MLDL EXPERIMENT 6

Aim: Apply K-Means and Hierarchical Clustering on sample datasets.

Dataset Description

- **Context:** This dataset is created for learning market basket analysis. It contains information about customers of a supermarket mall who have membership cards.
- **Attributes:**
    - **CustomerID:** Unique ID assigned to the customer.
    - **Gender:** Gender of the customer (Male/Female).
    - **Age:** Age of the customer.
    - **Annual Income (k$):** Annual income of the customer in thousands of dollars.
    - **Spending Score (1-100):** A score assigned by the mall based on customer behavior and spending nature.
- **Objective:** Segment customers into groups based on their Annual Income and Spending Score to help the marketing team plan targeted strategies.

Dataset Source

https://www.kaggle.com/datasets/vjchoudhary7/customer-segmentation-tutorial-in-python

Theory

K-Means is a centroid-based, iterative, non-deterministic algorithm. It operates on the principle of minimizing variance within each cluster.

Mathematical Objective

The algorithm seeks to partition n observations into k sets S={S1,S2…,Sk} so as to minimize the Within-Cluster Sum of Squares (WCSS). The objective function is:

$$\boxed{J = \sum_{j=1}^{k} \sum_{i \in S_j} \left\| x_i - \mu_j \right\|^2}$$

Distance Metric

The standard implementation uses Euclidean Distance to determine the proximity of points to centroids:

$$\boxed{d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}}$$

Because this distance is sensitive to the magnitude of numbers, Feature Scaling (Standardization) is mandatory to ensure that "Annual Income" (thousands) and "Age" (tens) are weighted equally.

Limitations:

- **Fixed Cluster Count:** The algorithm cannot "learn" the number of clusters on its own; it must be told.
- **Spherical Bias:** It assumes clusters are roughly circular/spherical. If the data is "C-shaped" or elongated, K-Means will split it incorrectly.
- **Scale Sensitivity:** Without normalization, the algorithm fails because features with larger numerical ranges dominate the distance calculation.
- **Impact of Outliers:** Since centroids are calculated as a mean, a single extreme outlier can pull the cluster center away from the dense population of data.

Workflow

The workflow for implementing K-Means clustering on the Mall Customers dataset follows a standard machine learning pipeline for unsupervised learning. Here is the structured breakdown of the process:

1. Data Acquisition & Inspection

- Action: Load the Mall_Customers.csv file.
- Goal: Check data types, handle missing values (if any), and understand the distribution of variables like Age, Annual Income, and Spending Score.

2. Feature Selection

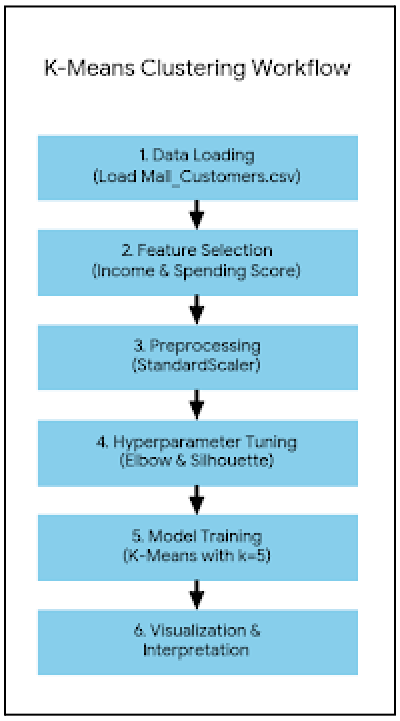
- Action: Select specific columns for clustering.
- Reason: In this dataset, Annual Income and Spending Score are the most effective features for identifying distinct marketing segments (e.g., high-income spenders vs. low-income savers).

3. Data Preprocessing (Scaling)

- Action: Apply StandardScaler.
- Reason: K-Means uses Euclidean Distance to measure similarity. Since Income (thousands) and Spending Score (1-100) have different scales, standardization ensures that both features contribute equally to the distance calculation.

4. Hyperparameter Tuning (Optimal k)

- Elbow Method: We run K-Means for k values from 1 to 10 and plot the WCSS (Within-Cluster Sum of Squares). We look for the "Elbow" point where the rate of decrease slows down.
- Silhouette Score: We calculate the silhouette coefficient for each k. A higher score indicates that points are well-matched to their own cluster and poorly matched to neighboring clusters.
- Selection: For this dataset, k=5 is typically the optimal balance between granularity and simplicity.


K-Means Clustering Workflow
1. Data Loading (Load Mall_Customers.csv)
2. Feature Selection (Income & Spending Score)
3. Preprocessing (StandardScaler)
4. Hyperparameter Tuning (Elbow & Silhouette)
5. Model Training (K-Means with k=5)
6. Visualization & Interpretation

5. Model Training

- Action: Initialize K-Means with the chosen k and init='k-means++'.
- Goal: The k-means++ initialization helps prevent the algorithm from getting stuck in poor local minima by spreading out the initial centroids.

6. Cluster Visualization & Interpretation

- Action: Plot the results on a 2D scatter plot and assign names to the clusters.
- Persona Mapping:
  - Cluster 1: Low Income, High Spending (Sensible/Careless)
  - Cluster 2: High Income, Low Spending (Frugal/Careful)
  - Cluster 3: Average Income, Average Spending (Middle Class)
  - Cluster 4: High Income, High Spending (Target/Luxury Customers)
  - Cluster 5: Low Income, Low Spending (Budget Conscious)

7. Business Deployment

- Action: Save the labels back to the original database.
- Goal: Use these segments to tailor marketing strategies (e.g., send luxury catalogs to Cluster 4 and discount coupons to Cluster 5).

Performance Analysis:

In unsupervised learning, we don't have "correct" labels to compare against. Instead, we analyze how well-separated and compact the clusters are.

Within-Cluster Sum of Squares (WCSS)

WCSS measures the compactness of the clusters. It is the sum of the squared Euclidean distances between each data point and its assigned cluster centroid.

- **The Math:** As the number of clusters (k) increases, WCSS naturally decreases because the centroids get closer to the points.
- **The Goal:** We want a low WCSS, but not so low that every point is its own cluster (overfitting).

Silhouette Score

This is a more sophisticated metric that measures both cohesion (how close a point is to its own cluster) and separation (how far it is from other clusters).

- **Value Range: -1 to +1.**
  - **Near +1:** The point is far from neighboring clusters (Good).
  - **Near 0:** The point is on or very close to the decision boundary between two neighboring clusters.
  - **Negative:** The point might have been assigned to the wrong cluster.

Hyperparameter Tuning

Hyperparameter tuning in K-Means primarily revolves around selecting the number of clusters (k). Since you can't "learn" k during training, we use heuristic methods.

The Elbow Method

You plotted WCSS against k. The "Elbow" is the point where the rate of decrease in WCSS shifts from vertical to horizontal.

- **Theory:** Before the elbow, adding a cluster significantly improves the fit. After the elbow, adding more clusters provides diminishing returns and likely starts capturing noise.

Silhouette Analysis

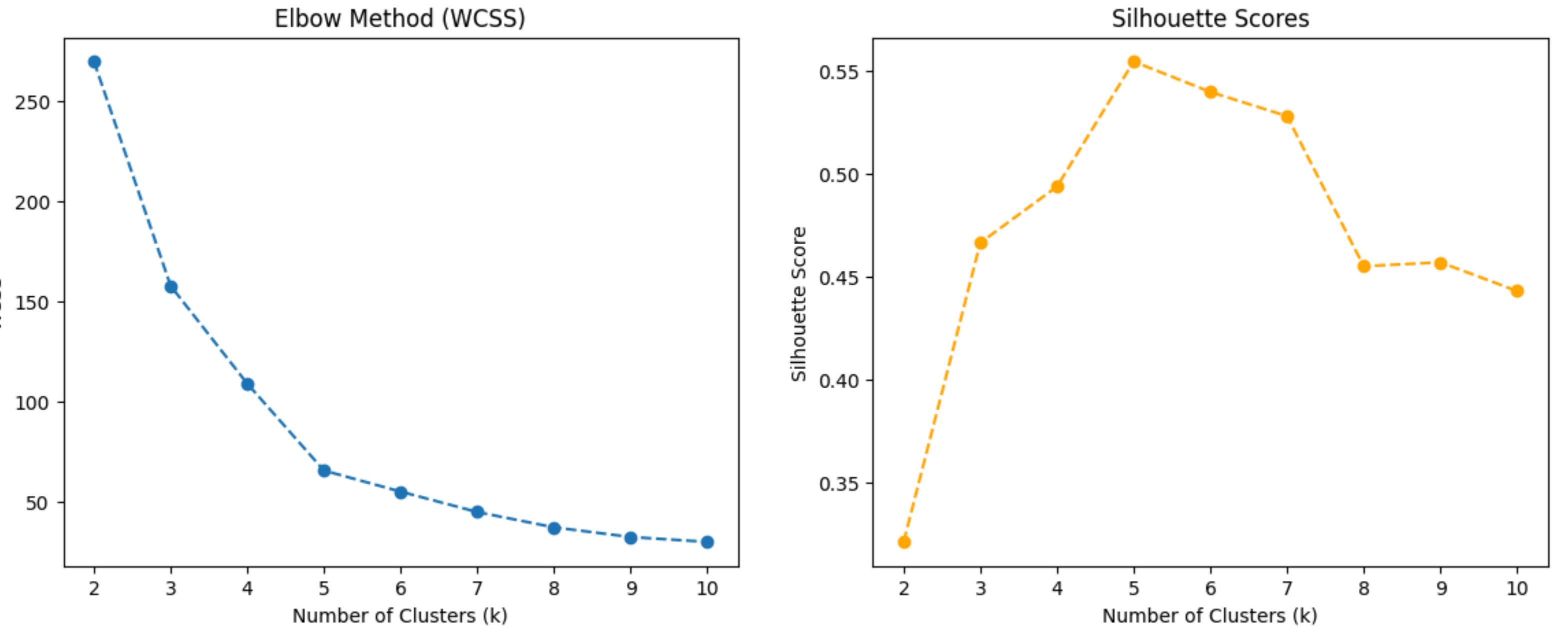
While the Elbow method is sometimes ambiguous, the Silhouette Score provides a clear peak.

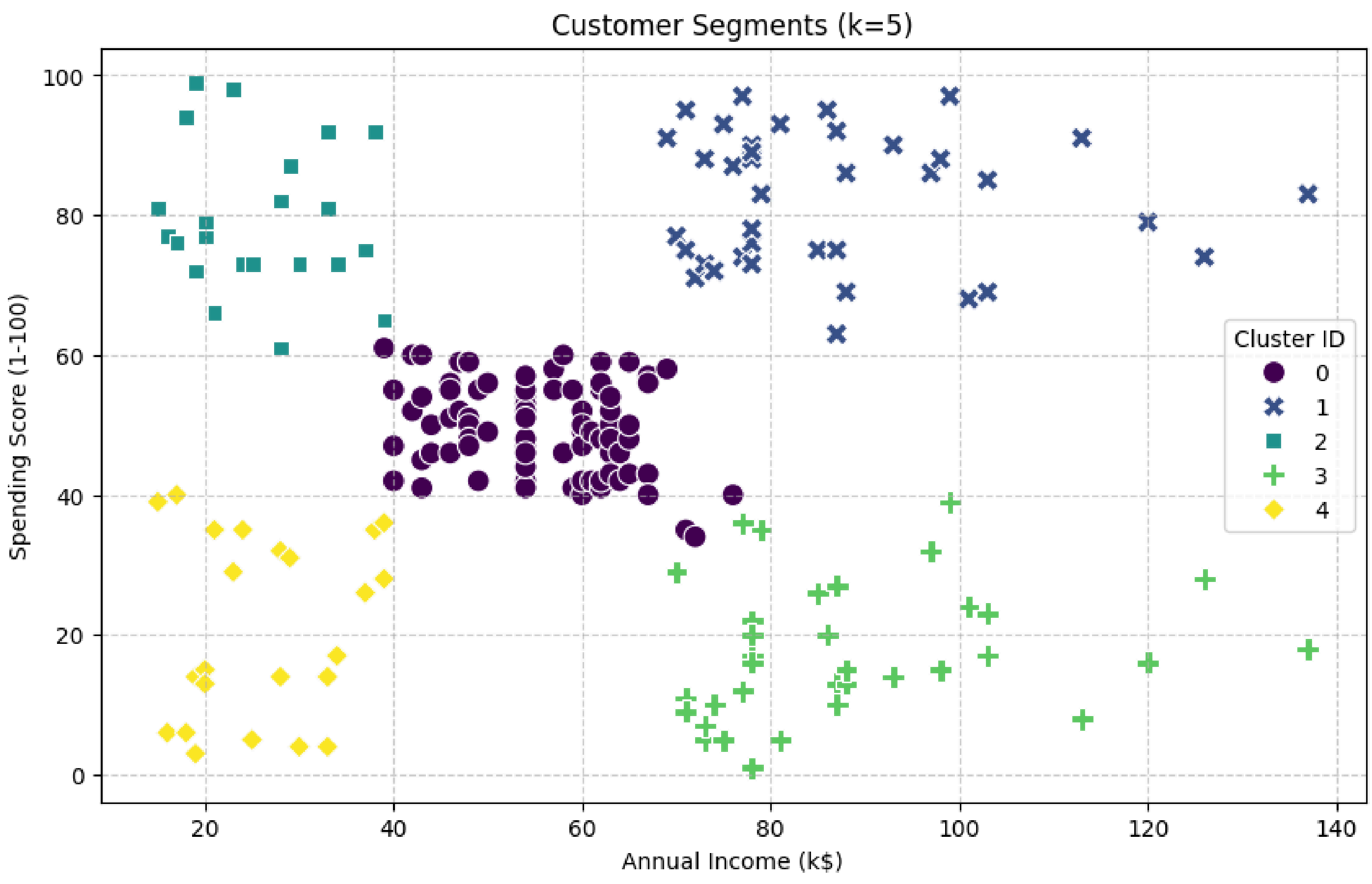
- **Theory:** We choose the k that maximizes the average Silhouette Score across all observations. In your specific results, you likely noticed a peak at k=5, confirming it as a statistically robust choice.

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import
StandardScaler
from sklearn.metrics import silhouette_score

# 1. Load the dataset
df = pd.read_csv('Mall_Customers.csv')

# 2. Feature Selection
# We use 'Annual Income' and 'Spending
Score' for segmenting customers
X = df[['Annual Income (k$)', 'Spending Score
(1-100)']]

# 3. Feature Scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 4. Hyperparameter Tuning (Finding optimal
k)
wcss = []
silhouette_scores = []
k_range = range(2, 11)

for k in k_range:
    # 'k-means++' ensures better initialization of
centroids
    kmeans = KMeans(n_clusters=k,
init='k-means++', random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

silhouette_scores.append(silhouette_score(X_
scaled, kmeans.labels_))

# Plotting Elbow and Silhouette Score for
Tuning
plt.figure(figsize=(14, 5))

plt.subplot(1, 2, 1)
plt.plot(k_range, wcss, marker='o',
linestyle='--')
plt.title('Elbow Method (WCSS)')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('WCSS')

plt.subplot(1, 2, 2)
plt.plot(k_range, silhouette_scores, marker='o',
linestyle='--', color='orange')
plt.title('Silhouette Scores')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Silhouette Score')

plt.show()

# 5. Training the Final Model
# Based on the plots, k=5 is the optimal
number of clusters
best_k = 5
kmeans_final = KMeans(n_clusters=best_k,
init='k-means++', random_state=42, n_init=10)
df['Cluster'] =
kmeans_final.fit_predict(X_scaled)

# 6. Visualization of Clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(
    data=df,
    x='Annual Income (k$)',
    y='Spending Score (1-100)',
    hue='Cluster',
    palette='viridis',
    s=100,
    style='Cluster'
)

plt.title(f'Customer Segments (k={best_k})')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend(title='Cluster ID')
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()

# Output segment counts
print("Customer counts per cluster:")
print(df['Cluster'].value_counts())
```

Output:


Elbow Method (WCSS)
250
200
150
100
50
2 3 4 5 6 7 8 9 10
Number of Clusters (k)
Silhouette Scores
Silhouette Score
0.55
0.50
0.45
0.40
0.35
2 3 4 5 6 7 8 9 10
Number of Clusters (k)


Customer Segments (k=5)
Spending Score (1-100)
100
80
60
40
20
0
20 40 60 80 100 120 140
Annual Income (k$)
Cluster ID
● 0
✖ 1
■ 2
✚ 3
◆ 4

Hierarchical Clustering

Theory:

Hierarchical Clustering is an unsupervised machine learning technique that organizes data into a multilevel tree structure. Unlike K-Means, which requires you to pre-define the number of clusters (k), hierarchical clustering creates a flexible map of relationships that can be analyzed at various levels of detail.

Core Mechanics

The algorithm operates through two primary strategies:

- **Agglomerative (Bottom-Up):** Starting with each data point as its own cluster, it progressively merges the most similar pairs until only one cluster remains.
- **Divisive (Top-Down):** Starting with the entire dataset as one cluster, it repeatedly splits groups into smaller, more distinct units.

Key Components

To build this hierarchy, the algorithm utilizes two fundamental measures:

1. **Distance Metric:** Determines how the "closeness" between individual data points is calculated (e.g., Euclidean or Manhattan distance).
2. **Linkage Criterion:** Defines how to measure the distance between groups of points to decide which clusters should be combined or split.

The Dendrogram

The primary output of this process is a dendrogram—a tree-like diagram that illustrates how clusters are nested. This visualization is a powerful tool for exploratory data analysis, as it allows researchers to "cut" the tree at different heights to discover the most natural number of clusters for their specific needs.

Limitations

- **Computational Complexity**: It is significantly slower than K-Means. The complexity is typically O(n^2) or O(n^3), making it impractical for very large datasets (e.g., millions of rows).
- **No "Undo" Button**: Once two clusters are merged, they cannot be separated later in the process. If a poor merge happens early due to noise, it persists through the entire hierarchy.
- **Sensitivity to Noise and Outliers**: A single outlier can significantly distort the tree structure, especially in single-linkage clustering.
- **Arbitrary Cutting**: While you don't need to define K initially, the choice of where to "cut" the dendrogram to get your final clusters is still subjective and requires domain knowledge.
- **Difficulty with Different Shapes**: Like K-Means, many linkage methods (like Ward's) struggle to identify clusters with non-spherical or irregular shapes.

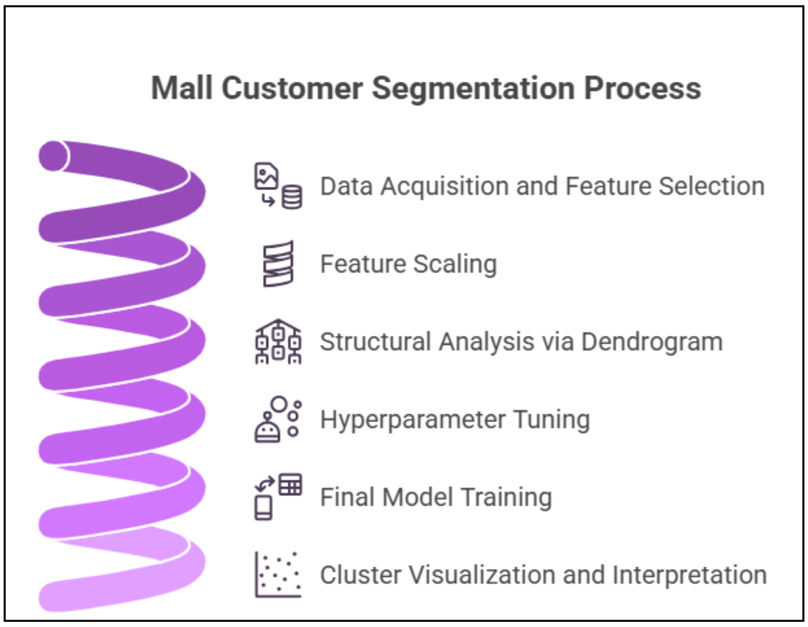
Workflow:

1. Data Acquisition and Feature Selection

- Action: Load the `Mall_Customers.csv` dataset.
- Logic: We extract specific features—Annual Income and Spending Score—because these columns directly represent consumer behavior. Other features like `CustomerID` are unique identifiers and do not provide patterns for clustering.

2. Feature Scaling (Standardization)

- Action: Apply `StandardScaler` to the feature set.
- Logic: Hierarchical clustering calculates distances (like Euclidean distance) between points. If one feature has a much larger range than another (e.g., Income in thousands vs. Score from 1–100), the larger variable will dominate the distance calculation. Scaling ensures all features contribute equally.


Mall Customer Segmentation Process
Data Acquisition and Feature Selection
Feature Scaling
Structural Analysis via Dendrogram
Hyperparameter Tuning
Final Model Training
Cluster Visualization and Interpretation

3. Structural Analysis via Dendrogram

- Action: Generate a tree-like diagram using `scipy`'s linkage functions.
- Logic: This is an exploratory step. By looking at the vertical lines in the dendrogram, we can visually estimate the "natural" number of clusters. It helps validate if the mathematical "best" number found in the next step makes visual sense.

4. Hyperparameter Tuning (Grid Search)

- Action: Iterate through different Linkage Methods (`ward`, `complete`, `average`, `single`) and a range of Cluster Counts (2 to 10).
- Metric: Use the Silhouette Score to evaluate each combination.
  - Note: A silhouette score closer to 1 indicates that data points are well-clustered (close to their own group and far from others).

5. Final Model Training

- Action: Identify the parameters that yielded the highest Silhouette Score and initialize the final `AgglomerativeClustering` model.
- Logic: Instead of guessing the number of clusters, this step uses the results of the tuning phase to ensure the model is mathematically optimized for the specific distribution of the mall customer data.

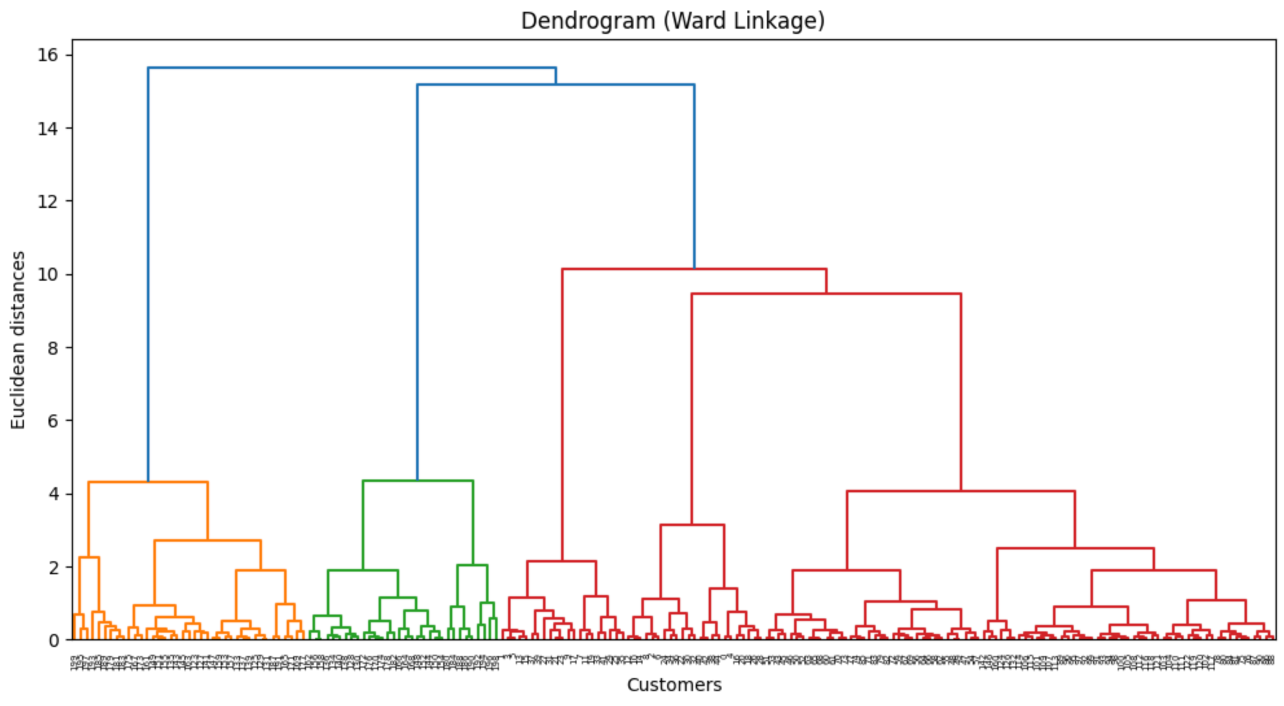
6. Cluster Visualization and Interpretation

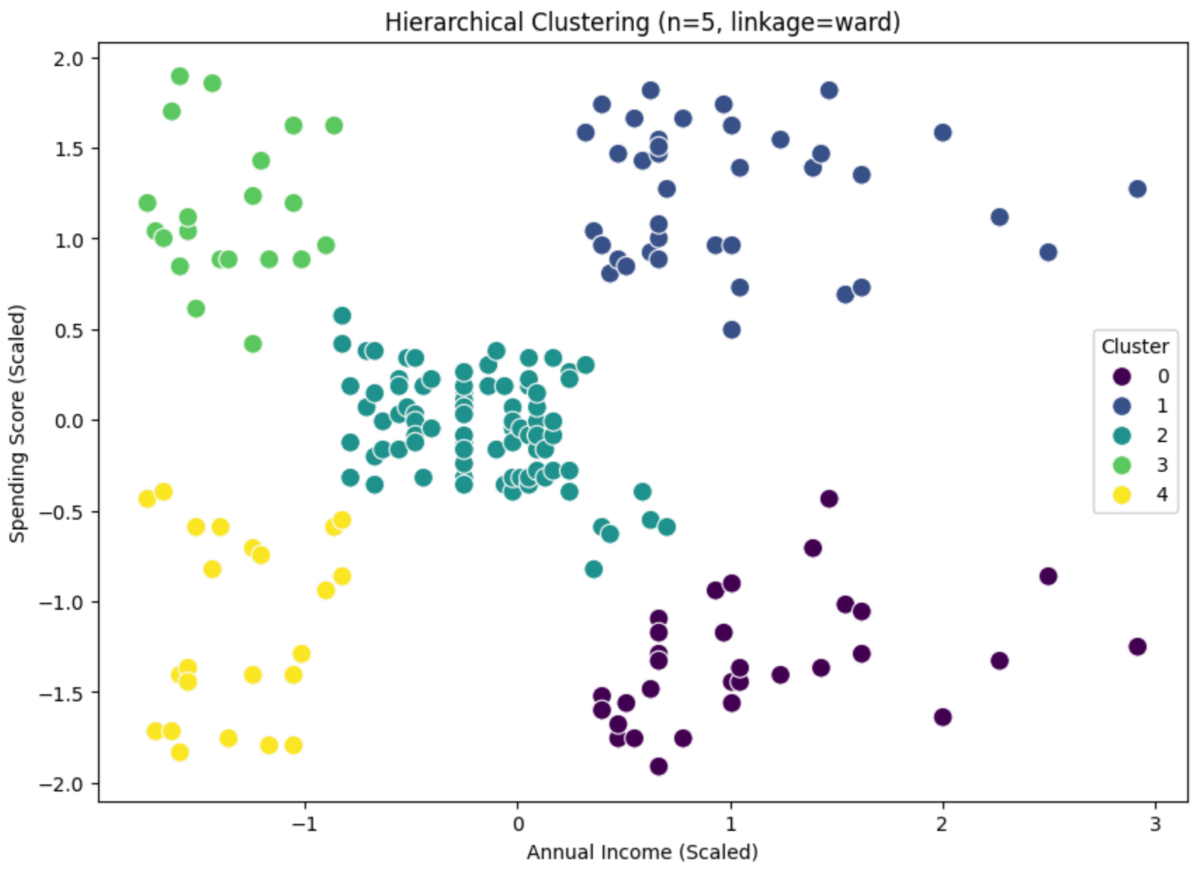
- Action: Create a 2D scatter plot where points are colored by their assigned cluster label.
- Logic: This transforms the mathematical output into a business insight. It allows you to see the distinct "segments" (e.g., "High Earners/High Spenders" vs. "High Earners/Low Spenders"), which is the ultimate goal of the analysis.

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import
StandardScaler
from sklearn.cluster import
AgglomerativeClustering
from sklearn.metrics import
silhouette_score
import scipy.cluster.hierarchy as sch

# 1. Load Data
df = pd.read_csv('Mall_Customers.csv')
X = df.iloc[:, [3, 4]].values # Annual
Income and Spending Score

# 2. Feature Scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 3. Dendrogram to visualize clusters
plt.figure(figsize=(12, 6))
dendrogram =
sch.dendrogram(sch.linkage(X_scaled,
method='ward'))
plt.title('Dendrogram (Ward Linkage)')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.savefig('dendrogram.png')

# 4. Hyperparameter Tuning (Linkage and
n_clusters)
linkage_methods = ['ward', 'complete',
'average', 'single']
n_clusters_range = range(2, 11)
results = []

for linkage in linkage_methods:
    for n in n_clusters_range:
        model =
AgglomerativeClustering(n_clusters=n,
linkage=linkage)
        labels = model.fit_predict(X_scaled)
        score = silhouette_score(X_scaled,
labels)
        results.append({'linkage': linkage,
'n_clusters': n, 'silhouette_score': score})

tuning_df = pd.DataFrame(results)
best_config =
tuning_df.loc[tuning_df['silhouette_score'].i
dxmax()]

print(f"Best Configuration:
Linkage={best_config['linkage']},
Clusters={best_config['n_clusters']}")
print(f"Max Silhouette Score:
{best_config['silhouette_score']:.4f}")

# 5. Final Model with Best Parameters
best_n = int(best_config['n_clusters'])
best_linkage = best_config['linkage']
final_model =
AgglomerativeClustering(n_clusters=best_
n, linkage=best_linkage)
y_hc = final_model.fit_predict(X_scaled)

# 6. Visualization of Final Clusters
plt.figure(figsize=(10, 7))
sns.scatterplot(x=X_scaled[:, 0],
y=X_scaled[:, 1], hue=y_hc,
palette='viridis', s=100)
plt.title('Hierarchical Clustering
(n={best_n}, linkage={best_linkage})')
plt.xlabel('Annual Income (Scaled)')
plt.ylabel('Spending Score (Scaled)')
plt.legend(title='Cluster')
plt.savefig('clusters_plot.png')
```

Output:




Hierarchical Clustering (n=5, linkage=ward)
Spending Score (Scaled)
Annual Income (Scaled)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1
0
1
2
3
Cluster
0
1
2
3
4