Data Mining – Assignment 2

Prem Mettupalli – 0826227

1. Introduction:

1.1 Clustering:

- In the previous assignment we have implemented machine learning classification models like decision trees and support vector machines. These classification models are called supervised learning models cause the data that we have used in developing these models is labelled.
- Unlike decision trees and SVMs, clustering models are unsupervised learning algorithms cause the data used with these models is not labelled, means the data is not segregated into different classes.
- Clustering analysis is an unsupervised technique in data mining which is performed to organize a set of data points into groups or clusters based on the similarities. The points from the same cluster are more similar to each other compared to the points from a different cluster.

1.2 Significance of clustering:

- Clustering helps us to identify and group similar patterns of data which allows businesses to understand the customers more clearly. This will help in identifying groups with similar interests or preferences such that businesses can focus on them.
- Business can target customers of same tastes or purchasing patterns with better marketing strategies.

1.3 Steps involved in clustering analysis:

- **Model selection:** Selecting an appropriate clustering model is very important and depends upon several factors such as the characteristics of the dataset like size, cluster shape, and density etc.
- **Data preparation:** After selecting an appropriate model for our use case we need to prepare our data for clustering. Data preparation involves cleaning the data, filling missing values, normalising etc. Clustering algorithms like K-means are sensitive to the scale of measurement and outliers. If the features are measured on different scales, they should be scaled to a similar range using StandardScalar.
- **Clustering:** In this step we will apply the selected clustering model on the processed dataset to group the data points into clusters based on the similarities. Different algorithms perform clustering in a different way, and each has its own pros and cons.
- **Evaluation:** After the clustering is done, we access the performance and efficiency of the algorithm is assessed by using metrics like Silhouette Score, Within-Cluster Sum of Squares (WCSS), Dendrograms and cluster size distribution.

1.4 Different types of clustering algorithms implemented:

1. **K – means Clustering:**
    - K-means clustering is a widely used partitioning type clustering algorithm. In this algorithm the data points are partitioned into a defined number (K) of clusters each cluster is represented by a centroid.
    - So first we will randomly select k initial centroids from the dataset. Instead of selecting these centroids randomly we can also use methds like k-means++ for better starting points.
    - Then in the next step we will assign each data point from the dataset to the nearest centroid based up on the distance between them.
    - After assigning the points to the initial centroids we will caluculate the new centroids by taking the mean of all the data points assigned to each cluster based on the initial centroids.

- Then we will reassign the points in the dataset to the new centroids.
- We will repeat thes setps till the centoids won't change or the datapoints stops changing from one cluster to another.
- Evaluation metrix such as Within-cluster sum of squares WCSS are used to assess the clustering algorithm performance. It measures the sum of squared distances between data points and their centroids, and we need to minimise it.

2. **Hierarchical Clustering:**
    - Hierarchical clustering produces a set of nested clusters organised as a hierarchical tree called dendrograms.
    - Dendrograms are a tree like structures that records the sequences of merges or splits.
    - There are 2 types of hierarchical clustering's they are:
        - Agglomerative: It is like a bottom-up approach. In this type of clustering we start with each data point as a separate cluster and will merges the closest cluster iteratively until we are left with only one cluster.
        - Divisive: it is like a Top-Down approach. In this clustering algorithm we start with one large cluster containing all the data points and will splits the clusters iteratively until each point becomes a separate cluster.
        - We use linkage methods like Max, Min, average and ward's method for determining the distance between clusters. This distance decides which clusters to be combined in each iteration.
3. **Density based Clustering (DBSCAN):**
    - Density based clustering algorithms group the data points based on density. These are very useful for identifying clusters of different shapes and sizes and also handles noise effectively.
    - In density-based clustering we will define parameters like Epsilon and MinPts for determining the radius or neighbourhood distance around a data point and the minimum number of points required to form a cluster or dense region
    - After defining the epsilon and MinPts we will identify the core, border and noise points to form dense regions
    - Core points are the once that are inside the defined radius epsilon and should have neighbours.
    - Border points are the points which don't have any neighbours but are within the distance of a core points.
    - Noise points are described as the points which are neither core nor border points.
    - Density clustering algorithms like DBSCAN starts with a point and if it is considered as acore point it forms a new cluster and then expands to include all the other core and border points.
    - The cluster continues to grow as long as the algorithm can find new core points.

2. Dataset description:

The data set given contains information about the annual spendings by customers across different categories of products sold by a wholesale distributor. Each row represents a customer and their spendings on different products like fresh produce, milk, groceries, frozen food, detergents and delicatessens and also has the information about the Channel and the region. we have 440 entries.

2.1 Attribute description:

Feature variables considered for clustering:

1. Fresh: Annual spending on fresh produce. (Continuous)
2. Milk: Annual spending on dairy based products. (Continuous)

3. Grocery: Annual spending on grocery items. (Continuous)
4. Frozen Foods: Annual spending on frozen products. (Continuous)
5. Detergents: Annual spending on cleaning products. (Continuous)
6. Delicatessen: Annual spending on delicatessen items. (Continuous)
7. Region: Customer region (Lisnon - 1, Oporto – 2 and Other - 3)
8. Channel: Customer channel. (Horeca – 1, Retail - 2)

3. Methodology:

Import the required libraries for implementing the clustering algorithm and data processing.

3.1 Data processing:

1. **Loading the data:** Load the data using the URL mentioned and store it in a panda's data frame. The dataset is maintained by UCI Machine Learning Repository.
2. **Finding missing values:** We don't need to check for any null values as it is mentioned in the UCI Machine Learning Repository source that our data doesn't have any missing values.
3. **Viewing and analysing the data:** Print the first five rows of the data frame using head so that you can view the data and the features available in the dataset for clustering.
4. **Considering the features available for clustering:** After the data set is printed, we can see that we have 8 columns, and we can use all these as features for clustering.
5. **Standardizing the data:** After deciding on the features, we need to standardize the data. Standardizing helps in keeping all the data on a single standard scale. Generally, datasets consist of multiple features and there is no guarantee that all these features are measured on the same scale, in order to prevent the skewing of the data points due to difference in scales, which affects the performance of the models like K-means clustering we need to perform standardizing. After standardizing the data, we can store the data in another variable and it will be stored as a NumPy array.

3.2 K-Means Clustering:

1. For using k-means clustering algorithm we fist need to decide on the number of clusters to be used for performing data partition.
2. We can do this by using Silhouette scores. Silhouette scores measure how similar each point is to its cluster compared to the other clusters.
3. First, we will take the number of clusters from the range of 2-11 and for each number of clusters we perform the K-means clustering and will append the associated silhouette score to an empty list.
4. The number of clusters for which we will get maximum Silhouette scores is considered for performing the K-Means clustering.
5. After deciding on the number of clusters to be used we will perform K-Means clustering and will get the related cluster labels for each data point stored into a variable.
6. We will then calculate the Silhouette scores, WCSS for K-Means clustering and will plot the clusters. Plotting the clusters is difficult due to the high dimensionality of the dataset as we have 8 features, and it is not possible to represent them in a 2-Dimenssional plane.
7. Hence, we will use Primary Component Analysis (PCA) to reduce the dimensionality, and we will then create a scatter plot to visualize the clusters.

3.3 Agglomerative Hierarchical Clustering:

1. In Agglomerative clustering we will initialize the model by choosing the number of clusters. In this case we will use the same number of clusters for both K-Means and agglomerative clustering. We will choose ward as the linkage criterion for this algorithm as it minimises the variance within the cluster.

2. After initializing the model, we will fit the model on the scaled data and the model will assign each data point to a cluster.
3. Then we will calculate the Silhouette score for the Agglomerative Clustering using the data scaled.
4. As Hierarchical Clustering doesn't have the concept of centroids there is no straight forward way to calculate the WCSS score, hence we will calculate the mean of all the points in the cluster and will treat it as a centroid and will then calculate the squared distance sum to the centroid. This will give us the WCSS score.
5. Plot the clusters created by the Agglomerative Clustering by following the same approach we used for plotting k-means clusters, that is use PCA for dimensionality reduction and plot the clusters in a 2-Dminetional plane.

3.4 DBSCAN Clustering:

1. While performing DBSCAN clustering we should first initiate the DBSCAN function with the epsilon and MinPts values.
2. Then we will fit that function to out scaled data for performing clustering of the data points. The clustering labels for the data points are stored in a variable.
3. All the points that are labelled as -1 are the considered as noise points and the cluster is defined as noise cluster.
4. After filtering out the noise points and cluster we will find the Silhouette Score for evaluating the DBSCAN performance. We should check for the Silhouette Score only when we have more than two clusters excluding the noise clusters.
5. Then we will plot the clusters using the PCA components for visualizing.

4. Results:

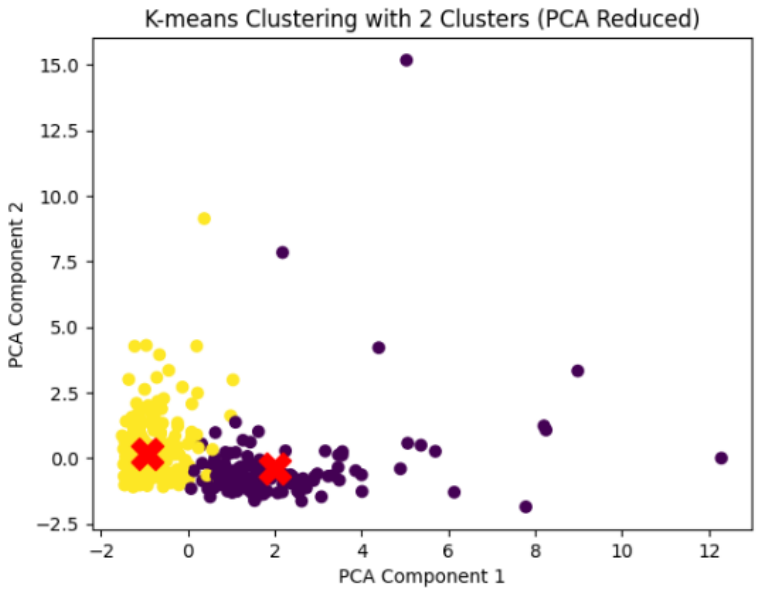
4.1 Silhouette scores and WCSS:

Clustering	Silhouette scores	WCSS
K-Means	0.3741	2600.38
Agglomerative Clustering	0.3681	2623.61
DBSCAN	0.4123	

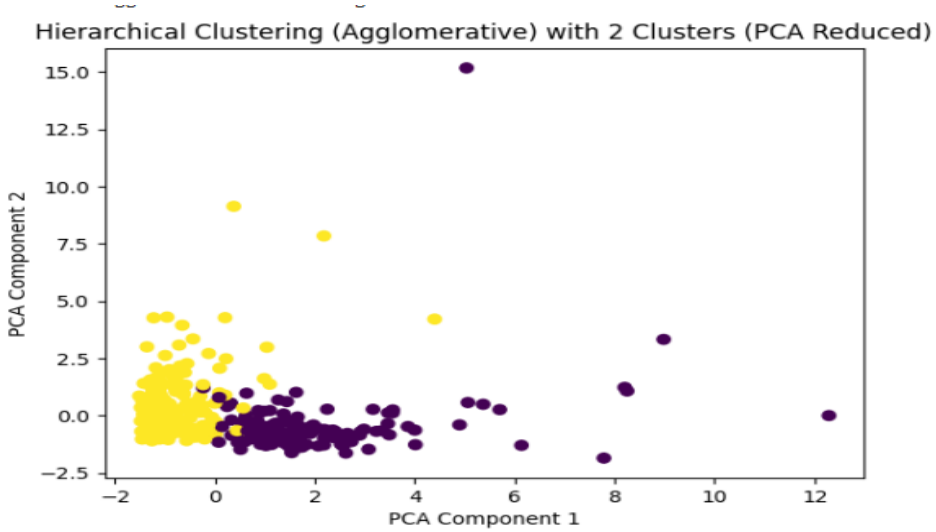
1. **Silhouette score** measures how similar a point is to its cluster compared to the other clusters. High silhouette score indicates better clustering. In our case K-Means have a slightly better silhouette score compared to the agglomerative clustering.
2. **WCSS** stands for with in cluster squared sum. It is the squared sum of the distances between the data points in a cluster and its centroid or mean point. Lower WCSS indicate more compact and better clustering. In our case K-Means clustering have lower WCSS value compared to agglomerative clustering.
3. From the results we can say that DBSCAN achieved the highest Silhouette Score of 0.4123 compared to the other clustering algorithms. This indicates that DBSCAN can perform better in capturing the natural structures of the data without demanding for any predefined shapes, unlike K-Means. We cannot compute WCSS for DBSCAN cause WCSS indicates how compact the clusters are and DBSCAN doesn't aim to form compact clusters. In terms of compactness K-Means performed better compared to Agglomerative clustering.

4.2 Scatter plots representing the clusters:

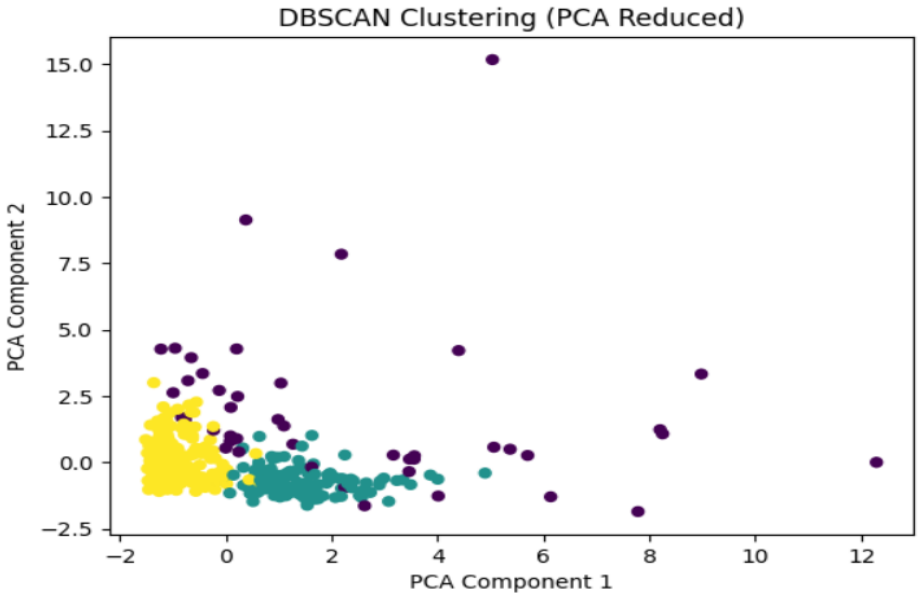
Plot 1: K-Means clustering:



Plot 2: Agglomerative clustering:



Plot 3: DBSCAN clustering:



4.3 Customer Spendings data with added cluster labels:

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

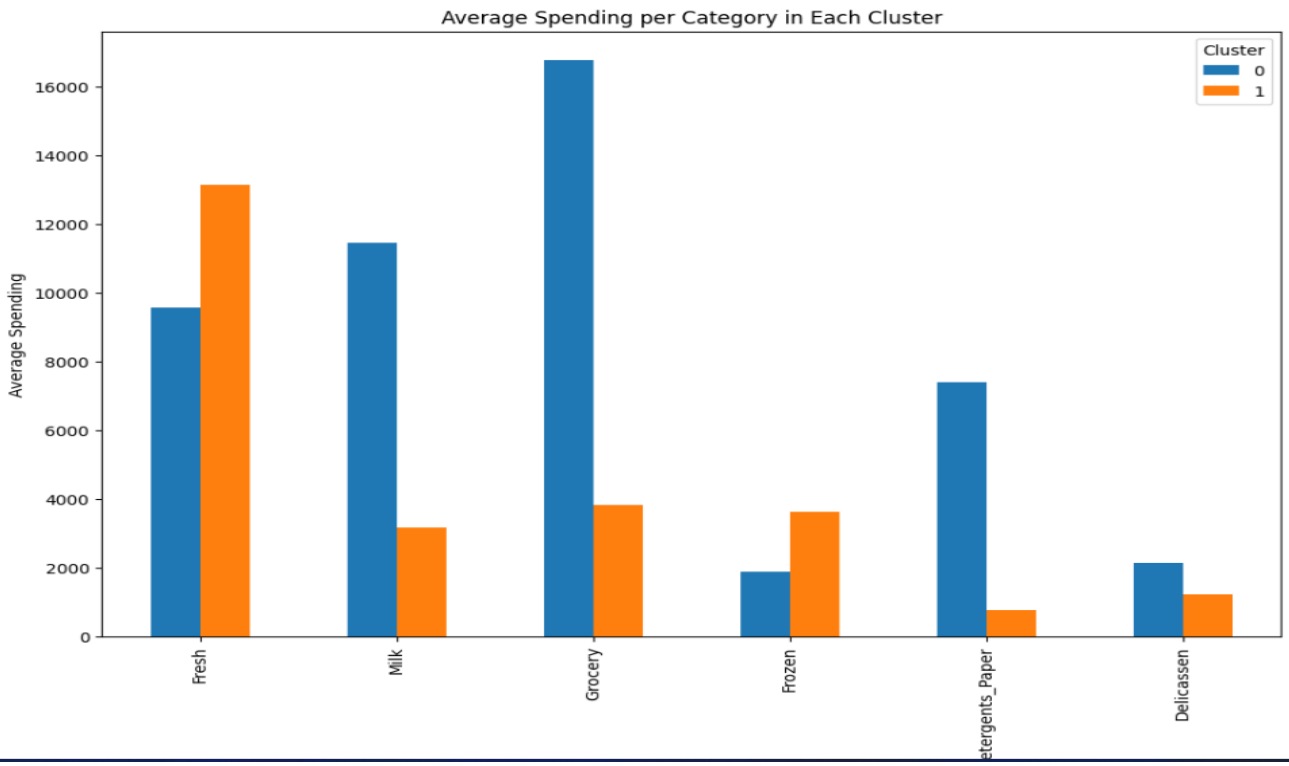
- I have appended the labels generated from K-Means algorithm to the dataset such that it will be easy to understand the clustering results.

4.4 Analysing the results of K-Means for identifying purchasing patterns:

1. Let us consider the K-Means clusters for analysing the purchasing patterns of the customers.
2. As we have used Primary component analysis for plotting the clusters by performing dimensionality reduction, we cannot directly interpret or understand how the individual features of a data point are influencing its placement in a specific cluster.
3. So instead of relying on the visualizations we can compare the mean values of each feature in a cluster to another clusters mean feature values.
4. This comparison will give us a basic idea on how and what basis the customers are segregated into clusters.
5. To perform this analysis first we will append the K-Means cluster labels of all the data points to the original dataset such that we can have each data point and their associated cluster label in one table.
6. Next, we calculate the mean of each feature across all data points within each cluster. This gives us the average spending for each category such as Fresh, Milk, Frozen, etc for each cluster (separately for cluster 1 and cluster 2).
7. Below table has the mean values of each feature across all the data points and their respective cluster labels:

Feature	Mean value for Cluster 0	Mean value for Cluster 1
Channel	1.98	1.01
Region	2.64	2.5
Fresh	9575.61	13131.82
Milk	11447.61	3158.97
Grocery	16774.73	3833.66
Frozen	1880.78	3627.80
Detergents_Paper	7389.94	777.55
Delicassen	2149.18	1233.52

8. The below bar chart is used for visually representing the differences in mean spendings on each feature or category across all the data points between the two cluster:



9. From the above bar plot and the mean table, we can say that people in cluster 0 have more average spendings on Milk, Groceries and Detergent_papers compare to people in cluster 1 i.e. people in cluster 0 are tend to spend more on Milk, Groceries and Detergent_papers, whereas people in cluster 1 have more average spendings on Fresh and Frozen i.e. people in cluster 1 are tend to spend more on Fresh and Frozen things.

5. Discussion:

5.1 K-Means Clustering:

- **Strengths:**
    1. The concepts of centroid in K-Means helps in representing average profile of each cluster, which makes it easy to understand the clusters
    2. It is computationally efficient, and ideal for large data sets as it can handle high dimensional data.
    3. This algorithm updates the centres of the clusters iteratively and will adjust them based on the WCSS scores for providing better clustering.
- **Weaknesses:**
    1. The initial position of the centroids may affect the results. Using methods like K-Means++ may help in such cases but it cannot eliminate this adjustment.
    2. K-Means algorithm assume that the clusters are spherical in shape and are equally sized.
    3. K-Means algorithm requires an assumption regarding the number of clusters in advance and this process of assumption might not always be accurate.

5.2 Agglomerative (Hierarchical) Clustering:

- **Strengths:**
    1. This algorithm doesn't assume that the clusters are of spherical shape and works well with non-spherical clusters of uneven size.
    2. We don't need to specify the number of clusters in advance.

- **Weaknesses:**
    1. These are computationally expensive as they have some complex calculations involved in the implementation of the clustering.
    2. Suitable only for small and medium datasets.

5.3 DBSCAN Clustering:

- **Strengths:**
    1. Unlike algorithms like K-Means for DBSCAN we don't need to don't need to specify number of clusters in advance.
    2. DBSCAN is capable of handling outliers and noise and can separate them form the clusters.
    3. DBSCAN can find clusters of any shape and doesn't need the assumption of a particular shape for the clusters. This makes it better for real world data.
- **Weaknesses:**
    1. DBSCAN is highly sensitive to the initial defined parameters like Epsilon and MinPts and finding the right values for these parameters can be challenging.
    2. DBSCAN might struggle with higher dimensional data as the number of dimensions increase the points might get far away from each other and this effects the density.

6. Conclusion:

From our analysis, we conclude that while all the algorithms provide clear segmentation of data points, from the WCSS results we can say that the **K-**Means clustering algorithm performed slightly better than Agglomerative Clustering. K-Means effectively separates customers based on average spending in each category, creating distinct groups that reflect meaningful purchasing behaviours across all data points. Based on the Silhouette Score we can say that DBSCAN performed better compared to the other two with the highest score of 0.4123. This explains that DBSCAN works better with data that might not have any predefined cluster shapes and can also handles noise and outliers. So, if a new data point is generated K-Means will assign it to a cluster whose centroid is near to the new data point.

Reference

1. Cardoso, M. (2013). Wholesale customers [Dataset]. UCI Machine Learning Repository. https://archive.ics.uci.edu/dataset/292/wholesale+customers.

Python Code:

```
# importing required libraries

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN

from sklearn.metrics import silhouette_score

from sklearn.decomposition import PCA
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

DATA PROCESSING:

```
# Load the dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-
databases/00292/Wholesale%20customers%20data.csv"
data = pd.read_csv(url)
print(type(data))
print(data.head())
# Define all features or columns for clustering
all_features = data[['Channel', 'Region', 'Fresh', 'Milk', 'Grocery', 'Frozen', 'Detergents_Paper', 'Delicassen']]
print(all_features.head())
print(type(all_features))
# scaling all features
scaler = StandardScaler()
data_scaled = scaler.fit_transform(all_features)
data_scaled
print(data_scaled[:5 ])
print(type(data_scaled))
```

K-Means Clustering:

```
# Using Silhouette score for finding best number of clusters.
silhouette_scores = []
for n_clusters in range(2, 11):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42, init='k-means++')
    cluster_labels = kmeans.fit_predict(data_scaled)
    silhouette_avg = silhouette_score(data_scaled, cluster_labels)
    silhouette_scores.append(silhouette_avg)
    print(f"number of clusters = {n_clusters}, silhouette score : {silhouette_avg}")

# Choose best number of clusters based on the highest silhouette score
```

```
best_k = silhouette_scores.index(max(silhouette_scores)) + 2
print(f"best number of clusters based on silhouette score: {best_k}")

# Apply K-Means with the best number of clusters
kmeans = KMeans(n_clusters=best_k, random_state=42)
kmeans_labels = kmeans.fit_predict(data_scaled)
silhouette_kmeans = silhouette_score(data_scaled, kmeans_labels)
print(f"K-means Silhouette Score with {best_k} clusters: {silhouette_kmeans}")

#WCSS score
wcss_2_clusters = kmeans.inertia_
print(f"WCSS for 2 clusters: {wcss_2_clusters}")

# Apply PCA to reduce the data to two dimensions for better plotting.
pca = PCA(n_components=2)
data_pca = pca.fit_transform(data_scaled)

# Visualize K-means clusters with PCA components
plt.scatter(data_pca[:, 0], data_pca[:, 1], c=kmeans_labels, cmap='viridis')
plt.scatter(pca.transform(kmeans.cluster_centers_)[:, 0], pca.transform(kmeans.cluster_centers_)[:, 1],
        s=300, c='red', marker='X')
plt.title(f"K-means Clustering with {best_k} Clusters (PCA Reduced)")
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.show()
```

Agglomerative Hierarchical clustering:

```
# Apply Hierarchical Clustering (Agglomerative Clustering) with the same number of clusters
agglo = AgglomerativeClustering(n_clusters=best_k, metric='euclidean', linkage='ward')
agglo_labels = agglo.fit_predict(data_scaled)
silhouette_agglo = silhouette_score(data_scaled, agglo_labels)
print("Hierarchical Clustering Silhouette Score with 2 clusters:", silhouette_agglo)
```

```
from sklearn.cluster import AgglomerativeClustering
import numpy as np

# Apply Agglomerative Clustering with 2 clusters
agglo = AgglomerativeClustering(n_clusters=2, linkage='ward')
agglo_labels = agglo.fit_predict(data_scaled)

# WCSS for Agglomerative Clustering
wcss_agglo = 0
for cluster_label in np.unique(agglo_labels):
    cluster_points = data_scaled[agglo_labels == cluster_label]
    cluster_centroid = cluster_points.mean(axis=0)
    # Sum of squared distances to the cluster centroid
    wcss_agglo += np.sum((cluster_points - cluster_centroid) ** 2)
print("WCSS for Agglomerative Clustering with 2 clusters:", wcss_agglo)

# Visualize Hierarchical Clustering (Agglomerative) with PCA components
plt.scatter(data_pca[:, 0], data_pca[:, 1], c=agglo_labels, cmap='viridis')
plt.title(f"Hierarchical Clustering (Agglomerative) with {best_k} Clusters (PCA Reduced)")
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.show()
# DBSCAN Clustering
dbscan = DBSCAN(eps=1.5, min_samples=5)  # Adjust eps and min_samples based on your data density
dbscan_labels = dbscan.fit_predict(data_scaled)

# Evaluate the clustering performance with the Silhouette Score, ignoring noise (-1)
non_noise_labels = dbscan_labels[dbscan_labels != -1]
non_noise_data = data_scaled[dbscan_labels != -1]

# Calculate Silhouette Score only if there are clusters (more than 1 unique non-noise label)
if len(set(non_noise_labels)) > 1:
```

```
    silhouette_dbscan = silhouette_score(non_noise_data, non_noise_labels)
    print(f"DBSCAN Silhouette Score (excluding noise): {silhouette_dbscan}")
else:
    print("Not enough clusters to calculate Silhouette Score.")



# Plot DBSCAN clusters using PCA components for visualization
plt.scatter(data_pca[:, 0], data_pca[:, 1], c=dbscan_labels, cmap='viridis', marker='o', s=25)
plt.title("DBSCAN Clustering (PCA Reduced)")
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.show()
```

Analysing the clusters results and purchasing patterns:

```
clustered_data = data.assign(Cluster=kmeans_labels)
print(clustered_data.head())
#Calculate the mean values for each feature in each cluster
cluster_means = clustered_data.groupby('Cluster').mean()
print("\nMean values per feature for each cluster (K-means):")
print(cluster_means)

# Bar plot of the average spending in each category per cluster
cluster_means_plot = cluster_means.drop(['Channel', 'Region'], axis=1)  # Drop categorical for easier plotting
cluster_means_plot.T.plot(kind='bar', figsize=(12, 8))
plt.title("Average Spending per Category in Each Cluster")
plt.xlabel("Spending Category")
plt.ylabel("Average Spending")
plt.show()
```