (minimum points required to form a cluster)—to form dense regions. This method was effective for capturing clusters of varying shapes and isolating noise points in the dataset.

Modifications Made

To enhance the clustering performance and tailor the algorithms to our specific dataset, the following modifications were made:

Optimal Cluster Number Selection:

For K-Means, in addition to the Elbow Method, the Silhouette Score was used to validate the number of clusters, providing a more data-driven approach to cluster selection.

Hybrid Approach:

In cases where K-Means alone did not capture the data structure effectively, hierarchical clustering was applied to the K-Means results, combining the strengths of both methods.

Distance Metric Adjustment:

Euclidean distance was used for K-Means clustering, while hierarchical clustering allowed flexibility to experiment with different distance metrics (e.g., Manhattan, cosine) to better fit the data characteristics. These preprocessing steps, algorithm choices, and modifications helped ensure robust and interpretable clustering results, aligning with the objectives of uncovering meaningful patterns and relationships in the dataset.

Results

Optimal Number of Clusters

The Elbow Method plot fig 2 shows a significant decrease in Within-Cluster Sum of Squares (WCSS) as the number of clusters increases. However, after 3 clusters, the rate of decrease slows, indicating diminishing returns. This “elbow” suggests that 3 clusters might be a good choice, balancing simplicity and explained variance.

The Silhouette Score plot fig 3 also supports this choice. The highest silhouette score occurs at 2 clusters, but it remains relatively high at 3 clusters before dropping sharply at 4 clusters. Thus, based on both WCSS and silhouette scores, 3 clusters is a suitable option for this dataset.

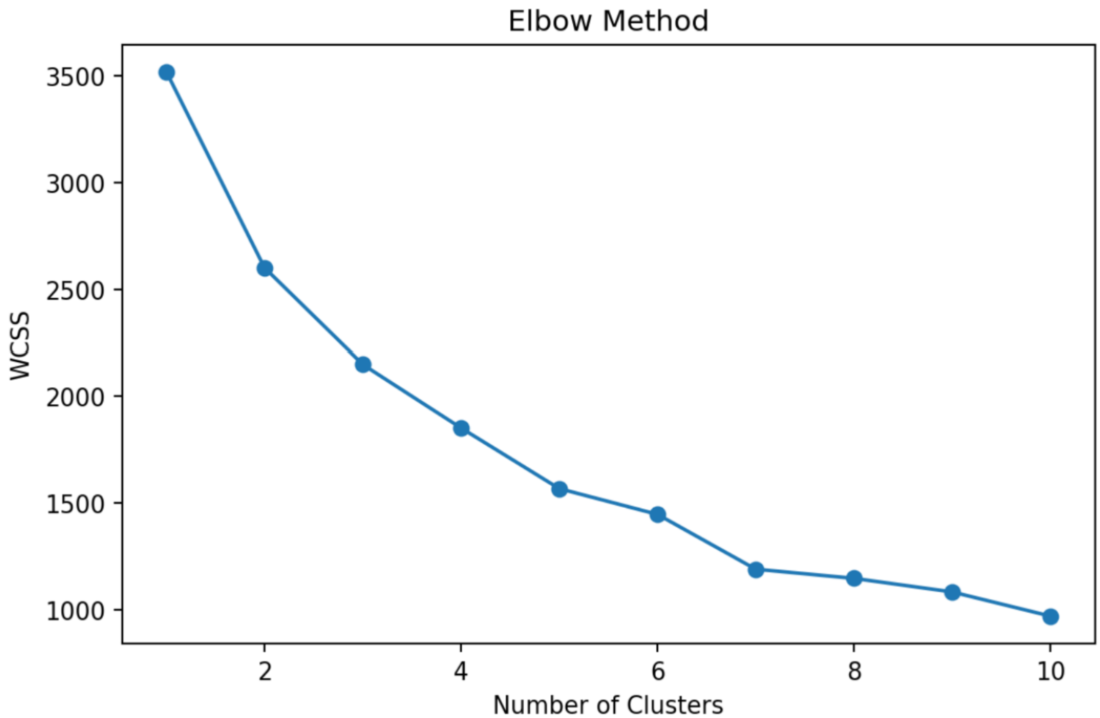


Figure 2: Fig 2: The Elbow Method plot

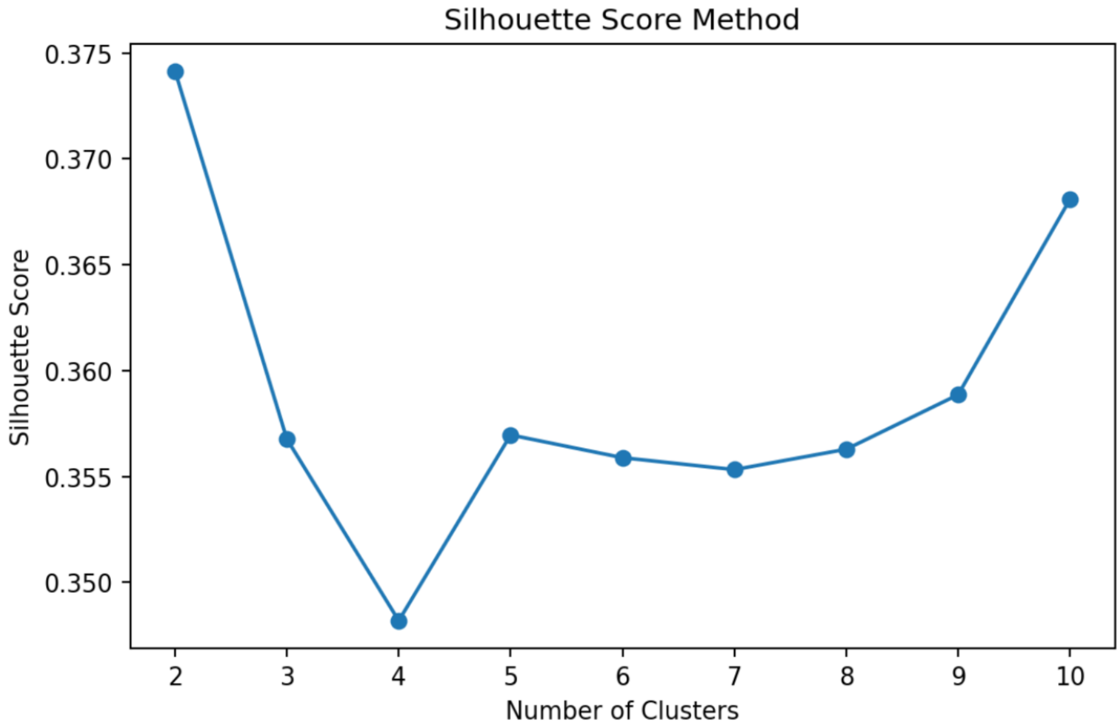


Figure 3: Fig 3: The Silhouette Score plot

Clustering Performance Metrics

To evaluate the clustering quality of each algorithm, we used metrics such as Silhouette Score and Within-Cluster Sum of Squares (WCSS) for K-means. The results are as follows:

K-means Clustering (3 clusters):

Silhouette Score: 0.357 – This moderate score suggests that the clusters are somewhat distinct, but there is room for improvement in separation.

WCSS: 2149.284 – The Elbow Method was used to determine that 3 clusters provide a reasonable trade-off between cluster compactness and model simplicity.

Hierarchical Clustering (3 clusters):

Silhouette Score: 0.360 – Similar to K-means, indicating comparable clustering quality. The hierarchical structure might capture certain nuances better, especially if there is an inherent hierarchy in the data.

DBSCAN:

Silhouette Score (excluding noise): 0.394 – The highest silhouette score among the methods. DBSCAN identifies clusters of varying shapes and densities, as well as noise points. The presence of noise points (labeled as -1) is beneficial for this dataset if it contains significant outliers or clusters of varying densities.

Visual Comparison

Using PCA for dimensionality reduction, we visualized the clusters identified by each algorithm as shown in Fig4 and Fig5:

K-means Clustering: The clusters were fairly compact and separated, with some overlap.

Hierarchical Clustering: Showed similar clusters to K-means, but slightly more defined due to its hierarchical approach.

DBSCAN: Produced clusters that were less compact but adaptable to irregular shapes, with noise points (outliers) separated from the main clusters.

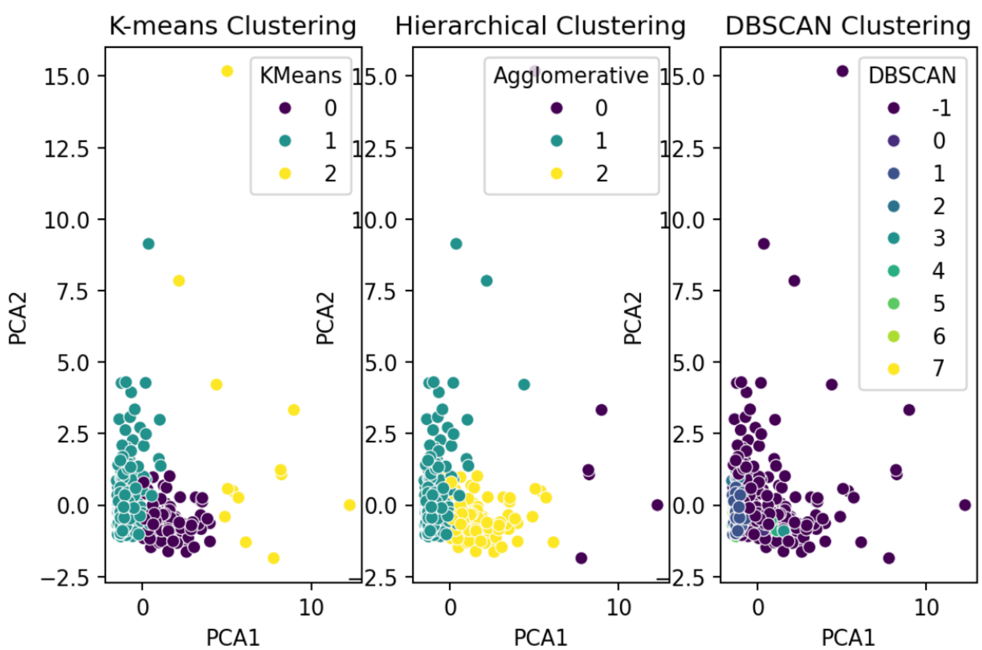


Figure 4: Fig 4: Visual Comparison of the clusters

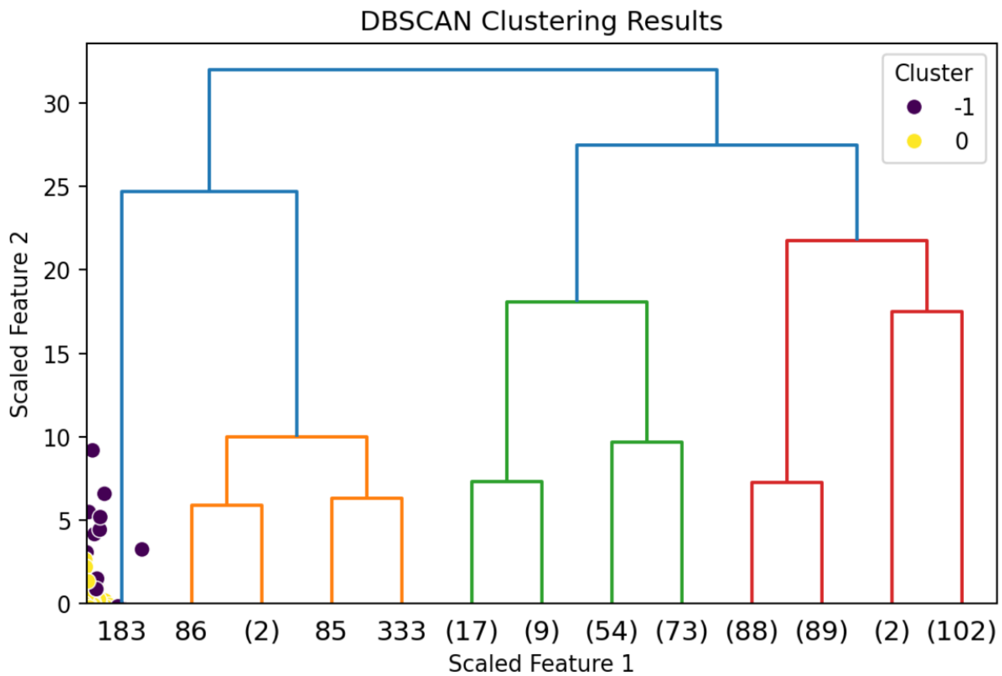


Figure 5: Fig 5: DBSCAN clustering

Discussion

Comparison of Clustering Algorithms

K-means Clustering:

Strengths: Simple to implement and computationally efficient. It works well with well-separated, spherical clusters and provides consistent results when given a specific number of clusters.

Weaknesses: K-means is sensitive to the initial selection of centroids, cluster shape (assuming spherical clusters), and the presence of outliers. In this case, the moderate silhouette score (0.357) indicates some overlap between clusters, potentially due to the complex structure of customer purchasing patterns.

Insights: The clusters formed by K-means likely represent groups of customers with similar spending habits across major categories. However, due to the algorithm's assumptions, K-means may not capture irregularly shaped clusters or variations in density effectively.

Hierarchical Clustering:

Strengths: Hierarchical clustering is capable of revealing a hierarchical structure, which can be beneficial if there is a natural hierarchy in the data. It does not require a predefined number of clusters, allowing for more flexibility in determining an appropriate number.

Weaknesses: Hierarchical clustering can be computationally expensive for large datasets and may struggle with high-density or noisy data.

Insights: Similar to K-means, hierarchical clustering produced moderately distinct clusters. However, the dendrogram visualization helps reveal potential sub-clusters within larger clusters, which could offer deeper segmentation insights if needed. This may highlight sub-groups with subtle variations in spending behavior.

DBSCAN:

Strengths: DBSCAN is effective at identifying clusters of varying shapes and densities, as well as separating out noise points (outliers). It does not require a predefined number of clusters, making it suitable for complex datasets with varied densities.

Weaknesses: DBSCAN is sensitive to parameter selection (eps and min_samples), which can significantly impact the results. If these parameters are not chosen well, DBSCAN may either overestimate noise or merge separate clusters.

Insights: DBSCAN achieved the highest silhouette score (0.394), indicating better-defined clusters than K-means and hierarchical clustering. This algorithm identified outliers as noise, which could represent customers with unique or infrequent spending patterns. The flexibility in cluster shape suggests DBSCAN might capture irregular groupings of customers that traditional algorithms could miss.

Customer Segmentation Insights

Based on the clustering analysis, we can provide insights into customer segmentation:

Cluster 1 (Moderate spenders): Found across K-means and hierarchical clustering, this group likely includes average customers with balanced spending across multiple categories. Marketing strategies for this group could include general promotions and loyalty programs to maintain engagement.

Cluster 2 (High-value customers): Present in all algorithms, these customers may show high spending in specific categories, such as “Fresh” or “Frozen” items. Targeted promotions for premium products or volume discounts could be effective for this segment.

Cluster 3 (Low-frequency or niche buyers): Especially visible in DBSCAN, this group may contain outliers or low-frequency buyers with irregular spending habits. Personalized outreach or re-engagement strategies could be effective for this group. Noise Points (Outliers): Identified by DBSCAN, these points represent unusual spending patterns that do not fit well into any main segment. Further analysis may reveal unique needs or occasional buyers, who could benefit from personalized offers or limited-time discounts.

Scenarios Where Cluster Analysis May Not Be Suitable or Effective

Data with No Natural Grouping Structure

Example: Financial data for stable, homogeneous customer groups may not contain significant variation to form meaningful clusters.

Reason: If the data lacks natural groupings, clustering will produce arbitrary segments without useful interpretation, leading to over-segmentation with little real-world relevance.

Highly Dynamic or Time-Sensitive Data

Example: Stock market data or real-time traffic data, where conditions change rapidly and clusters may shift frequently.

Reason: In dynamic environments, clusters formed today may no longer be relevant tomorrow. Frequent retraining may not be feasible, and clusters may not capture meaningful patterns over time.

Data Where Distance Metrics Are Not Meaningful

Example: Text data or data with categorical features that cannot be easily converted to a meaningful numeric scale.

Reason: Clustering often relies on distance metrics (e.g., Euclidean distance), which may not capture similarity well for non-numeric data, leading to meaningless clusters. Specialized clustering techniques (e.g., topic modeling for text) are often more appropriate.

Applications Requiring Labelled Groups

Example: Predictive modeling in healthcare, where specific diagnoses or categories are needed to guide treatment.

Reason: Clustering is an unsupervised method and does not provide labeled clusters. When labels are necessary, supervised learning methods or expert labeling may be more effective.

Highly Imbalanced Data

Example: Fraud detection, where fraudulent cases are rare compared to normal transactions.

Reason: In imbalanced datasets, clustering often results in clusters dominated by the majority class, while rare events are either ignored or grouped into broad clusters. Outlier detection methods or anomaly detection algorithms are better suited for such cases.

Summary of Findings

Best Performing Algorithm: DBSCAN achieved the highest silhouette score and identified distinct customer segments while handling outliers as noise, making it well-suited for this dataset.

Strengths of K-means and Hierarchical Clustering: Both algorithms performed reasonably well, especially in segmenting customers with regular, moderate spending patterns. However, their reliance on spherical clusters limits their ability to capture complex shapes.

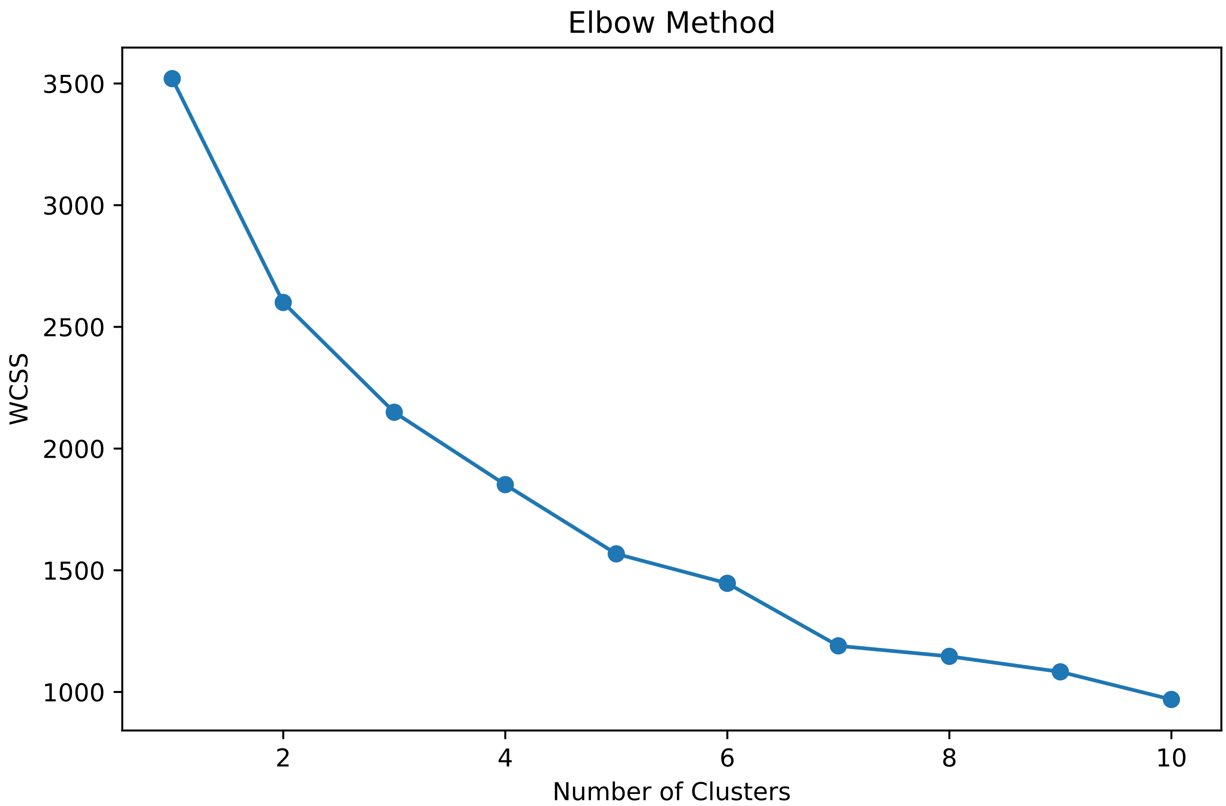
Future Work: For even more nuanced segmentation, combining clustering results (e.g., ensemble methods) or using additional features could improve the granularity of clusters and reveal hidden patterns.

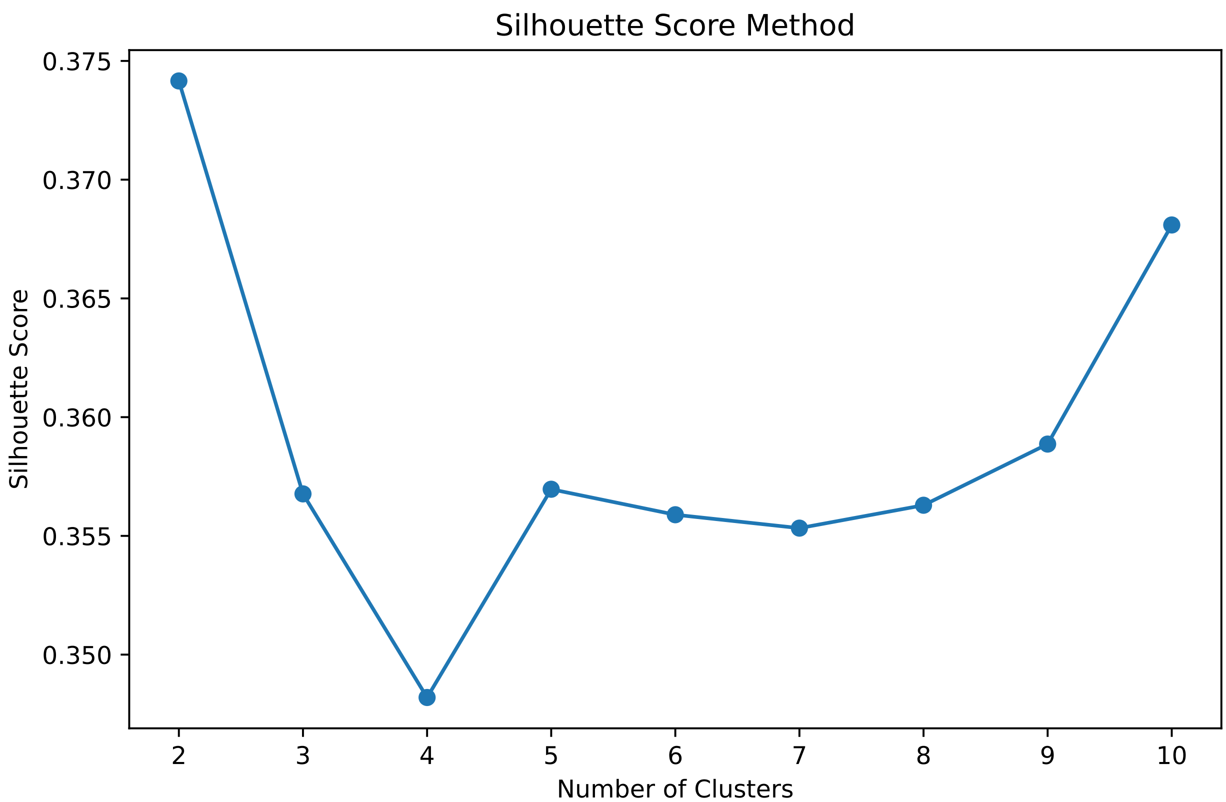
Appendix

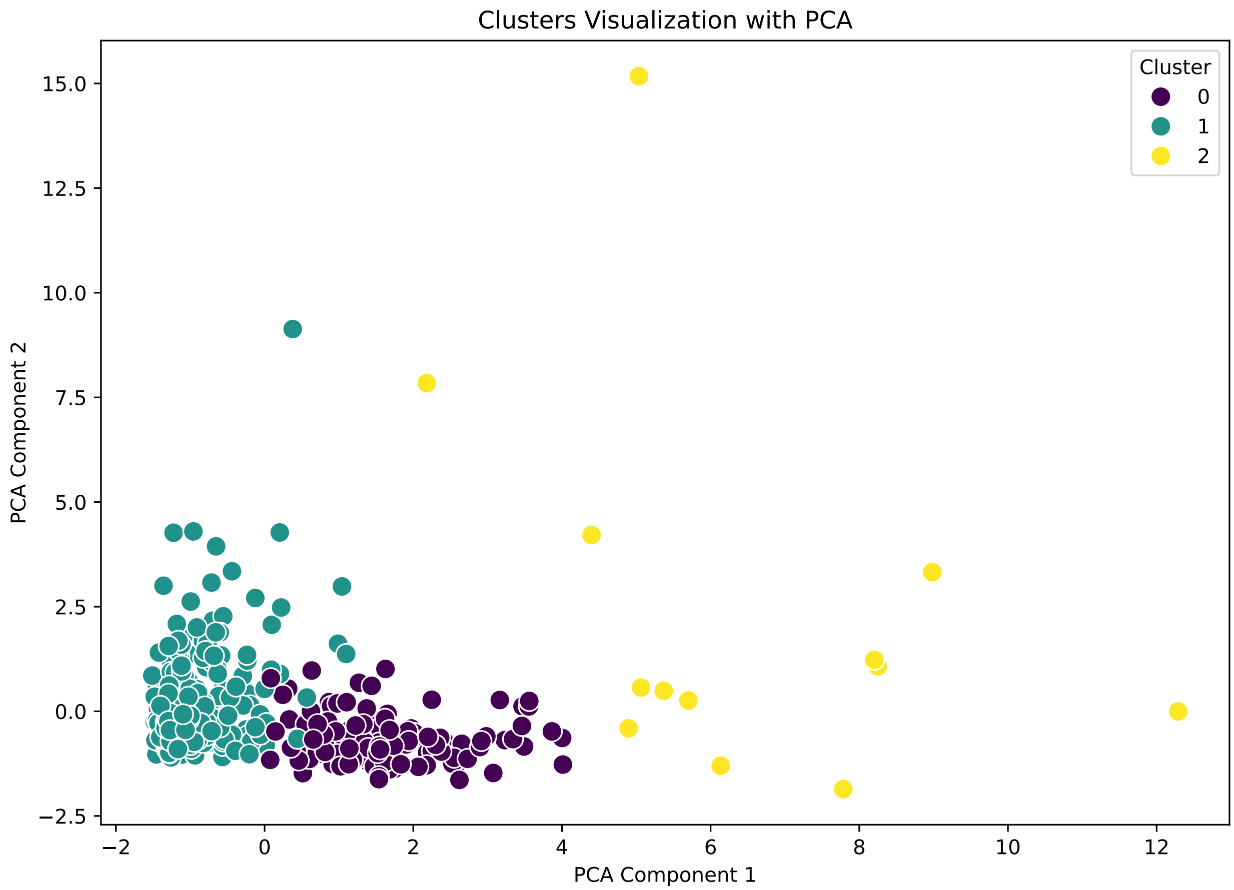
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
# Load your dataset (replace 'your_data.csv' with the actual filename)
df = pd.read_csv("Wholesale customers data.csv")

# Standardize the dataset if not already done
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(scaled_data)
    wcss.append(kmeans.inertia_)
```

```python
# Plot the Elbow Method results
plt.figure(figsize=(8, 5))
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
silhouette_scores = []
for i in range(2, 11):  # Silhouette score requires at least 2 clusters
    kmeans = KMeans(n_clusters=i, random_state=42)
    labels = kmeans.fit_predict(scaled_data)
    score = silhouette_score(scaled_data, labels)
    silhouette_scores.append(score)

# Plot the silhouette scores
plt.figure(figsize=(8, 5))
plt.plot(range(2, 11), silhouette_scores, marker='o')
plt.title('Silhouette Score Method')
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
plt.show()
# Set the optimal number of clusters (e.g., 3 here)
optimal_clusters = 3

# Fit K-means clustering
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42)
df['Cluster'] = kmeans.fit_predict(scaled_data)
from sklearn.decomposition import PCA

# Reduce dimensions for visualization
pca = PCA(n_components=2)
pca_data = pca.fit_transform(scaled_data)

# Add PCA results to the DataFrame for plotting
df['PCA1'] = pca_data[:, 0]
df['PCA2'] = pca_data[:, 1]

# Plot the clusters
plt.figure(figsize=(10, 7))
sns.scatterplot(data=df, x='PCA1', y='PCA2', hue='Cluster', palette='viridis', s=100)
plt.title('Clusters Visualization with PCA')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.legend(title='Cluster')
```

```
plt.show()
# Silhouette Score
silhouette_avg = silhouette_score(scaled_data, df['Cluster'])
print(f"Silhouette Score for {optimal_clusters} clusters: {silhouette_avg:.3f}")

# Within-Cluster Sum of Squares (WCSS)
print(f"Within-Cluster Sum of Squares (WCSS) for {optimal_clusters} clusters: {kmeans.in
```




Silhouette Score Method
Silhouette Score
0.375
0.370
0.365
0.360
0.355
0.350
2
3
4
5
6
7
8
9
10
Number of Clusters



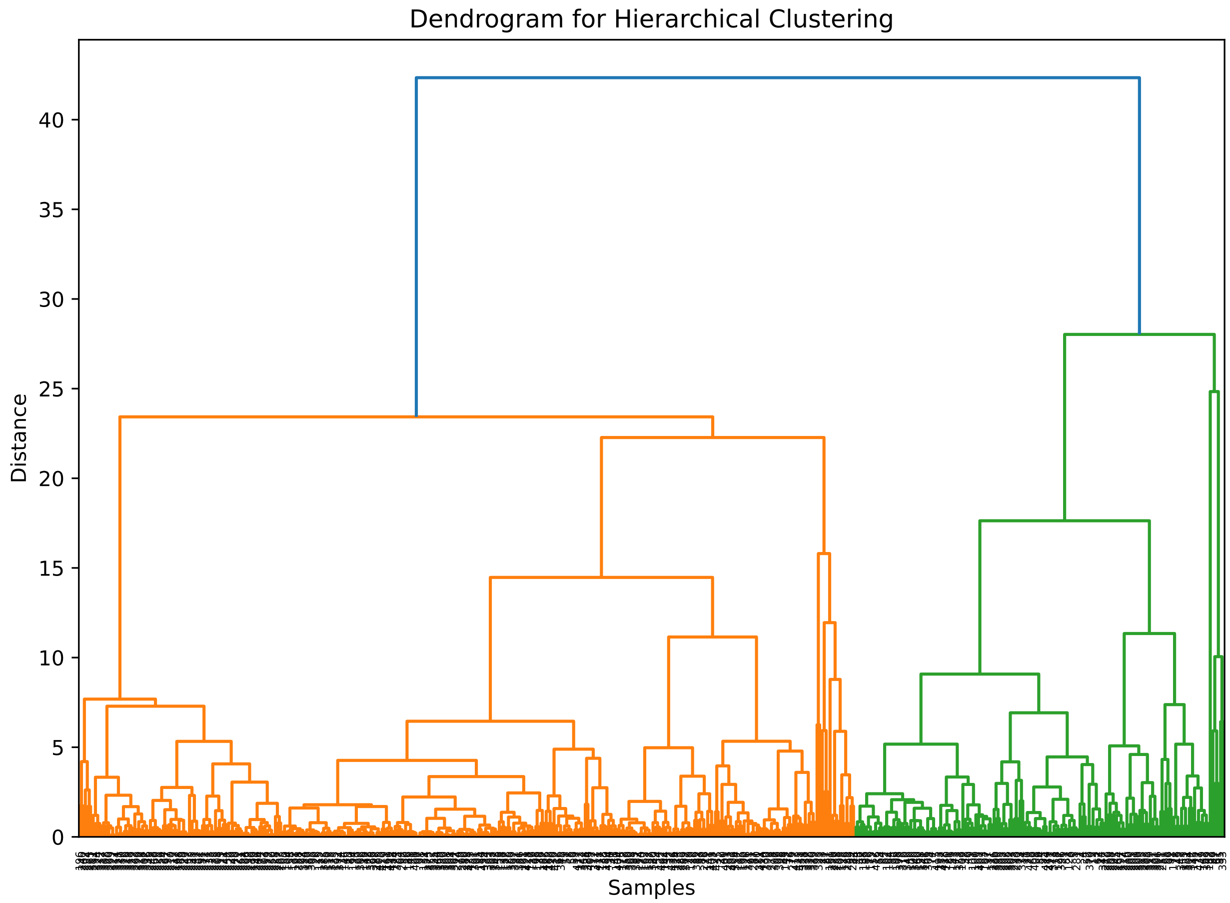

```
Silhouette Score for 3 clusters: 0.357
Within-Cluster Sum of Squares (WCSS) for 3 clusters: 2149.284
```

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.decomposition import PCA
# Load your dataset (replace 'your_data.csv' with the actual filename)
df = pd.read_csv("Wholesale customers data.csv")

# Standardize the dataset if not already done
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)
# Optimal clusters determined previously (e.g., 3 clusters)
kmeans = KMeans(n_clusters=3, random_state=42)
```

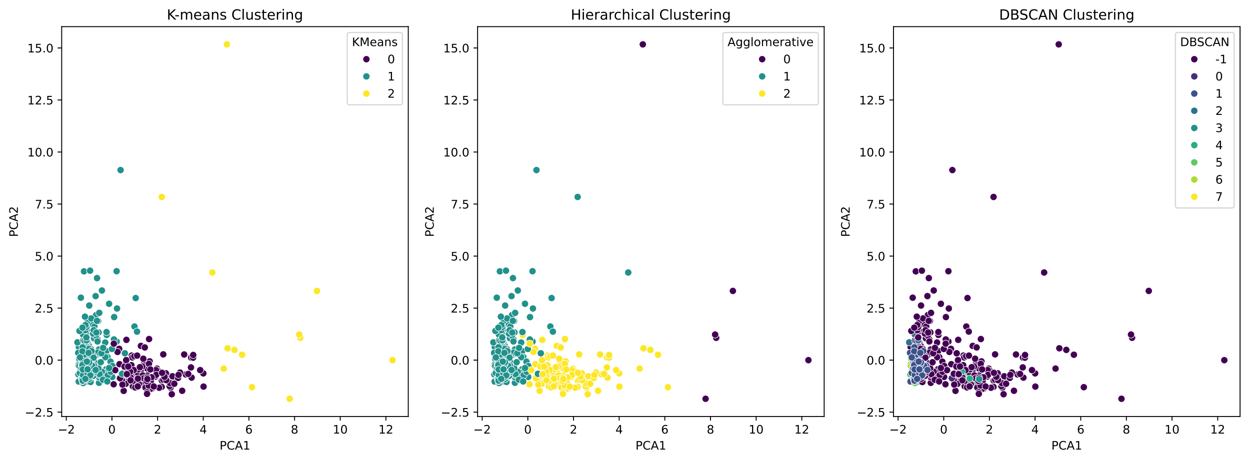
```
kmeans_labels = kmeans.fit_predict(scaled_data)

# Evaluate K-means
kmeans_silhouette = silhouette_score(scaled_data, kmeans_labels)
print(f"K-means Silhouette Score: {kmeans_silhouette:.3f}")
# Generate a linkage matrix for the dendrogram
linked = linkage(scaled_data, method='ward')

# Plot dendrogram
plt.figure(figsize=(10, 7))
dendrogram(linked)
plt.title("Dendrogram for Hierarchical Clustering")
plt.xlabel("Samples")
plt.ylabel("Distance")
plt.show()
# Apply hierarchical clustering with the chosen number of clusters
agglo = AgglomerativeClustering(n_clusters=3)
agglo_labels = agglo.fit_predict(scaled_data)

# Evaluate hierarchical clustering
agglo_silhouette = silhouette_score(scaled_data, agglo_labels)
print(f"Hierarchical Clustering Silhouette Score: {agglo_silhouette:.3f}")
# Apply DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(scaled_data)

# Filter out noise points (-1 labels) for silhouette score calculation
dbscan_silhouette = silhouette_score(scaled_data[dbscan_labels != -1], dbscan_labels[dbs
print(f"DBSCAN Silhouette Score (excluding noise): {dbscan_silhouette:.3f}")
# Reduce to 2D using PCA for visualization
pca = PCA(n_components=2)
pca_data = pca.fit_transform(scaled_data)

# Add clustering labels from each algorithm to a DataFrame
df_pca = pd.DataFrame(pca_data, columns=['PCA1', 'PCA2'])
df_pca['KMeans'] = kmeans_labels
df_pca['Agglomerative'] = agglo_labels
df_pca['DBSCAN'] = dbscan_labels

# Plotting each clustering result
fig, axs = plt.subplots(1, 3, figsize=(18, 6))
sns.scatterplot(data=df_pca, x='PCA1', y='PCA2', hue='KMeans', palette='viridis', ax=axs
axs[0].set_title("K-means Clustering")
sns.scatterplot(data=df_pca, x='PCA1', y='PCA2', hue='Agglomerative', palette='viridis',
```

```
axs[1].set_title("Hierarchical Clustering")
sns.scatterplot(data=df_pca, x='PCA1', y='PCA2', hue='DBSCAN', palette='viridis', ax=axs
axs[2].set_title("DBSCAN Clustering")
plt.show()
```

```
K-means Silhouette Score: 0.357
```




```
Hierarchical Clustering Silhouette Score: 0.360
DBSCAN Silhouette Score (excluding noise): 0.394
```



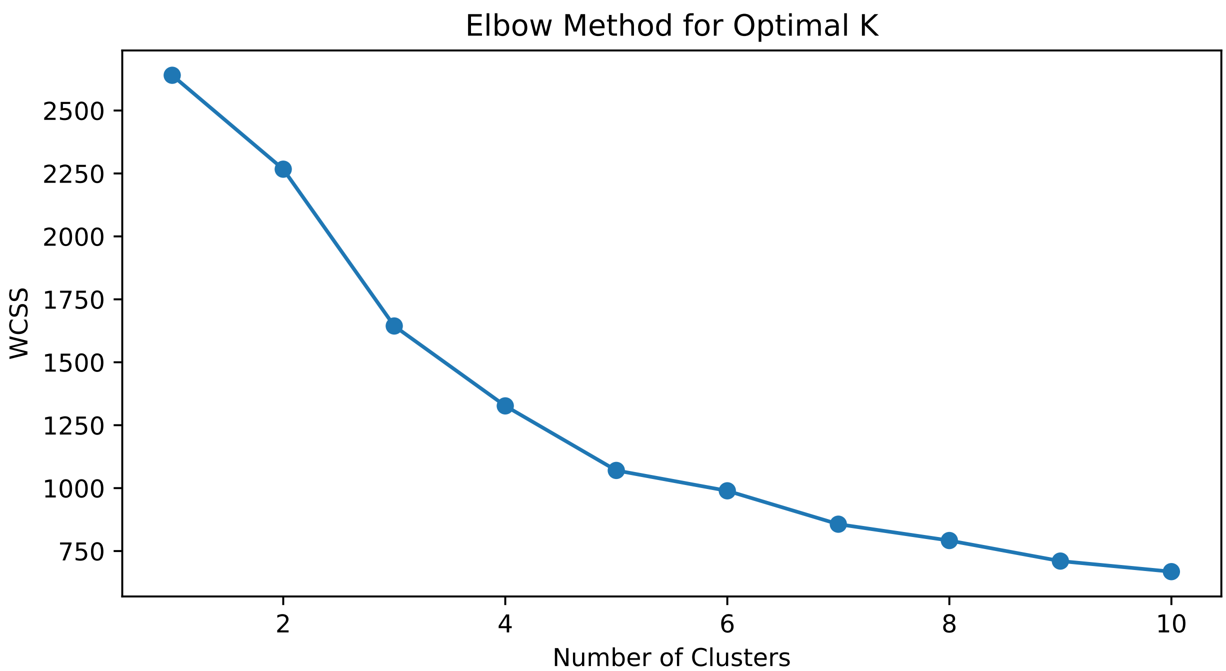

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
from scipy.cluster.hierarchy import dendrogram, linkage

# Step 1: Load the Dataset
df = pd.read_csv("Wholesale customers data.csv")

# Display the first few rows of the dataset to understand its structure
print("First few rows of the dataset:")
print(df.head())

# Generate descriptive statistics for the dataset
print("\nDescriptive Statistics:")
print(df.describe())

# Check for any missing values
print("\nMissing Values:")
print(df.isnull().sum())

# Step 2: Data Preprocessing
# Check for missing values
print("Missing values:\n", df.isnull().sum())
# Normalize the data for clustering
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df.iloc[:, 2:])  # Exclude Channel and Region columns i
```

```
# Step 3: Apply K-means Clustering
# Determine the optimal number of clusters using the Elbow method
wcss = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(df_scaled)
    wcss.append(kmeans.inertia_)

# Plot the Elbow curve
plt.figure(figsize=(8, 4))
plt.plot(range(1, 11), wcss, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.title('Elbow Method for Optimal K')
plt.show()

# Fit KMeans with the chosen number of clusters (e.g., 3 based on the Elbow curve)
optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
kmeans_labels = kmeans.fit_predict(df_scaled)

# Step 4: Visualize Clustering Results
df['Cluster'] = kmeans_labels
sns.pairplot(df, hue='Cluster', palette='viridis', markers=["o", "s", "D"])
plt.suptitle("K-means Clustering Results", y=1.02)
plt.show()

# Step 5: Evaluate Clustering Quality
silhouette_avg = silhouette_score(df_scaled, kmeans_labels)
print(f"Silhouette Score for K-means with {optimal_k} clusters: {silhouette_avg}")

# Step 6: Interpret and Analyze the Clusters
# Describe characteristics of each cluster
print("Cluster analysis:")
for cluster in range(optimal_k):
    print(f"\nCluster {cluster}")
    print(df[df['Cluster'] == cluster].describe())

# Step 7: Experiment with Other Clustering Algorithms

# Hierarchical Clustering (Agglomerative)
agg_cluster = AgglomerativeClustering(n_clusters=optimal_k)
agg_labels = agg_cluster.fit_predict(df_scaled)
df['Agg_Cluster'] = agg_labels
```

```
# Plot Dendrogram for hierarchical clustering
linked = linkage(df_scaled, 'ward')
plt.figure(figsize=(10, 5))
dendrogram(linked, truncate_mode='level', p=3)
plt.title("Hierarchical Clustering Dendrogram")
plt.show()

# DBSCAN Clustering
dbscan = DBSCAN(eps=1.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(df_scaled)
df['DBSCAN_Cluster'] = dbscan_labels

# Visualize DBSCAN results
sns.scatterplot(x=df_scaled[:, 0], y=df_scaled[:, 1], hue=dbscan_labels, palette="viridi
plt.xlabel("Scaled Feature 1")
plt.ylabel("Scaled Feature 2")
plt.title("DBSCAN Clustering Results")
plt.legend(title='Cluster')
plt.show()

# Step 8: Reflect on Limitations and Challenges
# Print a summary for reflection in the report
print("\n--- Summary of Clustering Results ---")
print("K-means Silhouette Score:", silhouette_avg)
print("Other clustering algorithms like hierarchical and DBSCAN provide additional persp
```

```
First few rows of the dataset:
   Channel  Region  Fresh  Milk  Grocery  Frozen  Detergents_Paper  Delicassen
0        2       3  12669  9656     7561     214              2674        1338
1        2       3   7057  9810     9568    1762              3293        1776
2        2       3   6353  8808     7684    2405              3516        7844
3        1       3  13265  1196     4221    6404               507        1788
4        2       3  22615  5410     7198    3915              1777        5185

Descriptive Statistics:
          Channel      Region          Fresh          Milk       Grocery  \
count  440.000000  440.000000     440.000000    440.000000    440.000000
mean     1.322727    2.543182   12000.297727   5796.265909   7951.277273
std      0.468052    0.774272   12647.328865   7380.377175   9503.162829
min      1.000000    1.000000       3.000000     55.000000      3.000000
25%      1.000000    2.000000    3127.750000   1533.000000   2153.000000
50%      1.000000    3.000000    8504.000000   3627.000000   4755.500000
75%      2.000000    3.000000   16933.750000   7190.250000  10655.750000
max      2.000000    3.000000  112151.000000  73498.000000  92780.000000
```

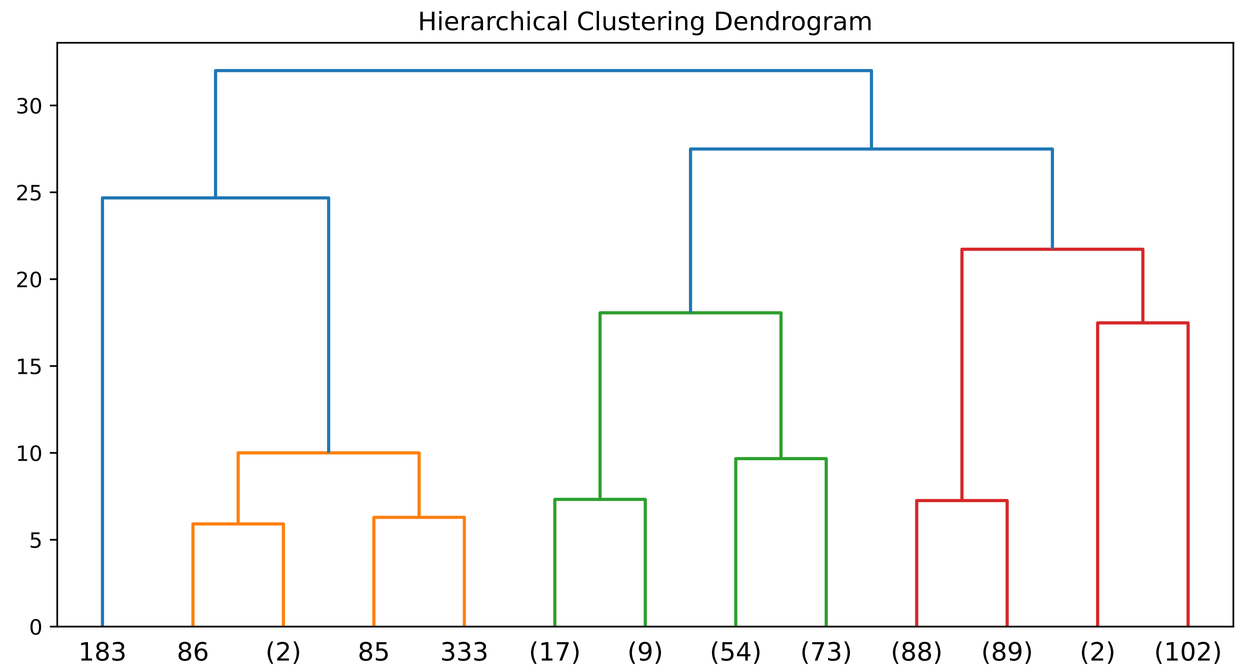
```
             Frozen  Detergents_Paper    Delicassen
count    440.000000        440.000000    440.000000
mean    3071.931818       2881.493182   1524.870455
std     4854.673333       4767.854448   2820.105937
min       25.000000          3.000000      3.000000
25%      742.250000        256.750000    408.250000
50%     1526.000000        816.500000    965.500000
75%     3554.250000       3922.000000   1820.250000
max    60869.000000      40827.000000  47943.000000

Missing Values:
Channel             0
Region              0
Fresh               0
Milk                0
Grocery             0
Frozen              0
Detergents_Paper    0
Delicassen          0
dtype: int64
Missing values:
 Channel             0
Region              0
Fresh               0
Milk                0
Grocery             0
Frozen              0
Detergents_Paper    0
Delicassen          0
dtype: int64
```

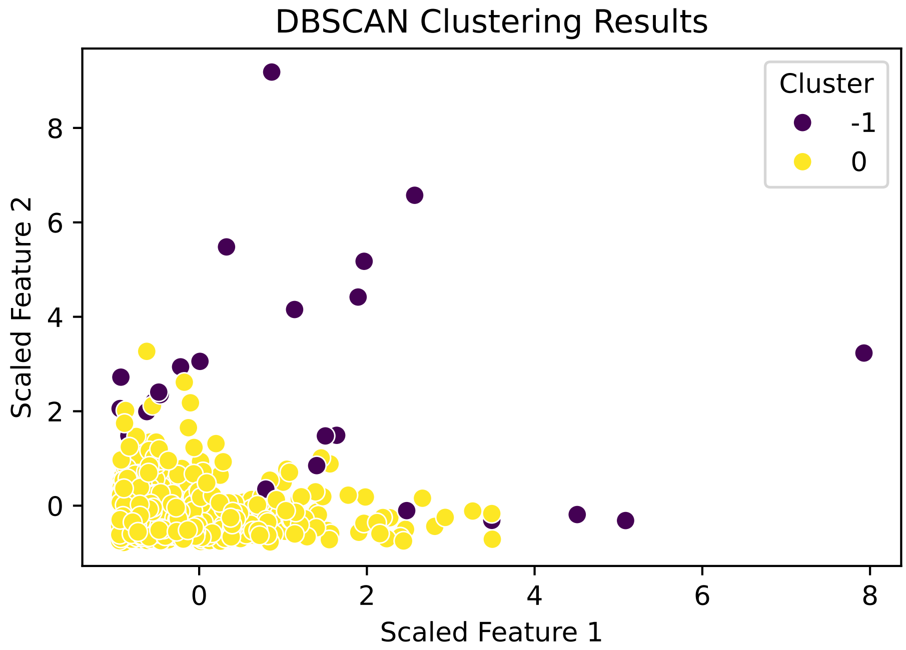

Elbow Method for Optimal K
WCSS
2500
2250
2000
1750
1500
1250
1000
750
2
4
6
8
10
Number of Clusters


```
Silhouette Score for K-means with 3 clusters: 0.4582633767207058
Cluster analysis:

Cluster 0
          Channel      Region        Fresh         Milk      Grocery  \
count  350.000000  350.000000   350.000000   350.000000   350.000000
mean     1.282857    2.534286  8935.500000  4228.528571  5848.034286
std      0.451032    0.781389  7273.779311  3745.104358  5056.662882
min      1.000000    1.000000     3.000000    55.000000     3.000000
25%      1.000000    2.000000  2867.250000  1367.250000  2023.500000
50%      1.000000    3.000000  7326.500000  3141.500000  3830.500000
```

```
75%      2.000000    3.000000  13235.250000   6244.500000   8842.500000
max      2.000000    3.000000  31812.000000  23527.000000  28986.000000

             Frozen  Detergents_Paper   Delicassen  Cluster
count    350.000000        350.000000   350.000000    350.0
mean    2167.231429       1913.605714  1102.120000      0.0
std     2315.944430       2341.879697  1051.689045      0.0
min       25.000000          3.000000     3.000000      0.0
25%      634.250000        240.250000   373.500000      0.0
50%     1292.000000        708.000000   777.000000      0.0
75%     2796.250000       3363.750000  1520.500000      0.0
max    15601.000000      10069.000000  7844.000000      0.0

Cluster 1
         Channel     Region          Fresh          Milk       Grocery  \
count  53.000000  53.000000      53.000000     53.000000     53.000000
mean    1.113208   2.698113   34540.113208   5860.358491   6122.622642
std     0.319878   0.695726   18329.740161   7332.281154   4942.528636
min     1.000000   1.000000    4983.000000    286.000000    471.000000
25%     1.000000   3.000000   23632.000000   2408.000000   2593.000000
50%     1.000000   3.000000   31614.000000   3944.000000   4955.000000
75%     1.000000   3.000000   42312.000000   5506.000000   7336.000000
max     2.000000   3.000000  112151.000000  43950.000000  21042.000000

             Frozen  Detergents_Paper    Delicassen  Cluster
count     53.000000         53.000000     53.000000     53.0
mean    9841.735849        981.471698   3664.245283      1.0
std    10371.448091       1146.313583   6858.590495      0.0
min      287.000000         20.000000      3.000000      1.0
25%     3242.000000        246.000000   1163.000000      1.0
50%     7368.000000        603.000000   2235.000000      1.0
75%    13135.000000       1145.000000   2931.000000      1.0
max    60869.000000       4948.000000  47943.000000      1.0

Cluster 2
       Channel     Region         Fresh          Milk       Grocery  \
count     37.0  37.000000     37.000000     37.000000     37.000000
mean       2.0   2.405405   8704.864865  20534.405405  30466.243243
std        0.0   0.797895  10095.084635  14263.733391  15785.274957
min        2.0   1.000000     85.000000   3737.000000  13567.000000
25%        2.0   2.000000   2137.000000  12697.000000  21570.000000
50%        2.0   3.000000   5283.000000  15488.000000  25957.000000
75%        2.0   3.000000  11223.000000  25071.000000  32114.000000
max        2.0   3.000000  44466.000000  73498.000000  92780.000000
```

	Frozen	Detergents_Paper	Delicassen	Cluster
count	37.000000	37.000000	37.000000	37.0
mean	1932.621622	14758.837838	2459.351351	2.0
std	1805.946249	7920.259983	2989.900093	0.0
min	36.000000	4337.000000	37.000000	2.0
25%	864.000000	9529.000000	797.000000	2.0
50%	1456.000000	12591.000000	1452.000000	2.0
75%	2367.000000	17740.000000	2944.000000	2.0
max	7782.000000	40827.000000	16523.000000	2.0






```
--- Summary of Clustering Results ---
K-means Silhouette Score: 0.4582633767207058
Other clustering algorithms like hierarchical and DBSCAN provide additional perspectives
```