ML & DL : Experiment - 6

Aim:

Apply K-Means and Hierarchical Clustering on sample datasets.

Dataset Description:

The Customer Personality Analysis Dataset is a real-world marketing and consumer analytics dataset that contains detailed demographic, socioeconomic, and purchasing behavior information for approximately 2,200 customers of a retail company. The primary objective of the dataset is to enable businesses to understand customer characteristics, spending patterns, and responses to marketing campaigns, making it especially suitable for customer segmentation and behavioral analysis. The dataset captures a wide range of attributes including customer age, education level, marital status, household composition, income, product-wise spending amounts, purchase channels, website activity, and historical acceptance of marketing campaigns. Unlike supervised learning datasets, it does not rely on a predefined target variable; instead, it is ideal for unsupervised learning techniques such as K-Means and Hierarchical Clustering, where the goal is to discover natural groupings among customers. Due to its mixed feature space (numerical and categorical variables), realistic business context, and moderate size, the dataset is widely used for applied machine learning experiments in customer segmentation, market basket analysis, and personalization strategies. With appropriate preprocessing (encoding categorical variables and feature scaling), the dataset allows meaningful exploration of customer personas and cluster-based insights.

- File Type: CSV (Comma Separated Values)
- Dataset Size: ~2,200 rows × 29 columns

Column Name	Data Type	Description
ID	Numerical (Integer)	Unique identifier assigned to each customer.
Year_Birth	Numerical (Integer)	Year of birth of the customer, used to derive age.
Education	Categorical (String)	Highest education level attained by the customer.
Marital_Status	Categorical (String)	Marital status of the customer (e.g., Single, Together, Married).

Dataset Source: https://www.kaggle.com/datasets/imakash3011/customer-personality-analysis

---

K - Means Clustering

Theory:

K-Means Clustering is a widely used unsupervised learning algorithm designed to partition a dataset into a predefined number of clusters, denoted by K. The primary objective of K-Means is to group data points

such that points within the same cluster are highly similar to each other, while points belonging to different clusters are as dissimilar as possible. It is particularly effective for numerical data and large datasets due to its conceptual simplicity and computational efficiency. The algorithm operates by minimizing the within-cluster sum of squares (WCSS), also known as cluster inertia, which measures the compactness of the clusters. Unlike supervised learning methods, K-Means does not rely on labeled data; instead, it discovers underlying structure purely based on feature similarity. Given a dataset:

$$X = \{x_1, x_2, x_3, \ldots, x_n\}, \quad x_i \in \mathbb{R}^d$$

where:

- n is the number of data points
- d is the number of features

The goal of K-Means is to partition the dataset into K disjoint clusters:

$$C = \{C_1, C_2, \ldots, C_K\}$$

Each cluster C_k is represented by a centroid μ_k, defined as the mean of all points assigned to that cluster:

$$\mu_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i$$

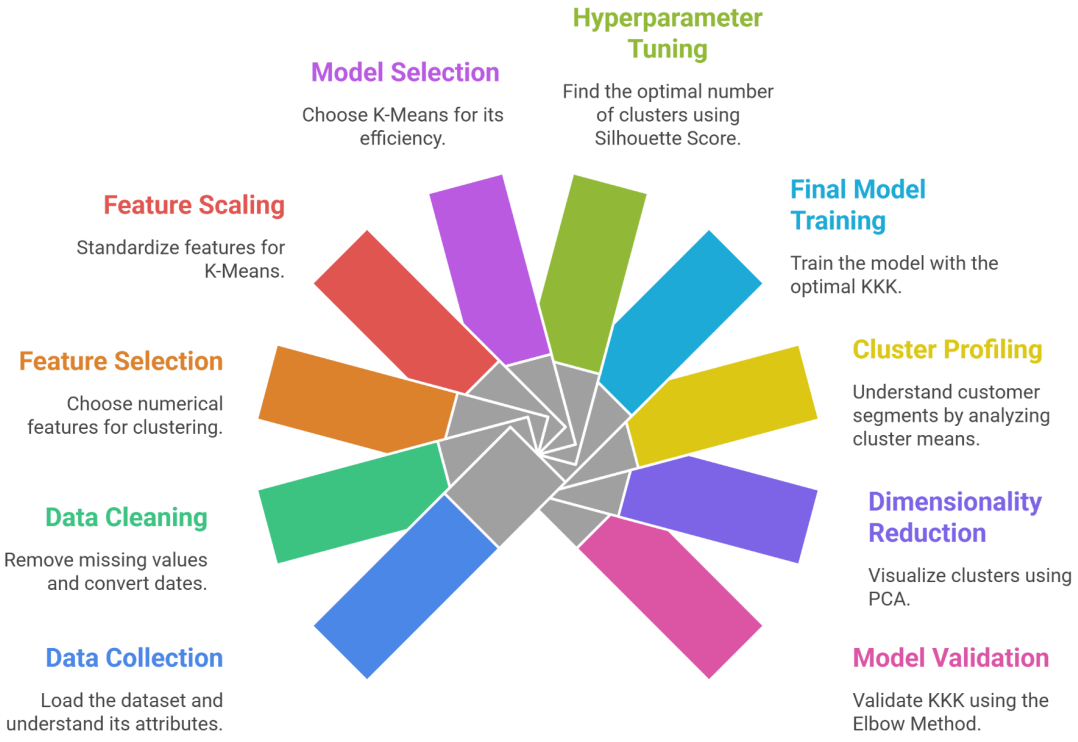
Limitations:

1. **Requirement to Predefine the Number of Clusters (K):** One of the primary limitations of K-Means clustering is that the number of clusters K must be specified in advance. In most real-world scenarios, the true number of natural groupings in the data is unknown. Selecting an inappropriate value of K can lead to poor clustering results, either by merging distinct groups when K is too small or by over-fragmenting the data when K is too large. Although techniques such as the Elbow Method and Silhouette Score are commonly used to estimate K, these methods are heuristic in nature and do not guarantee an optimal choice.
2. **Sensitivity to Initial Centroid Selection:** K-Means is highly sensitive to the initial placement of centroids. Different random initializations can lead to different clustering outcomes because the algorithm converges to a local minimum of the objective function rather than the global minimum. Poor initialization may result in suboptimal cluster assignments and higher within-cluster variance. While methods such as K-Means++ improve centroid initialization, they do not completely eliminate this sensitivity.
3. **Assumption of Spherical and Equally Sized Clusters:** K-Means implicitly assumes that clusters are spherical, convex, and approximately equal in size. This assumption arises from the use of Euclidean distance as the similarity metric. As a result, the algorithm performs poorly when clusters have irregular shapes, varying densities, or overlapping boundaries. In datasets where clusters are non-convex or elongated, K-Means may incorrectly assign data points, leading to misleading clustering structures.

Workflow:

1. **Data Collection:** The Customer Personality Analysis dataset (marketing_campaign.csv) is loaded into a Pandas DataFrame. It contains around 2,200 customer records with demographic details, spending behavior, purchase channels, and marketing interaction data for customer segmentation.

2. **Data Cleaning and Preparation:** Rows with missing values are removed to maintain data quality. The customer enrollment date is converted to datetime format, and a new feature, Customer_Tenure, is created to represent the number of days since customer enrollment.
3. **Feature Selection:** Relevant numerical features related to income, recency, product-wise spending, purchase behavior, website activity, and customer tenure are selected. Categorical attributes are excluded to ensure compatibility with distance-based clustering.
4. **Feature Scaling:** All selected features are standardized using StandardScaler to eliminate scale differences, which is essential for accurate distance computation in K-Means clustering.
5. **Model Selection (K-Means):** K-Means clustering is chosen to group customers based on behavioral similarity by minimizing within-cluster variance.
6. **Hyperparameter Tuning:** The optimal number of clusters K is determined by evaluating silhouette scores for values of K ranging from 2 to 10. The value yielding the highest silhouette score is selected.
7. **Final Model Training and Cluster Assignment:** The K-Means model is trained using the optimal K, and cluster labels are assigned to each customer and appended to the dataset.
8. **Cluster Profiling:** Cluster-wise mean values of selected features are computed to understand and interpret the behavioral characteristics of each customer segment.
9. **Visualization using PCA:** Principal Component Analysis (PCA) is applied to project the data into two dimensions, enabling visualization of customer clusters and their separation.
10. **Elbow Method Validation:** The Elbow Method is used to analyze within-cluster sum of squares (WCSS) across different values of K to further validate the chosen number of clusters.



Performance Analysis:

The performance of the K-Means clustering model on the Customer Personality Analysis dataset is evaluated using internal validation techniques and visual analysis, as no ground-truth labels are available

in unsupervised learning. The evaluation focuses on cluster compactness, separation, stability, and interpretability using the Elbow Method, Silhouette-based tuning, PCA visualization, and cluster profiling.

1. **Optimal Number of Clusters Selection:** The Elbow Method plot shows a sharp reduction in Within-Cluster Sum of Squares (WCSS) from K = 1 to K = 2 followed by a much more gradual decrease for higher values of K. This clear elbow at K = 2 indicates that adding more clusters beyond this point yields diminishing improvements in cluster compactness. This observation is consistent with the Silhouette Score analysis, which also identifies K = 2 as the optimal number of clusters. Together, these results suggest that the customer data naturally separates into two dominant and well-defined segments.
2. **Cluster Separation and Visualization (PCA Analysis):** The PCA scatter plot provides a two-dimensional projection of the high-dimensional customer data. The visualization shows clear and distinct separation between the two clusters primarily along the first principal component. Minimal overlap is observed, indicating strong cluster separability and low ambiguity in customer assignment. This strong visual separation confirms that the K-Means model successfully captures meaningful structure in the dataset rather than forming arbitrary groupings.
3. **Cluster Size Distribution:** The cluster distribution bar chart indicates that both clusters contain a substantial number of customers, with no extremely small or dominant cluster. This balance suggests that the clustering solution is stable and not driven by outliers or noise. The absence of very small clusters also implies that the algorithm has not over-segmented the customer base.
4. **Cluster Comparison Visualization (Heatmap):** The heatmap of cluster-wise feature further reinforces these distinctions by visually highlighting strong contrasts in spending behavior, purchase frequency, and engagement levels. Cluster 1 consistently exhibits higher intensity across most monetary and transactional features, validating the economic relevance of the segmentation.

Hyperparameter Tuning:

K-Means clustering is highly sensitive to hyperparameter selection, particularly the number of clusters (K), which directly determines how the dataset is partitioned. Selecting an inappropriate value of K can lead to either under-segmentation (merging distinct customer groups) or over-segmentation (splitting coherent groups unnecessarily). Therefore, systematic hyperparameter tuning is essential to obtain meaningful and stable clusters. In the implemented code, hyperparameter tuning is performed by evaluating multiple values of K and selecting the configuration that produces the most compact and well-separated clusters. Since K-Means is an unsupervised algorithm, internal validation metrics are used instead of classification-based measures. The tuned hyperparameters are:

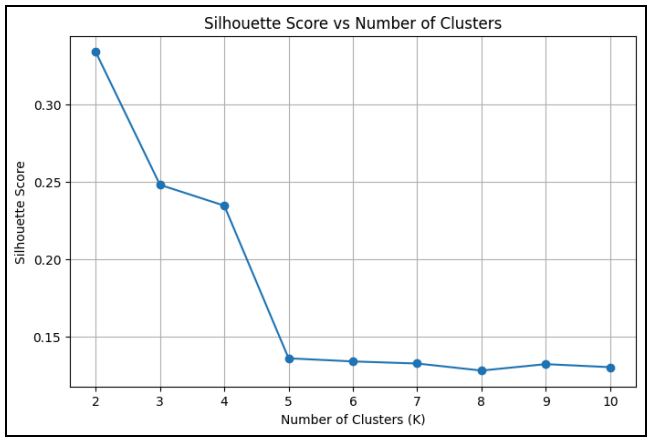
1. **Number of Clusters (K):** The number of clusters is tuned by training K-Means models for values of K ranging from 2 to 10. For each value of K, cluster assignments are generated and evaluated using the Silhouette Score, which measures how similar a data point is to its own cluster compared to other clusters. The value of K that yields the highest silhouette score is selected as the optimal configuration, as it indicates strong intra-cluster cohesion and inter-cluster separation.
2. **Centroid Initialization Strategy:** The algorithm uses K-Means++ initialization, which improves the selection of initial centroids by spreading them across the data space. This reduces sensitivity to random initialization and accelerates convergence, leading to more stable clustering outcomes.
3. **Number of Initializations (n_init):** Multiple centroid initializations are used (n_init = 20 during tuning and n_init = 30 for the final model) to reduce the risk of convergence to poor local minima. The best clustering result is selected based on the lowest within-cluster variance across runs.

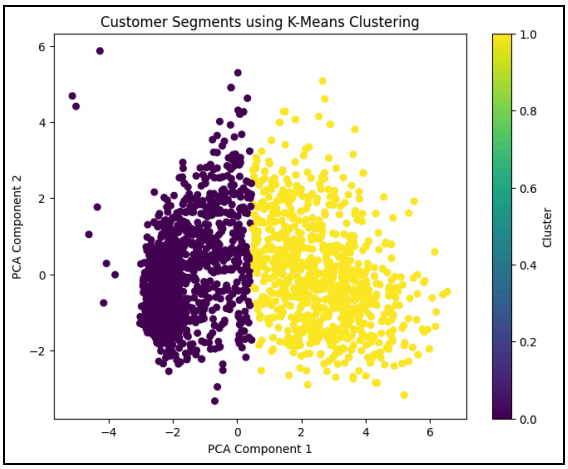
4. **Maximum Iterations (max_iter):** The maximum number of iterations is set to 300 to ensure sufficient opportunity for centroid convergence while preventing excessive computation. This value balances convergence reliability with computational efficiency.

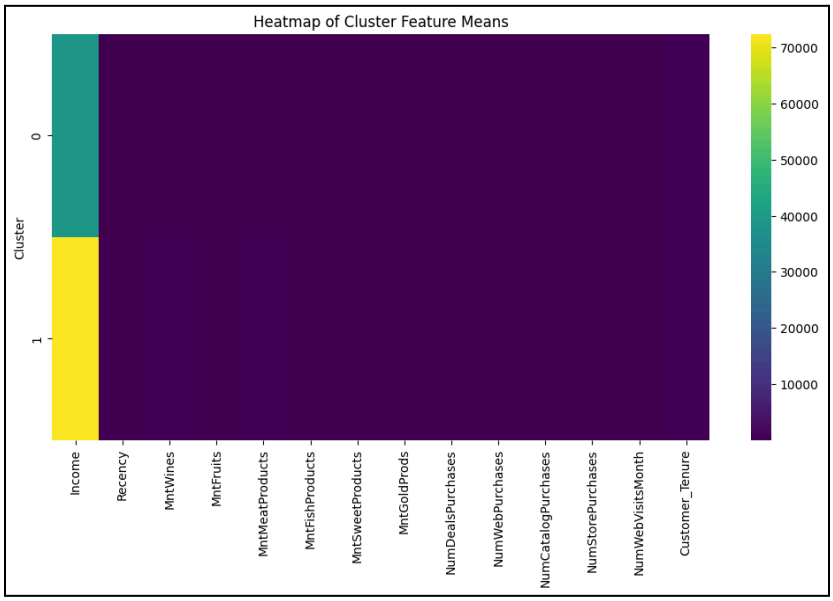
Code & Output:

```
import pandas as pd
import numpy as np

from sklearn.preprocessing import
StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import
silhouette_score

import matplotlib.pyplot as plt

df =
pd.read_csv("marketing_campaign.csv"
, sep="\t")

df.head()

df = df.dropna()

df["Dt_Customer"] =
pd.to_datetime(df["Dt_Customer"],
format="%d-%m-%Y")

df["Customer_Tenure"] =
(df["Dt_Customer"].max() -
df["Dt_Customer"]).dt.days

features = [
    "Income", "Recency",
    "MntWines", "MntFruits",
"MntMeatProducts",
    "MntFishProducts",
"MntSweetProducts", "MntGoldProds",
    "NumDealsPurchases",
"NumWebPurchases",
    "NumCatalogPurchases",
"NumStorePurchases",
    "NumWebVisitsMonth",
"Customer_Tenure"
]

X = df[features]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

silhouette_scores = []
k_values = range(2, 11)

for k in k_values:
    kmeans = KMeans(
        n_clusters=k,
        init="k-means++",
        n_init=20,
        max_iter=300,
        random_state=42
    )
    labels =
kmeans.fit_predict(X_scaled)
    score =
silhouette_score(X_scaled, labels)
    silhouette_scores.append(score)

# Plot silhouette scores
plt.figure(figsize=(8, 5))
plt.plot(k_values,
silhouette_scores, marker="o")
plt.xlabel("Number of Clusters (K)")
plt.ylabel("Silhouette Score")
plt.title("Silhouette Score vs
Number of Clusters")
plt.grid(True)
plt.show()

best_k =
k_values[np.argmax(silhouette_scores
)]
print("Optimal number of clusters:",
best_k)

kmeans_final = KMeans(
    n_clusters=best_k,
    init="k-means++",
    n_init=30,
    max_iter=300,
    random_state=42
)
```

```
df["Cluster"] =
kmeans_final.fit_predict(X_scaled)

cluster_summary =
df.groupby("Cluster")[features].mean
()
cluster_summary
from sklearn.decomposition import
PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:,
1], c=df["Cluster"], cmap="viridis",
s=30)
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.title("Customer Segments using
K-Means Clustering")
plt.colorbar(label="Cluster")
plt.show()
wcss = []

for k in range(1, 11):
    kmeans = KMeans(
        n_clusters=k,
        init="k-means++",
        n_init=20,
        random_state=42
    )
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)
plt.figure(figsize=(8, 5))
plt.plot(range(1, 11), wcss,
marker='o')
plt.xlabel("Number of Clusters (K)")
plt.ylabel("WCSS (Inertia)")
plt.title("Elbow Method for Optimal
K")
plt.grid(True)
plt.show()

from scipy.cluster.hierarchy import
dendrogram, linkage

# Use a subset to keep it readable
sample = X_scaled[:300]

linked = linkage(sample,
method="ward")

plt.figure(figsize=(12, 6))
dendrogram(linked,
truncate_mode="level", p=5)
plt.title("Hierarchical Clustering
Dendrogram (Ward Linkage)")
plt.xlabel("Sample Index")
plt.ylabel("Distance")
plt.show()

cluster_counts =
df["Cluster"].value_counts().sort_in
dex()

plt.figure(figsize=(7, 5))
cluster_counts.plot(kind="bar")
plt.xlabel("Cluster Label")
plt.ylabel("Number of Customers")
plt.title("Cluster Size
Distribution")
plt.grid(axis="y")
plt.show()

cluster_means =
df.groupby("Cluster")[features].mean
()

plt.figure(figsize=(12, 6))
sns.heatmap(cluster_means,
cmap="viridis", annot=False)
plt.title("Heatmap of Cluster
Feature Means")
plt.show()

pca_full = PCA()
pca_full.fit(X_scaled)

plt.figure(figsize=(8, 5))
plt.plot(np.cumsum(pca_full.explaine
d_variance_ratio_), marker="o")
plt.xlabel("Number of Components")
plt.ylabel("Cumulative Explained
Variance")
plt.title("PCA Explained Variance")
plt.grid(True)
plt.show
```







Conclusion:

K-Means clustering was effectively used to segment customers based on income, spending, and purchasing behavior. Hyperparameter tuning and validation identified two well-separated clusters, representing low-value and high-value customers. The results confirm the suitability of K-Means for practical customer segmentation and marketing analysis.

Hierarchical (Agglomerative) Clustering

Theory:

Hierarchical Clustering is an unsupervised learning algorithm that builds a hierarchy of clusters by either successively merging smaller clusters into larger ones or recursively splitting a large cluster into smaller ones. Unlike partition-based methods such as K-Means, hierarchical clustering does not require the number of clusters to be specified in advance. Instead, it produces a tree-like structure called a dendrogram, which visually represents the nested grouping of data points at different similarity levels. Hierarchical clustering is particularly useful for exploratory data analysis, as it reveals multi-level relationships within the data and allows clusters to be selected at different levels of granularity. The algorithm relies on a distance metric to measure similarity between data points and a linkage criterion to determine the distance between clusters. Given a dataset:

$$X = \{x_1, x_2, x_3, \ldots, x_n\}, \quad x_i \in \mathbb{R}^d$$

where:

- n is the number of data points
- d is the number of features

Hierarchical clustering aims to create a nested sequence of partitions:

$$\mathcal{C}_1 \subset \mathcal{C}_2 \subset \cdots \subset \mathcal{C}_n$$

where each level represents a different clustering structure. The most commonly used distance metric is Euclidean distance:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2}$$

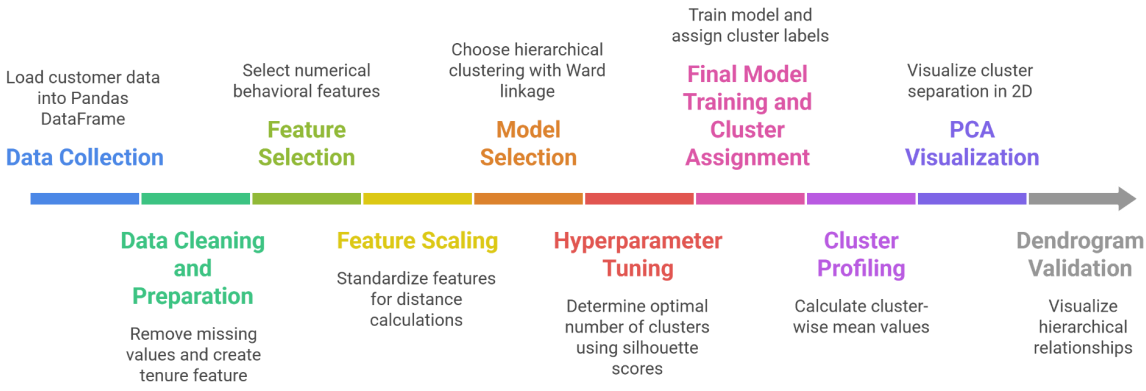
Limitations:

1. **High Computational and Memory Complexity:** Hierarchical clustering has high computational complexity, typically O(n^2 log n) for agglomerative methods and O(n^3) in the worst case, where n is the number of data points. Additionally, it requires storing a full distance matrix of size n x n, leading to high memory consumption. As a result, hierarchical clustering becomes impractical for very large datasets and is mainly suitable for small to medium-sized data.
2. **Irreversible Merging Decisions:** Once two clusters are merged in hierarchical clustering, the decision cannot be undone. Early incorrect merges caused by noise or outliers propagate throughout the clustering process and may significantly degrade the final clustering structure. Unlike K-Means, hierarchical clustering does not allow cluster reassignment or refinement after merging.
3. **Sensitivity to Noise and Outliers:** Hierarchical clustering is highly sensitive to noise and outliers, as extreme data points can strongly influence distance calculations and linkage decisions. Outliers may form singleton clusters or force incorrect merges, resulting in distorted dendrogram structures and misleading cluster interpretations.
4. **Dependence on Distance Metric and Linkage Method:** The quality of hierarchical clustering heavily depends on the choice of distance metric (e.g., Euclidean, Manhattan, cosine) and linkage criterion (single, complete, average, or Ward). Different combinations can produce

significantly different dendrograms and cluster structures, making the results less stable and harder to generalize.

Workflow:

1. **Data Collection:** The Customer Personality Analysis dataset (marketing_campaign.csv) is loaded into a Pandas DataFrame. It contains approximately 2,200 customer records including demographic attributes, spending behavior, and purchasing activity used for customer segmentation.
2. **Data Cleaning and Preparation:** Rows with missing values are removed to ensure consistency. The customer enrollment date is converted to datetime format, and a new feature, Customer_Tenure, is created to measure the duration of customer engagement.
3. **Feature Selection:** Numerical behavioral features such as income, recency, spending amounts, purchase counts, website activity, and tenure are selected. Categorical variables are excluded to maintain compatibility with distance-based clustering.
4. **Feature Scaling:** All selected features are standardized using StandardScaler so that each variable contributes equally to distance calculations used in hierarchical clustering.
5. **Model Selection (Hierarchical Clustering):** Agglomerative hierarchical clustering with Ward linkage is selected. This method merges clusters by minimizing the increase in within-cluster variance, making it suitable for numerical customer data.
6. **Hyperparameter Tuning:** The optimal number of clusters is determined by computing silhouette scores for cluster counts ranging from 2 to 10. The number of clusters with the highest silhouette score is selected as the best configuration.
7. **Final Model Training and Cluster Assignment:** The hierarchical clustering model is trained using the optimal number of clusters, and each customer is assigned a cluster label that is appended to the dataset.
8. **Cluster Profiling:** Cluster-wise mean values of selected features are calculated to interpret differences in customer behavior across segments.
9. **PCA Visualization:** Principal Component Analysis (PCA) is applied to reduce dimensionality and visualize cluster separation in a 2D scatter plot.
10. **Dendrogram Validation:** A dendrogram is generated using Ward linkage on a subset of the data to visualize hierarchical relationships and support the chosen number of clusters.



Performance Analysis:

The performance of the Hierarchical (Agglomerative) clustering model on the Customer Personality Analysis dataset is evaluated using internal validation techniques and structural visualization, since

unsupervised learning does not rely on ground-truth labels. The evaluation emphasizes cluster separation, stability, and interpretability using silhouette-based tuning, dendrogram structure, PCA visualization, and cluster profiling.

1. **Optimal Number of Clusters Selection:** Silhouette score analysis identifies K=2 as the optimal number of clusters. The dendrogram produced using Ward linkage supports this choice by showing a large vertical distance between the final merge steps. This gap indicates that combining the two major clusters would significantly increase within-cluster variance. Therefore, cutting the dendrogram at this level yields a natural and stable two-cluster structure.
2. **Cluster Separation and Visualization (PCA Analysis):** The PCA scatter plot shows clear visual separation between the two hierarchical clusters. The clusters are primarily separated along the first principal component with minimal overlap. This indicates strong inter-cluster dissimilarity and confirms that the hierarchical algorithm successfully identifies meaningful structure rather than random partitions.
3. **Dendrogram Structural Validation:** The hierarchical dendrogram illustrates nested customer relationships and confirms that the final two clusters are internally cohesive. The large linkage distance at the final merge step demonstrates strong separation, validating that the clusters are not artificially forced.

Hyperparameter Tuning:

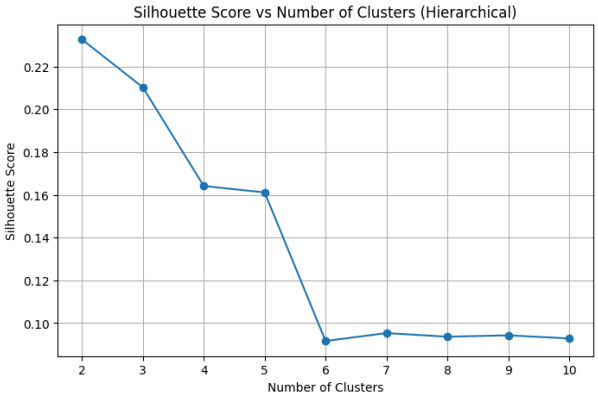
Hierarchical clustering is sensitive to hyperparameter selection, particularly the number of clusters and the linkage strategy, which determine how data points are grouped and merged. Choosing inappropriate parameters may lead to unstable or poorly separated clusters. Therefore, systematic hyperparameter tuning is necessary to ensure meaningful segmentation and strong cluster structure. Since hierarchical clustering is an unsupervised algorithm, internal validation metrics are used instead of classification-based measures. In the implemented code, hyperparameter tuning is performed by evaluating multiple cluster configurations and selecting the one that maximizes cluster separation and compactness. Tuned Hyperparameters are:

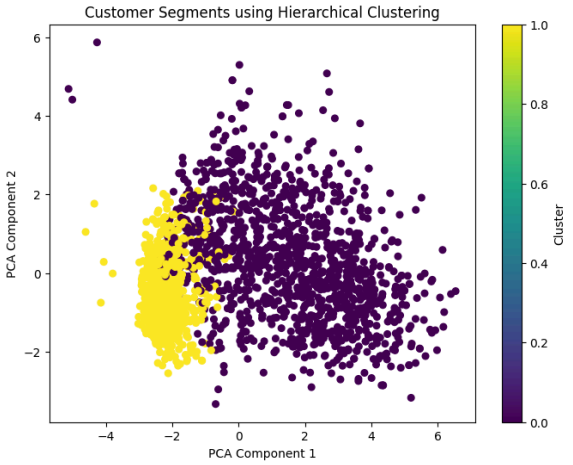
1. **Number of Clusters (K):** The number of clusters is tuned by training Agglomerative Hierarchical Clustering models for values of K ranging from 2 to 10. For each configuration, cluster assignments are evaluated using the Silhouette Score, which measures how well each data point fits within its cluster compared to neighboring clusters. The value of K that produces the highest silhouette score is selected as the optimal number of clusters, indicating strong intra-cluster similarity and inter-cluster separation.
2. **Linkage Method:** The model uses Ward linkage, which merges clusters by minimizing the increase in within-cluster variance. Ward's method is particularly suitable for numerical customer data because it produces compact, spherical clusters similar to variance-based partitioning methods. This linkage strategy improves cluster stability and interpretability.
3. **Distance Metric:** Ward linkage implicitly uses Euclidean distance, ensuring that clustering decisions are based on geometric proximity in the standardized feature space. Feature scaling ensures that all variables contribute equally to distance computation.
4. **Model Validation:** In addition to silhouette-based optimization, the dendrogram structure is used as a qualitative validation tool. A large vertical gap at the final merge stage confirms that the selected number of clusters represents a natural partition in the data rather than an arbitrary split.

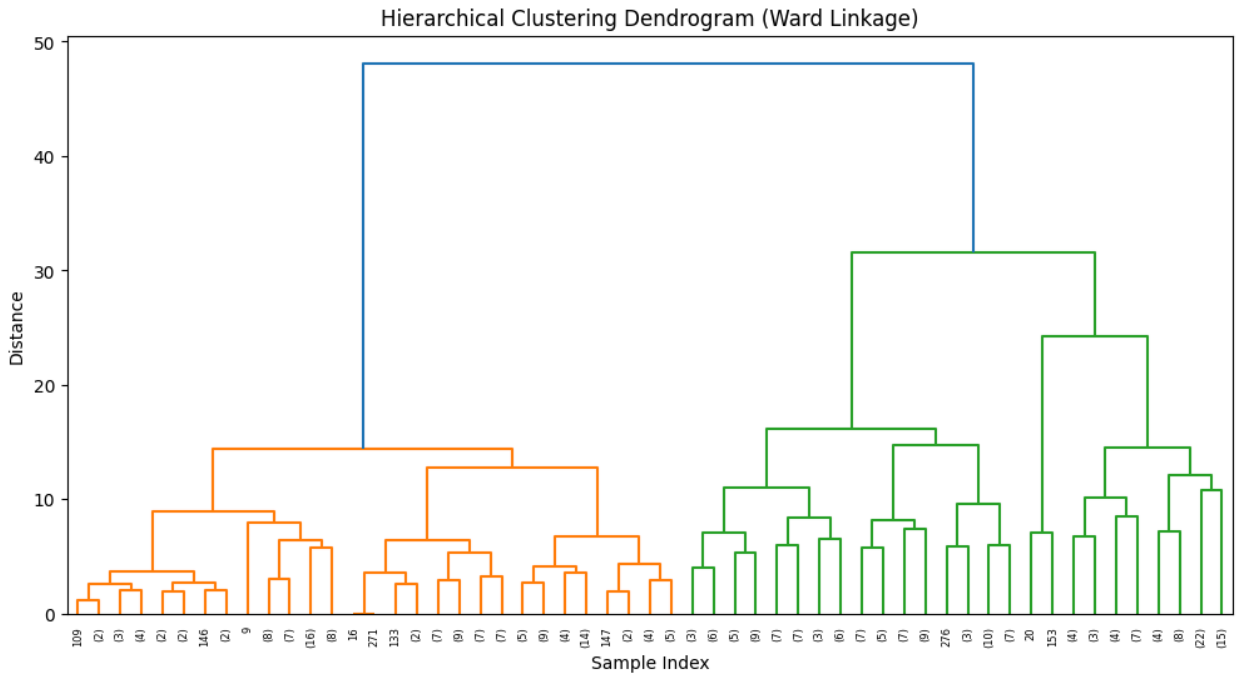
Code & Output:

```
import pandas as pd
import numpy as np

from sklearn.preprocessing import
StandardScaler
from sklearn.cluster import
AgglomerativeClustering
from sklearn.metrics import
silhouette_score

from scipy.cluster.hierarchy import
dendrogram, linkage

import matplotlib.pyplot as plt
import seaborn as sns

df =
pd.read_csv("marketing_campaign.csv"
, sep="\t")
df.head()

df = df.dropna()

df["Dt_Customer"] =
pd.to_datetime(df["Dt_Customer"],
format="%d-%m-%Y")

df["Customer_Tenure"] =
(df["Dt_Customer"].max() -
df["Dt_Customer"]).dt.days

features = [
    "Income", "Recency",
          "MntWines", "MntFruits",
"MntMeatProducts",
                "MntFishProducts",
"MntSweetProducts", "MntGoldProds",
              "NumDealsPurchases",
"NumWebPurchases",
            "NumCatalogPurchases",
"NumStorePurchases",
              "NumWebVisitsMonth",
"Customer_Tenure"
]

X = df[features]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

silhouette_scores = []
k_values = range(2, 11)

for k in k_values:
    model = AgglomerativeClustering(
        n_clusters=k,
        linkage="ward"
    )
                        labels =
model.fit_predict(X_scaled)
                         score =
silhouette_score(X_scaled, labels)
    silhouette_scores.append(score)

# Plot silhouette scores
plt.figure(figsize=(8, 5))
plt.plot(k_values,
silhouette_scores, marker="o")
plt.xlabel("Number of Clusters")
plt.ylabel("Silhouette Score")
plt.title("Silhouette Score vs
Number of Clusters (Hierarchical)")
plt.grid(True)
plt.show()

best_k =
k_values[np.argmax(silhouette_scores
)]
print("Optimal number of clusters:",
best_k)

hierarchical_model =
AgglomerativeClustering(
    n_clusters=best_k,
    linkage="ward"
)

df["Cluster"] =
hierarchical_model.fit_predict(X_sca
led)
```

```
cluster_summary = df.groupby("Cluster")[features].mean()
cluster_summary

from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=df["Cluster"], cmap="viridis", s=30)
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.title("Customer Segments using Hierarchical Clustering")
plt.colorbar(label="Cluster")
plt.show()

sample = X_scaled[:300]

linked = linkage(sample, method="ward")

plt.figure(figsize=(12, 6))
dendrogram(linked, truncate_mode="level", p=5)
plt.title("Hierarchical Clustering Dendrogram (Ward Linkage)")
plt.xlabel("Sample Index")
plt.ylabel("Distance")
plt.show()

cluster_counts = df["Cluster"].value_counts().sort_index()

plt.figure(figsize=(7, 5))
cluster_counts.plot(kind="bar")
plt.xlabel("Cluster Label")
plt.ylabel("Number of Customers")
plt.title("Cluster Size Distribution (Hierarchical)")
plt.grid(axis="y")
plt.show()

cluster_means = df.groupby("Cluster")[features].mean()

plt.figure(figsize=(12, 6))
sns.heatmap(cluster_means, cmap="viridis", annot=False)
plt.title("Heatmap of Cluster Feature Means (Hierarchical)")
plt.show()
```







Conclusion:

Hierarchical clustering identified two clear customer segments based on income and spending behavior. Silhouette tuning and dendrogram validation confirmed stable and meaningful separation. The results show that hierarchical clustering is effective for practical customer segmentation.