VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SWE2009: DATA MINING AND TECHNIQUES

A report

By

SREEVALLABH 22MIS1170

DIGITAL ASSIGNMENT 2: Implement all types of clustering using any one tool except Python/Google co-lab. One page about your implementation algorithms/dataset details and excepted output along with GitHub link.

NAME OF THE FACULTY: DR. PATTABIRAMAN V

DATASET DETAILS

The dataset is named as “Mall_customers.csv” and it has been taken from Kaggle. The dataset contains attributes like customer ID, gender, Annual Income and Spending score.

Link : https://www.kaggle.com/datasets/devahuja2808/mall-customers-hierarchical-clustering

Hierarchical clustering:

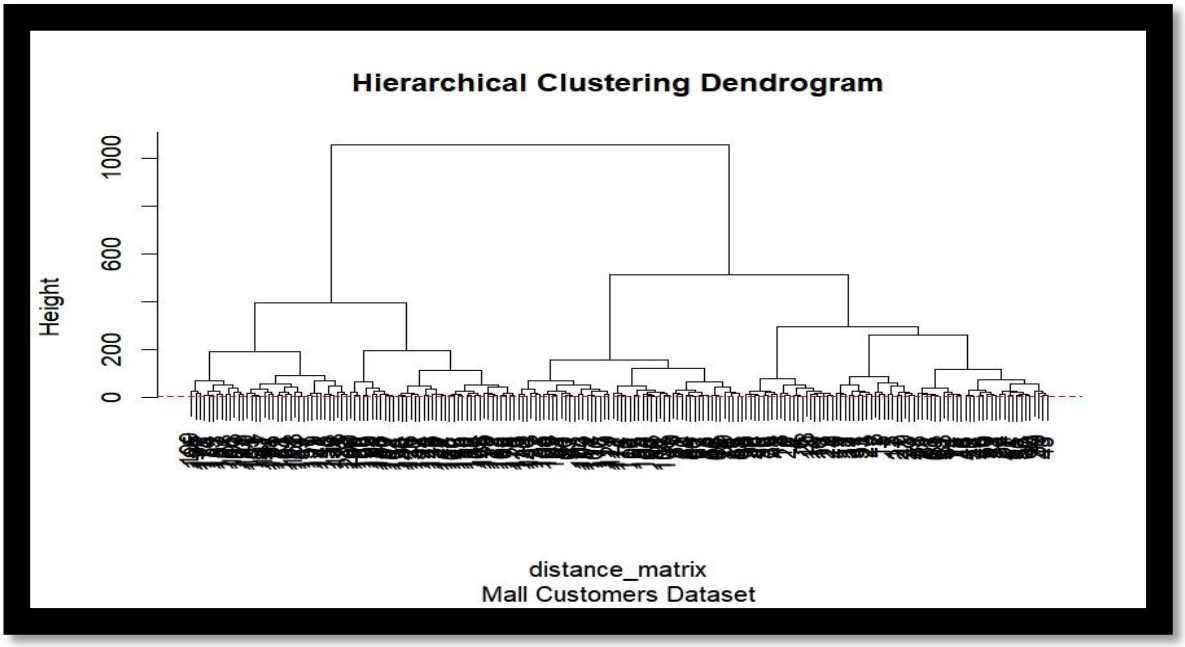
- **Implementation algorithm:**

The code implements agglomerative hierarchical clustering:

Preprocess data (optional): Handle non-numeric columns and standardize features.

Calculate distances: Create a matrix showing distances between all data points.
Cluster iteratively: Merge the closest clusters based on distances (Ward's method).
Visualize: Generate a dendrogram to see how clusters merge.
Cut dendrogram: Choose a desired number of clusters (k) based on the dendrogram.
Assign clusters: Assign each data point to a cluster based on the chosen level in the dendrogram.

Dendrogram:



K means clustering:

- ## Implementation algorithm:

1. **Preprocess data (optional):** Handle non-numeric data and potentially standardize features.
2. **Define number of clusters (k):** Choose the desired number of customer segments.
3. **Initialize:** Randomly select k centroids (cluster centers) from the data points.
4. **Assignment:** Assign each data point to the closest centroid based on distance (usually Euclidean).
5. **Update:** Recompute centroids as the mean of points assigned to each cluster.
6. **Repeat steps 4 & 5:** Until assignments stabilize (no significant changes occur).
7. **Output:** Set of clusters and their centroids, with each data point assigned to a cluster.

K medoids clustering

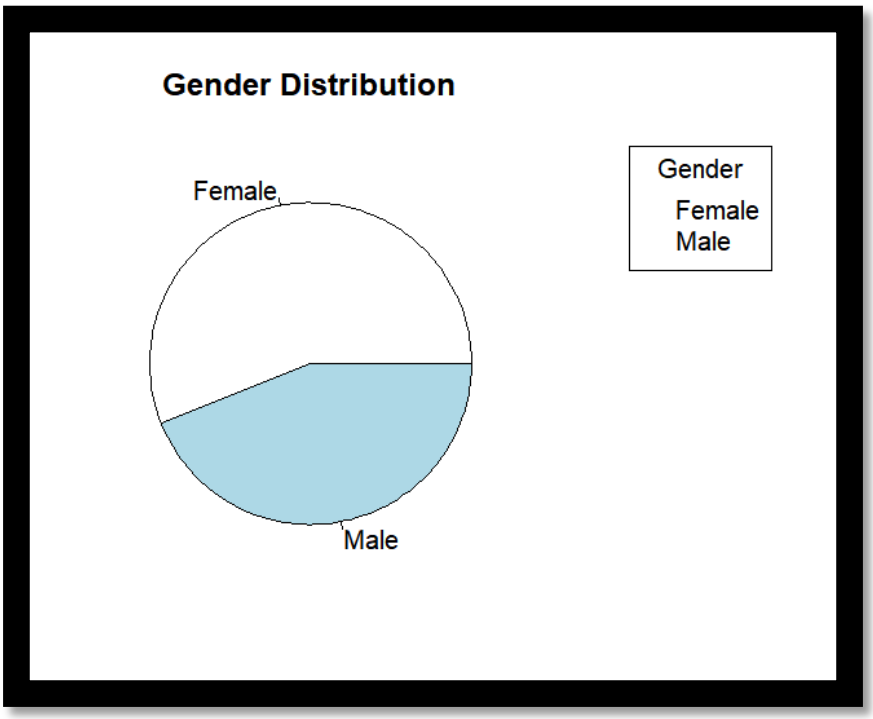
- **Implementation algorithm**

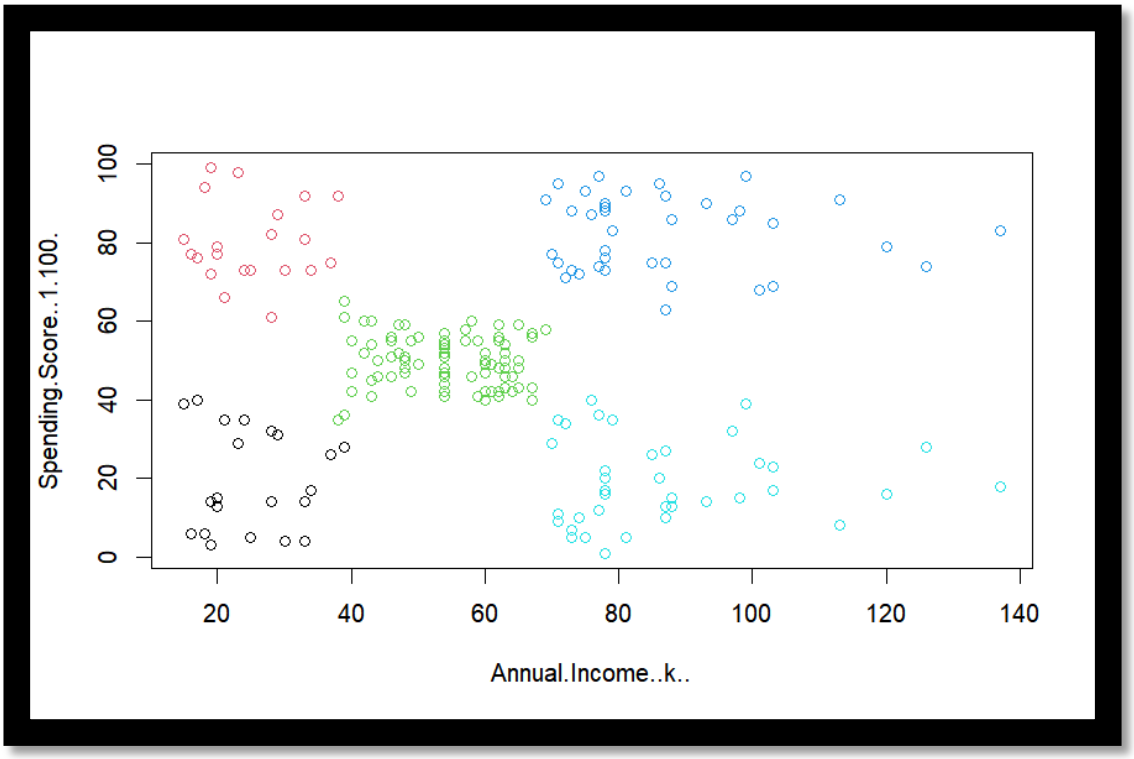
Perform K-Medoids clustering: Execute the K-Medoids code you have, which assigns cluster labels to each data point.

Access the data frame: After running the code, locate the data frame in the "Environment" or "History" pane of R Studio.

Examine the "Cluster" column: The data frame should now have a new column named "Cluster" (or whatever you named it). This column contains the cluster label (1, 2, etc.) for each data point in your dataset.

Identify cluster membership: By looking at the corresponding row in the "Cluster" column for each data point (e.g., customer A), you can determine which cluster that data point belongs to base on the assigned cluster label.





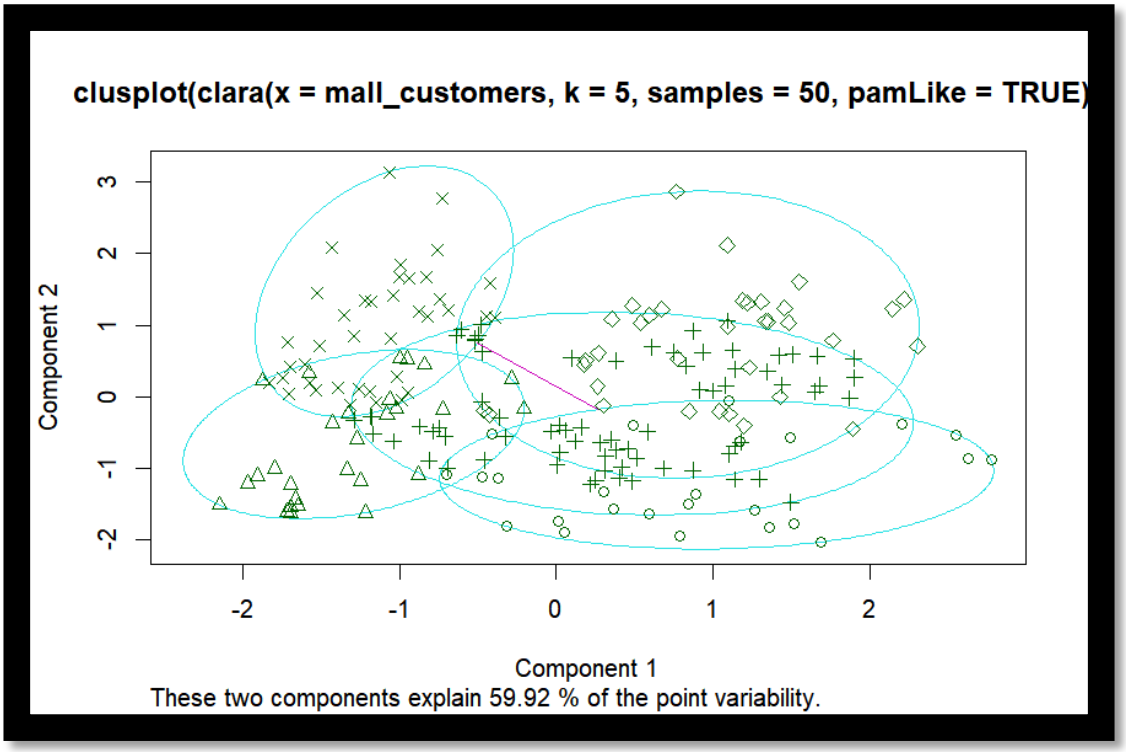
CLARA ALGORITHM:

IMPLEMENTATION ALGORITHM:

1. **Loading Necessary Package**: The code begins by loading the cluster library, which provides functions for clustering analysis in R.
2. **Setting Working Directory:** The working directory is set to the location where the data file, "Mall_Customers.csv," is located. This ensures that R knows where to find the file when it is read.
3. **Reading Data from CSV File:** The data from the CSV file is read into a data frame named df.
4. **Converting Categorical Variables to Numeric**: The "Genre" column in the data frame is converted from categorical (e.g., "Male" and "Female") to numeric format. This conversion is necessary for clustering algorithms to work with categorical data.
5. **Performing CLARA Clustering:** The CLARA clustering algorithm is applied to the subset of the data containing columns 2 through 5 (Age, Annual Income, and Spending Score), with the number of

clusters (k) set to 5. CLARA is a clustering algorithm suitable for large datasets because it samples the data and applies PAM (Partitioning Around Medoids) clustering to each sample.

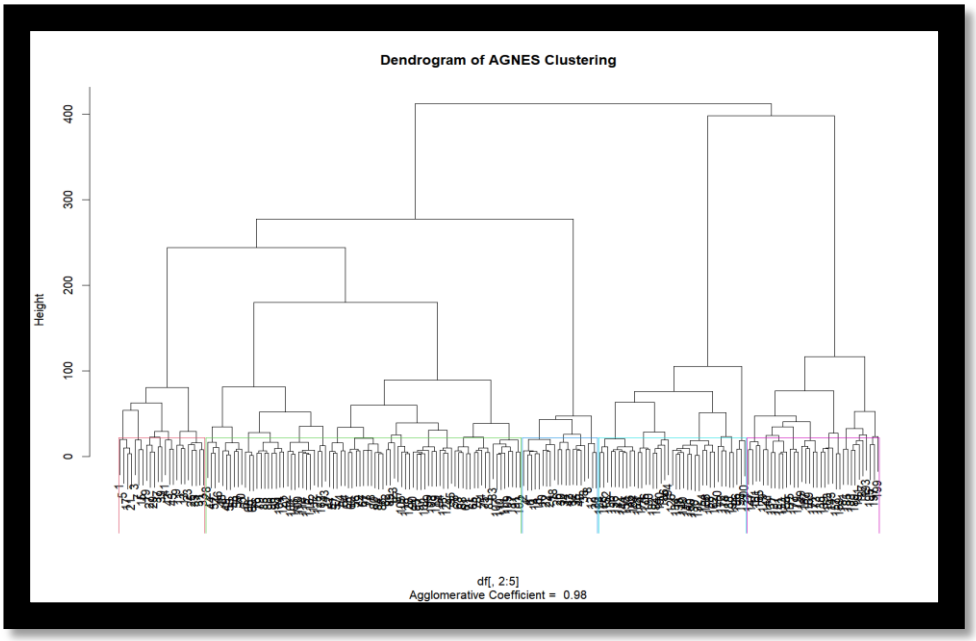
6. **Assigning Cluster Labels**: The cluster assignments obtained from CLARA clustering are added as a new column named "cluster" to the original data frame df.
7. **Printing the Data with Cluster Assignments**: The data frame df, now containing the original data along with the cluster assignments, is printed to the console.
8. **Increasing Max.Print Limit (Optional):** Optionally, the max.print option is increased to a large value (10,000 in this case) to ensure that all rows of the data frame are printed to the console, even if it exceeds the default maximum print limit.
9. **Printing the Data Again:** Finally, the data frame df is printed again to the console, now with the increased maximum print limit to display all rows.



AGNES CLUSTERING (AGglomerative NESting):

IMPLEMENTATION ALGORITHM:

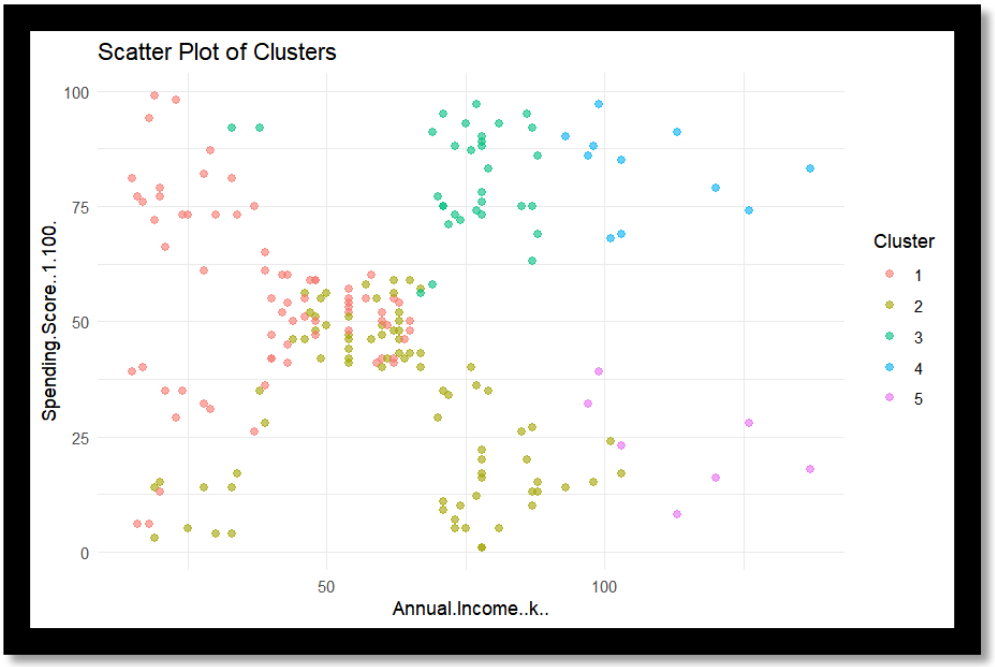
1. **Load Necessary Package**: The code begins by loading the **cluster** library, which provides functions for clustering analysis in R.
2. **Set the Working Directory**: The working directory is set to the location where the data file, "Mall_Customers.csv," is located. This ensures that R knows where to find the file when it is read.
3. **Read Data from CSV File**: The data from the CSV file is read into a data frame named **df**.
4. **Convert Categorical Variables to Numeric**: The "Genre" column in the data frame is converted from categorical (e.g., "Male" and "Female") to numeric format. This conversion is necessary for clustering algorithms to work with categorical data.
5. **Perform AGNES Clustering**: The AGNES clustering algorithm is applied to the subset of the data containing columns 2 through 5 (Age, Annual Income, and Spending Score). The **method** parameter is set to "ward", which specifies the linkage method used in hierarchical clustering.
6. **Plot the Dendrogram**: The resulting dendrogram from AGNES clustering is plotted using the **plot()** function. The main title of the plot is set to "Dendrogram of AGNES Clustering", and the size of the labels is adjusted using the **cex** parameter.



DIANA CLUSTERING (DIvisive ANAlysis):

IMPLEMENTATION ALGORITHM:

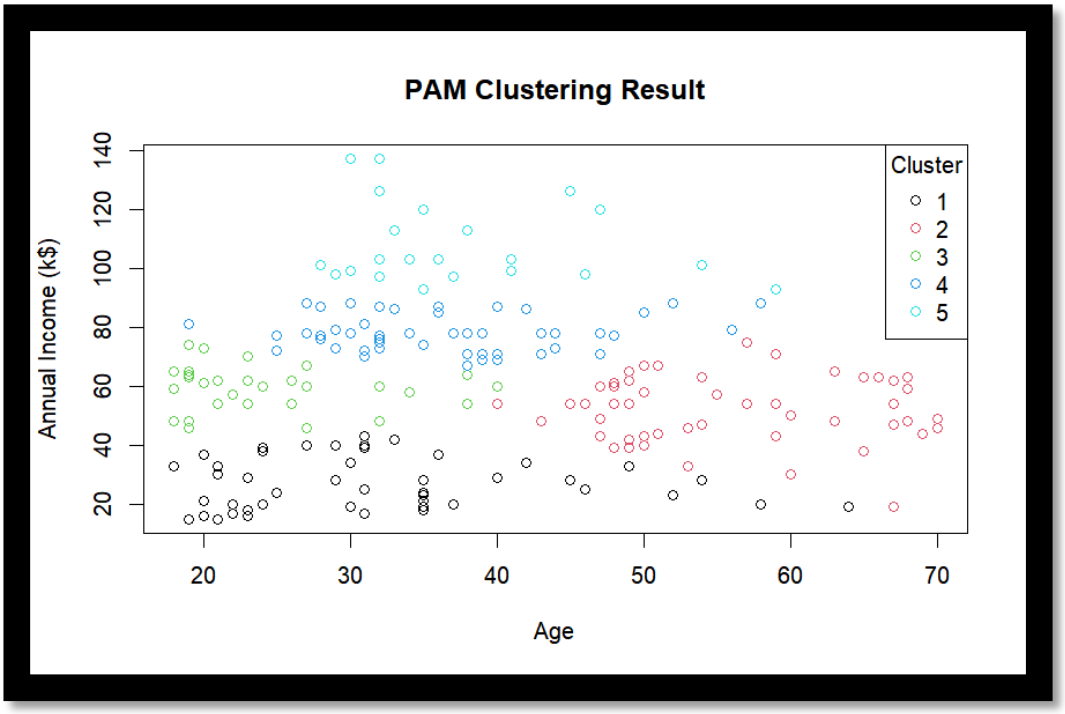
1. **Load Necessary Package:** The code begins by loading the cluster library, which provides functions for clustering analysis in R.
2. **Set the Working Directory:** The working directory is set to the location where the data file, "Mall_Customers.csv," is located. This ensures that R knows where to find the file when it is read.
3. **Read Data from CSV File:** The data from the CSV file is read into a data frame named df**.**
4. **Convert Categorical Variables to Numeric:** The "Genre" column in the data frame is converted from categorical (e.g., "Male" and "Female") to numeric format. This conversion is necessary for clustering algorithms to work with categorical data.
5. **Perform DIANA Clustering:** The DIANA clustering algorithm is applied to the subset of the data containing columns 2 through 5 (Age, Annual Income, and Spending Score). Unlike AGNES, DIANA does not require specifying a method for linkage because it is a divisive hierarchical clustering algorithm.
6. **Plot the Dendrogram:** The resulting dendrogram from DIANA clustering is plotted using the plot() function. The main title of the plot is set to "Dendrogram of DIANA Clustering", and the size of the labels is adjusted using the cex parameter.



PAM ALGORITHM (Partitioning Around Medoids):

IMPLEMENTATION ALGORITHM:

1. **Initialization:**
   - Select k data points from the dataset as initial medoids.
   - Assign each data point to the nearest medoid to form initial clusters.
2. **Iteration:**
   - Repeat until convergence:
     - For each cluster:
       - For each non-medoid data point in the cluster:
         - Swap the non-medoid with the medoid.
         - Calculate the total cost (e.g., sum of distances) of the cluster with the new medoid.
         - If the total cost decreases, keep the new medoid; otherwise, revert the swap.
3. **Termination:**
   - When no more swaps result in a decrease in total cost, the algorithm converges, and the final clusters are formed.



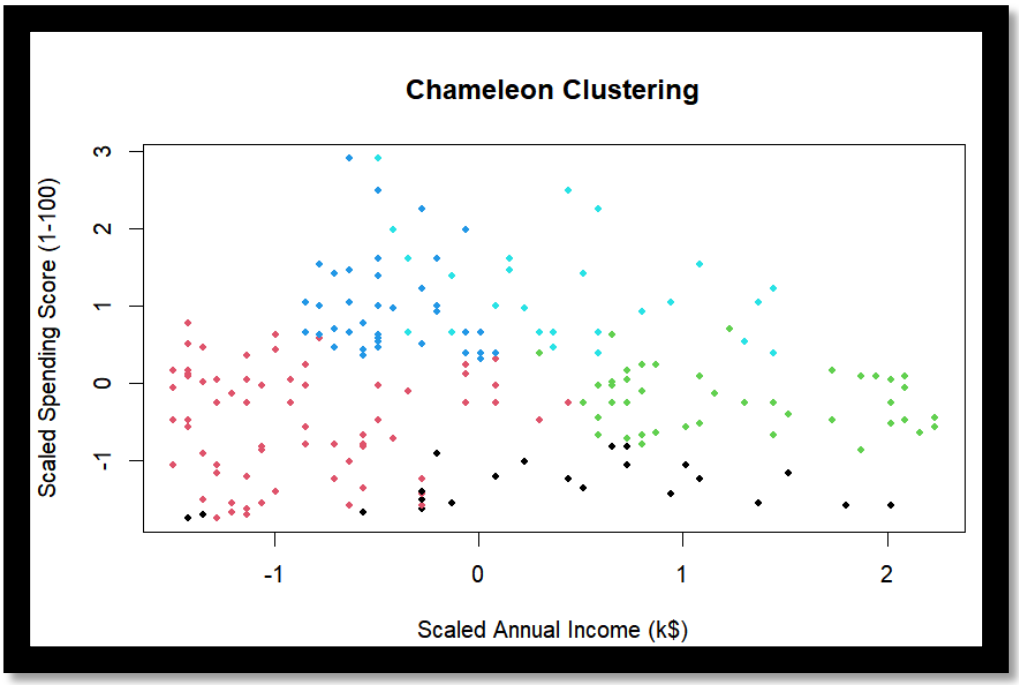
CHAMELEON CLUSTERING:

IMPLEMENTATION ALGORITHM:

1. **Input**:
   - **Dataset**: Input dataset with data points.
   - **k_nearest_neighbors**: Parameter to specify the number of nearest neighbors for the algorithm.
   - **threshold**: Threshold parameter for merging clusters.
2. **Initialization**:
   - Assign each data point to its own initial cluster.
3. **Compute Similarity Matrix**:
   - Compute the similarity matrix between all pairs of data points using a similarity measure like Euclidean distance or cosine similarity.
4. **Compute k-Nearest Neighbors**:
   - For each data point, find its k-nearest neighbors based on the similarity matrix.
5. **Merge Clusters**:
   - For each data point, compare its cluster with the clusters of its k-nearest neighbors.

- If the majority of the k-nearest neighbors belong to a different cluster and the similarity between the data point and the other cluster is greater than a threshold, merge the two clusters.

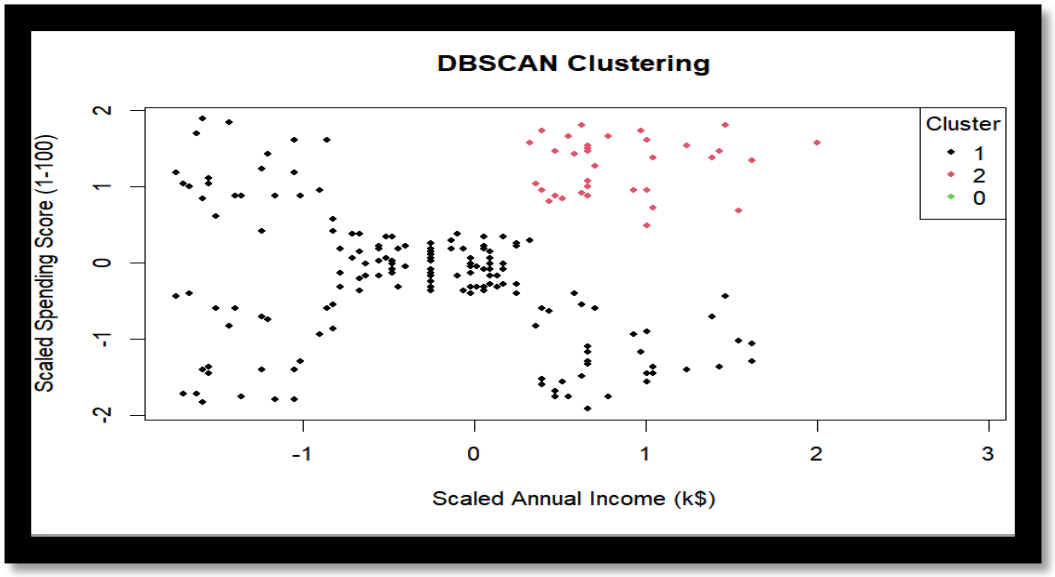
6. **Update Cluster Membership**:
   - Update the cluster memberships of data points based on the merged clusters.
7. **Repeat**:
   - Repeat steps 3-6 until convergence (i.e., no more merges occur or until a maximum number of iterations is reached).



Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

IMPLEMENTATION ALGORITHM:

1. **Input**:
    - **Dataset**: Input dataset with data points.
    - **Epsilon (ε)**: Parameter to define the radius of neighborhood around a data point.
    - **MinPts**: Minimum number of data points required to form a dense region (core point).
2. **Initialization**:
    - Assign each data point as unvisited and unclassified.
3. **Core Point Identification**:
    - For each data point, compute the number of neighboring points within a distance ε (including itself).
    - If the number of neighbors is greater than or equal to MinPts, mark the data point as a core point.
4. **Density-Reachability**:
    - For each core point:
        - Recursively expand the cluster by visiting all directly density-reachable points (data points within ε distance) from the core point.
        - Assign them to the same cluster as the core point if they are not already assigned to a cluster.
        - Mark them as visited.
5. **Noise Identification**:
    - Any data point that is not a core point and is not density-reachable from any core point is considered noise and remains unclassified.



ROCK CLUSTERING:

IMPLEMENTATION ALGORITHM

1. **Input**:
   - **Dataset**: Input dataset with data points.
   - **k**: Number of clusters to be formed.
   - **δ (delta)**: Distance threshold for forming links between data points.
2. **Initialization**:
   - Assign each data point as unvisited and unclassified.
3. **Link Formation**:
   - For each data point, find all other data points within a distance of δ.
   - Form links between data points that are mutual nearest neighbors within their δ-neighborhoods.
4. **Cluster Formation**:
   - Identify connected components in the graph formed by the links.
   - Each connected component represents a potential cluster.
5. **Noise Identification**:
   - Identify outliers as data points not included in any cluster.
6. **Merge Clusters** (Optional):

- Evaluate the similarity between clusters based on a similarity metric (e.g., Jaccard similarity coefficient).
- Merge clusters that have a high similarity above a certain threshold.

7. **Output**:
   - Final clustering result with clusters and noise points.