B M S COLLEGE OF ENGINEERING

(An Autonomous Institution Affiliated to VTU, Belagavi)

Post Box No.: 1908, Bull Temple Road, Bengaluru – 560 019

DEPARTMENT OF MACHINE LEARNING

Academic Year: 2023-2024 (Session: Nov 2023 - Feb 2024)

KNOWLEDGE DISCOVERY (23AM5PEKDI)

ALTERNATIVE ASSESSMENT TOOL (AAT)

AGGLOMERATIVE CLUSTERING

Submitted by

Student Name:	**Chetna Mundra**	**Aryaman Sharma**
USN:	**1BM21AI036**	**1BM21AI027**
Date:	**03-02-2024**	
Semester & Section:	**V-A**	
Total Pages:	**18**	
Student Signature:		

Valuation Report **(to be filled by the faculty)**

Score:	
Comments:	
Faculty In-charge:	**Prof. Pallavi B**
Faculty Signature: with date	

INDEX

1. INTRODUCTION

Agglomerative clustering is an unsupervised machine-learning algorithm that groups data into clusters. It is a bottom-up approach hierarchical clustering approach, where each data point is assigned to its own cluster and then clusters are merged as the algorithm progresses.

Each cluster in agglomerative clustering is formed using the distance between the data points, which can be calculated using various techniques such as Euclidean distance or Manhattan distance. The algorithm starts by assigning each data point to its own cluster. Then, the algorithm looks for the two closest clusters and joins them into a single cluster. This process is repeated until all points are grouped into one cluster or until the desired number of clusters is achieved.

Agglomerative clustering is a useful approach that can help identify patterns in a dataset and group data points into meaningful clusters. It is often used for exploratory data analysis, customer segmentation, and feature extraction.

The distance between clusters in agglomerative clustering can be calculated using three approaches namely single linkage, complete linkage, and average linkage.

- In the single linkage approach, the distance between the nearest points in two clusters as the distance between the clusters is taken.
- In the complete linkage approach, the distance between the farthest points in two clusters as the distance between the clusters is taken.
- In the average linkage approach, the average distance between each pair of points in two given clusters as the distance between the clusters is taken. The distance between the centroids of the clusters as their distance from each other is also taken.

2. METHODOLOGY

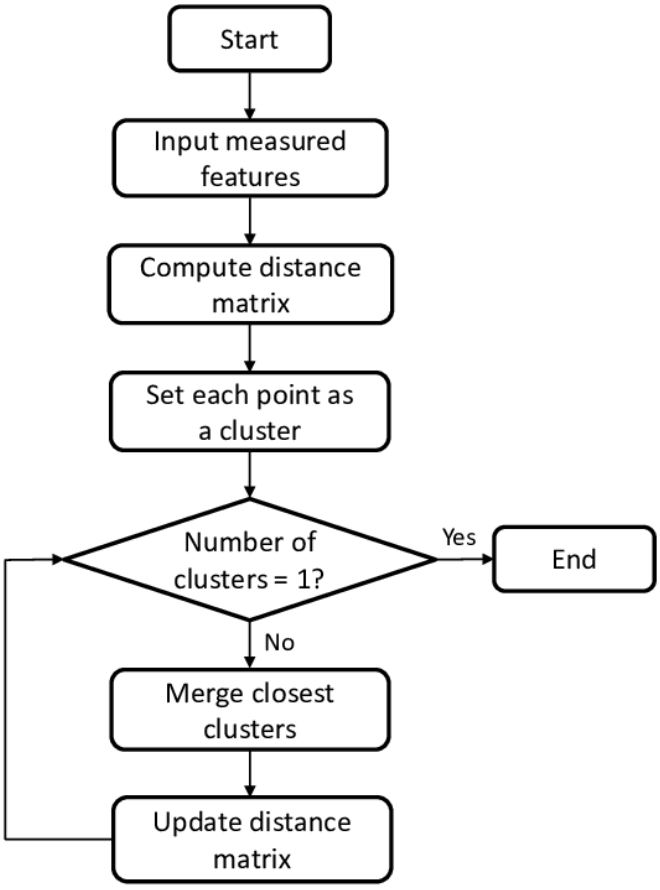
2.1 PROCEDURE :

1. Initialize clusters:
    - Start with each data point as a singleton cluster.
    - Define a distance metric (e.g., Euclidean distance) to measure the dissimilarity between clusters.
2. Compute pairwise distances:
    - Calculate the distance matrix between all pairs of clusters using the chosen distance metric.
    - The distance between two clusters can be computed using various linkage criteria such as single linkage, complete linkage, or average linkage.
3. Find the closest clusters:
    - Identify the pair of clusters with the smallest distance in the distance matrix.
    - These clusters will be merged in the next step.
4. Merge closest clusters:
    - Combine the two closest clusters into a single cluster.
    - Update the distance matrix to reflect the new distances between the merged cluster and the remaining clusters.
5. Repeat steps 3-4:
    - Repeat steps 3 and 4 until only a specified number of clusters remain or all data points belong to a single cluster.

2.2 ALGORITHM :

```
function agglomerative_clustering(Data, Linkage, Num_clusters):
    // Initialization
    clusters = [{point} for point in Data]

    // Repeat until the desired number of clusters is reached
    while len(clusters) > Num_clusters:
        // Compute pairwise distances
        distance_matrix = compute_distance_matrix(clusters, Linkage)

        // Find the closest clusters
        cluster_i, cluster_j = find_closest_clusters(distance_matrix)

        // Merge closest clusters
        merged_cluster = merge_clusters(clusters[cluster_i], clusters[cluster_j])
        clusters.pop(max(cluster_i, cluster_j))
        clusters.pop(min(cluster_i, cluster_j))
        clusters.append(merged_cluster)

    // Output
    return clusters

function compute_distance_matrix(clusters, Linkage):
    // Calculate pairwise distances between clusters based on chosen linkage
    // Return a distance matrix

function find_closest_clusters(distance_matrix):
    // Find the indices of the closest clusters in the distance matrix
    // Return the indices (i, j) of the closest clusters

function merge_clusters(cluster_i, cluster_j):
    // Combine two clusters into a single cluster
    // Return the merged cluster
```

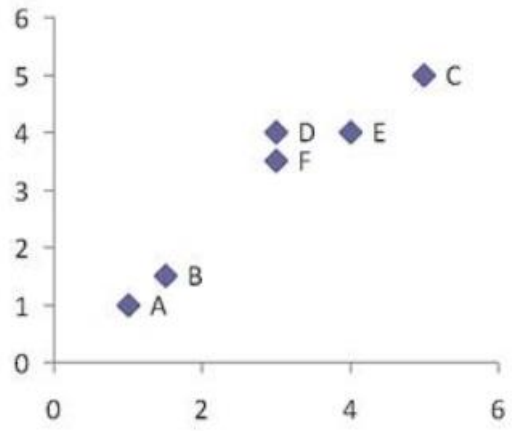
2.3 FLOWCHART :



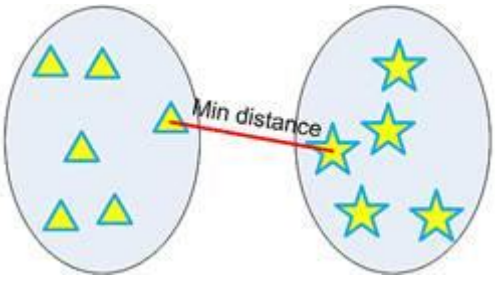
3. MATHEMATICAL IMPLEMENTATION

To illustrate hierarchical clustering algorithm, let us use the following simple example. Suppose we have 6 objects (with name A, B, C, D, E and F) and each object have two measured feature X1 and X2. We can plot the features in a scattered plot to get the visualisation of proximity between objects.

	X1	X2
A	1	1
B	1.5	1.5
C	5	5
D	3	4
E	4	4
F	3	3.5



Minimum distance clustering is also called as single linkage hierarchical clustering or nearest neighbor clustering. Distance between two clusters is defined by the minimum distance between objects of the two clusters, as shown below:



For example, we have given an input distance matrix of size 6 by 6. This distance matrix was calculated based on the object features as explained in the previous section.

Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

We have 6 objects and we put each object into one cluster. Thus, in the beginning we have 6 clusters. Our goal is to group those 6 clusters such that at the end of the iterations, we will have only single cluster consists of the whole six original objects.

In each step of the iteration, we find the closest pair clusters. In this case, the closest cluster is between cluster F and D with shortest distance of 0.5. Thus, we group cluster D and F into cluster (D, F). Then we update the distance matrix (see distance matrix below). Distance between ungrouped clusters will not change from the original distance matrix. Now the problem is how to calculate distance between newly grouped clusters (D, F) and other clusters?

Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	?	4.24
B	0.71	0.00	4.95	?	3.54
C	5.66	4.95	0.00	?	1.41
D, F	?	?	?	0.00	?
E	4.24	3.54	1.41	?	0.00

That is exactly where the linkage rule comes into effect. Using single linkage, we specify minimum distance between original objects of the two clusters.

Using the input distance matrix, distance between cluster (D, F) and cluster A is computed as

$$d_{(D,F)\to A} = \min\left(d_{DA}, d_{FA}\right) = \min\left(3.61, 3.20\right) = 3.20$$

Distance between cluster (D, F) and cluster B is

$$d_{(D,F)\to B} = \min\left(d_{DB}, d_{FB}\right) = \min\left(2.92, 2.50\right) = 2.50$$

Similarly, distance between cluster (D, F) and cluster C is

$$d_{(D,F)\to C} = \min\left(d_{DC}, d_{FC}\right) = \min\left(2.24, 2.50\right) = 2.24$$

Finally, distance between cluster E and cluster (D, F) is calculated as

$$d_{E\to(D,F)} = \min\left(d_{ED}, d_{EF}\right) = \min\left(1.00, 1.12\right) = 1.00$$

Then, the updated distance matrix becomes

Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	3.20	2.50	2.24	0.00	1.00
E	4.24	3.54	1.41	1.00	0.00

Looking at the lower triangular updated distance matrix, we found out that the closest distance between cluster B and cluster A is now 0.71. Thus, we group cluster A and cluster B into a single cluster name (A, B).

Now we update the distance matrix. Aside from the first row and first column, all the other elements of the new distance matrix are not changed.

Dist	A,B	C	(D, F)	E
A,B	0	?	?	?
C	?	0	2.24	1.41
(D, F)	?	2.24	0	1.00
E	?	1.41	1.00	0

Using the input distance matrix (size 6 by 6), distance between cluster C and cluster (D, F) is computed as

$$d_{C\to(A,B)} = \min\left(d_{CA}, d_{CB}\right) = \min\left(5.66, 4.95\right) = 4.95$$

Distance between cluster (D, F) and cluster (A, B) is the minimum distance between all objects involves in the two clusters

$$d_{(D,F)\to(A,B)} = \min\left(d_{DA}, d_{DB}, d_{FA}, d_{FB}\right) = \min\left(3.61, 2.92, 3.20, 2.50\right) = 2.50$$

Similarly, distance between cluster E and (A, B) is

$$d_{E\to(A,B)} = \min\left(d_{EA}, d_{EB}\right) = \min\left(4.24, 3.54\right) = 3.54$$

Then the updated distance matrix is

Min Distance (Single Linkage)

Dist	A,B	C	(D, F)	E
A,B	0	4.95	2.50	3.54
C	4.95	0	2.24	1.41
(D, F)	2.50	2.24	0	1.00
E	3.54	1.41	1.00	0

Observing the lower triangular of the updated distance matrix, we can see that the closest distance between clusters happens between cluster E and (D, F) at distance 1.00. Thus, we cluster them together into cluster ((D, F), E).

The updated distance matrix is given below.

Min Distance (Single Linkage)

Dist	(A,B)	C	(D, F), E
(A,B)	0.00	4.95	2.50
C	4.95	0.00	1.41
(D, F), E	**2.50**	**1.41**	0.00

Distance between cluster ((D, F), E) and cluster (A, B) is calculated as

$$d_{((D,F),E)\mapsto(A,B)} = \min\left(d_{DA}, d_{DB}, d_{FA}, d_{FB}, d_{EA}, d_{EB}\right) = \min\left(3.61, 2.92, 3.20, 2.50, 4.24, 3.54\right) = 2.50$$

Distance between cluster ((D, F), E) and cluster C yields the minimum distance of 1.41. This distance is computed as

$$d_{((D,F),E)\mapsto C} = \min\left(d_{DC}, d_{FC}, d_{EC}\right) = \min\left(2.24, 2.50, 1.41\right) = 1.41$$

After that, we merge cluster ((D, F), E) and cluster C into a new cluster name (((D, F), E), C).

The updated distance matrix is shown in the figure below

Min Distance (Single Linkage)

Dist	(A,B)	(D, F), E),C
(A,B)	0.00	2.50
((D, F), E),C	**2.50**	0.00

The minimum distance of 2.5 is the result of the following computation

$$d_{(((D,F),E),C)\mapsto(A,B)} = \min\left(d_{DA}, d_{DB}, d_{FA}, d_{FB}, d_{EA}, d_{EB}, d_{CA}, d_{CB}\right)$$

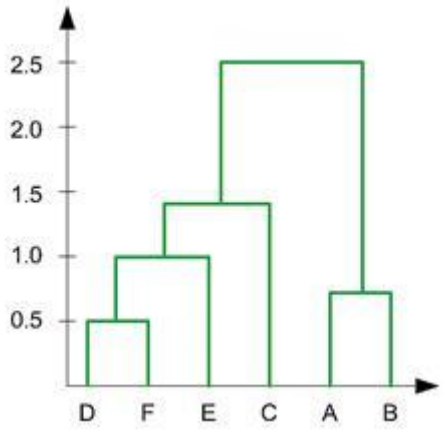
$$d_{(((D,F),E),C)\mapsto(A,B)} = \min\left(3.61, 2.92, 3.20, 2.50, 4.24, 3.54, 5.66, 4.95\right) = 2.50$$

Now if we merge the remaining two clusters, we will get only single cluster contain the whole 6 objects. Thus, our computation is finished. We summarized the results of computation as follow:

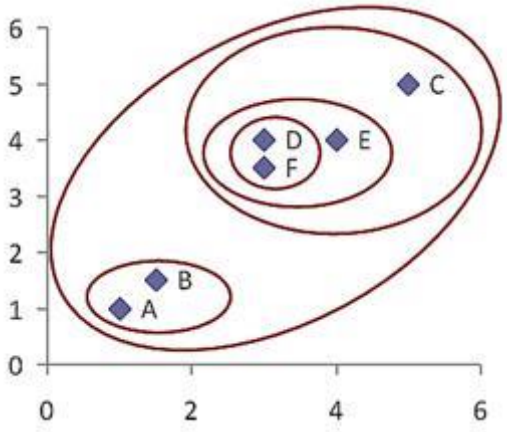
1. In the beginning we have 6 clusters: A, B, C, D, E and F
2. We merge cluster D and F into cluster (D, F) at distance 0.50

3. We merge cluster A and cluster B into (A, B) at distance 0.71
4. We merge cluster E and (D, F) into ((D, F), E) at distance 1.00
5. We merge cluster ((D, F), E) and C into (((D, F), E), C) at distance 1.41
6. We merge cluster (((D, F), E), C) and (A, B) into ((((D, F), E), C), (A, B)) at distance 2.50
7. The last cluster contain all the objects, thus conclude the computation

Using this information, we can now draw the final results of a dendogram. The dendogram is drawn based on the distances to merge the clusters above.



The hierarchy is given as (((D, F), E),C), (A,B). We can also plot the clustering hierarchy into XY space



4. IMPLEMENTATION IN PYTHON

4.1 ABOUT DATASET :

Overview:

The dataset under consideration pertains to customer information in a mall, encompassing key attributes such as `CustomerID`, `Gender`, `Age`, `Annual Income (k$)`, and `Spending Score (1-100)`.

Data Statistics:

- Gender Distribution: The gender distribution in the dataset reveals that 56% of customers are female, while 44% are male.
- Age Distribution: The age of the customers spans from 18 to 70, with various ranges capturing the diversity of age groups.
- Annual Income Distribution: Customers exhibit diverse annual incomes, covering a spectrum of income ranges.
- Spending Score Distribution: Spending scores are distributed across different ranges, showcasing a variety of spending behaviors.

Clustering Labels:

The dataset has been categorized into clusters based on spending score ranges, each labeled with a specific identifier and accompanied by the count of customers within that cluster.

Insights:

The dataset provides insights into the diverse demographics of customers, including age, gender, annual income, and spending behaviors.

4.2 FUNCTIONS & TOOLS USED :

The following are the Functions and Tools Used in Clustering Analysis :

1. **Data Handling and Exploration:**
    - **NumPy and Pandas**: These powerful libraries were employed to facilitate efficient data manipulation and exploration. NumPy provided support for numerical operations, while Pandas streamlined the handling of tabular data.
    - Dataset Loading: The dataset, residing in a CSV file, was seamlessly loaded into a Pandas DataFrame. This step laid the groundwork for subsequent analysis.
2. **Dendrogram Visualization:**
    - **Scipy's Hierarchical Clustering:** Leveraging Scipy's hierarchical clustering functionality, the generation of a dendrogram was facilitated. This visualization illuminated the hierarchical relationships among data points, aiding in cluster determination.
    - **Matplotlib for Visualization:** Matplotlib, a versatile plotting library, was instrumental in creating the dendrogram plot. The visual representation helped in identifying an optimal number of clusters.
3. **Agglomerative Clustering:**
    - **Scikit-Learn's AgglomerativeClustering:** The scikit-learn library provided a robust implementation of agglomerative clustering. The `AgglomerativeClustering` class was configured with parameters for the desired number of clusters, affinity metric, and linkage criterion.
4. **Cluster Visualization:**
    - **Matplotlib for Scatter Plotting:** Matplotlib was once again employed to visualize the clusters formed by the agglomerative clustering algorithm. Different clusters were distinguished using color-coded scatter plots, enhancing the interpretability of the results.

These essential tools and functions played pivotal roles at various stages of the agglomerative clustering analysis, contributing to data preparation, exploration, clustering, and visualization.

4.3 RESULTS & ANALYSIS :

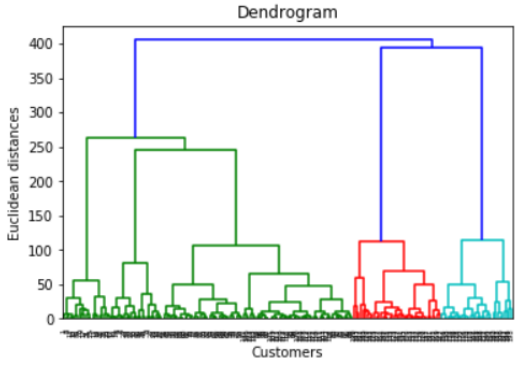
Sure, here are the results and analysis of the agglomerative clustering you sent me:

- There are four clusters of customers. These clusters are based on the customers' annual income and spending score.
- The cluster with the highest annual income is also the cluster with the highest spending score. This cluster is labeled "Careful Target".
- The cluster with the lowest annual income is also the cluster with the lowest spending score. This cluster is labeled "Miser".
- The other two clusters have intermediate annual incomes and spending scores. These clusters are labeled "Careless" and "General".

Here are some additional insights that can be drawn from the clustering:

- The "Careful Target" cluster is the most profitable for the business. Customers in this cluster are likely to make high-value purchases.
- The "Miser" cluster is the least profitable for the business. Customers in this cluster are likely to make low-value purchases.
- The "Careless" and "General" clusters have the potential to be more profitable for the business. However, the business will need to develop targeted marketing campaigns to reach these customers.

It is important to note that these are just general insights, and the specific results of the clustering will vary depending on the data that is used. However, agglomerative clustering can be a useful tool for understanding customer behavior and segmenting customers into groups for targeted marketing campaigns.



5. APPLICATIONS

Agglomerative clustering, a hierarchical clustering technique, finds applications in various fields due to its ability to reveal hierarchical structures and natural groupings within data. Some notable applications include:

1. **Biology and Bioinformatics:**
   - Identifying genetic similarities among species.
   - Clustering gene expression data to identify co-regulated genes.
2. **Marketing and Customer Segmentation:**
   - Segmentation of customers based on purchasing behavior.
   - Grouping customers for targeted marketing strategies.
3. **Image and Object Recognition:**
   - Clustering pixels in images to identify regions or objects.
   - Grouping similar images or patterns in computer vision.
4. **Social Network Analysis:**
   - Identifying communities or groups within social networks.
   - Analyzing user behavior to improve content recommendations.
5. **Document Clustering and Text Mining:**
   - Grouping similar documents for topic modeling.
   - Clustering words or phrases to identify patterns in text data.
6. **Finance and Risk Management:**
   - Identifying patterns in financial time series data.
   - Analyzing credit risk by grouping similar financial profiles.
7. **Medical Imaging and Diagnostics:**
   - Grouping similar medical images for disease classification.
   - Clustering patient data for personalized medicine.
8. **Speech and Audio Processing:**
   - Clustering similar audio segments for music recommendation.
   - Grouping phonemes or speech features in linguistics studies.

Agglomerative clustering's versatility makes it a valuable tool across disciplines, offering insights into complex structures within diverse datasets. Its hierarchical nature allows for nuanced exploration and interpretation of relationships among data points.

6. ADVANTAGES & DISADVANTAGES

6.1 ADVANTAGES

Agglomerative clustering has many advantages. Some of them are listed below.

- Agglomerative clustering is simple to implement and easy to interpret. You can implement it very easily in programming languages like python.
- It is a bottom-up approach that produces a hierarchical structure of clusters. So, you can choose any level of hierarchy to select a suitable number of clusters. Hence, it allows the clusters to build up from individual elements to the desired level of granularity.
- Agglomerative clustering is robust to noise and outliers. As the clusters are derived by combining the nearest clusters into one, the outliers will always stay in a different cluster and will not affect other clusters.
- The algorithm does not require any assumptions about the structure of the data or the number of clusters. It can be used with different types of data such as numerical, categorical, and binary. You can take any dataset and define a distance metric between the data points to calculate the distance matrix. After that, you can use the distance matrix to perform clustering as usual.
- It can be used for datasets of any size. Agglomerative clustering is a scalable algorithm. You can use it for 10 data points as well as 10 million data points. It is computationally efficient and scalable in both cases.
- Agglomerative clustering can be used with a variety of distance or similarity metrics. As the clustering algorithm uses the distance matrix for calculating clusters at each step instead of the actual data points, you can define ant distance or similarity metric to calculate the distance matrix.
- Agglomerative clustering is capable of producing clusters with non-convex shapes. It can identify clusters of different shapes and sizes.
- It is well-suited for exploratory data analysis and is well-suited for finding natural clusters in a dataset.

6.2 DISADVANTAGES

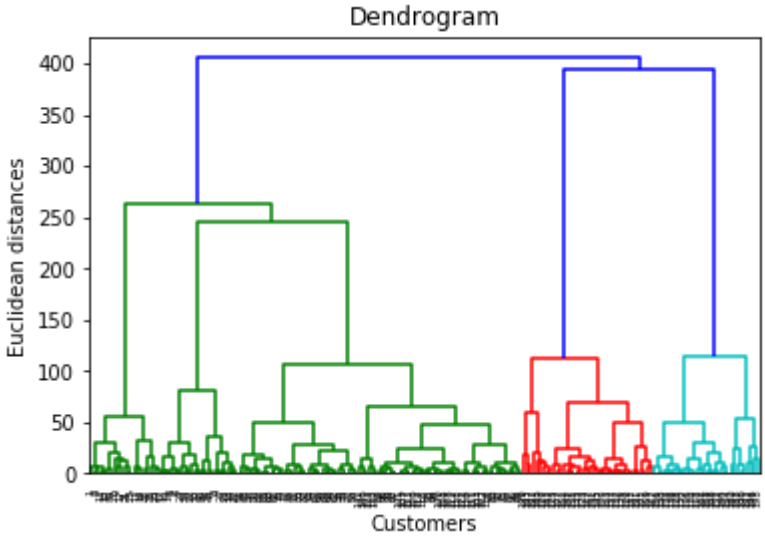
- It is difficult to determine the right number of clusters in agglomerative clustering. While the clustering process produces clusters in a hierarchical manner, you might find it difficult to choose the granularity level to select the right number of clusters.
- It can be difficult to interpret the results of clustering. If you don't have an understanding of the dataset and domain knowledge, you won't be able to what features the clusters at each level have.
- Agglomerative clustering may not produce the same results each time. As shown in the numerical example discussed above, different distance measures produce different results in agglomerative clustering. Even the approach using the same distance measure can create different clusters in different executions.
- Agglomerative clustering is not very suitable for high-dimensional data and data with non-convex shapes.
- Agglomerative clustering does not consider the intrinsic properties of the data. It just uses the distance between the data points to group them into clusters.
- Agglomerative clustering is not suitable for classifying data with overlapping clusters. It doesn't take into account the relationship between the clusters.
- Agglomerative clustering isn't suitable for data with different densities, variances, and shapes. Sometimes, it might also be difficult to decide on the right linkage criteria for clustering.
- Agglomerative clustering is a greedy algorithm. It always takes the nearest clusters to combine them into bigger clusters. Hence, it might not always produce the best results with optimal clusters.
- There is no guarantee that the resulting clusters in agglomerative clustering are meaningful. The algorithm does not provide any probabilistic guarantees about the quality of the clusters.

7. CONCLUSION

In conclusion, the utilization of agglomerative clustering on the mall customer dataset has proven to be a powerful analytical tool, shedding light on intricate patterns within customer behavior. Employing essential data manipulation tools like NumPy and Pandas, coupled with visualization techniques from Scipy and Scikit-Learn, facilitated a robust exploration of the dataset. The dendrogram, a visual representation of hierarchical relationships, played a pivotal role in determining the optimal number of clusters. Scikit-Learn's AgglomerativeClustering then effectively grouped customers into distinct segments based on their annual income and spending scores, revealing clusters labeled as 'Careful,' 'General,' 'Target,' 'Careless,' and 'Miser.' These clusters offer actionable insights for targeted marketing and engagement strategies, presenting a unique opportunity for businesses to tailor their approaches to specific customer segments. The hierarchical clustering approach not only identifies groupings but also captures the nuanced relationships among clusters. This analysis lays the groundwork for further refinement of business strategies, ensuring a more personalized and effective approach to customer satisfaction and overall business growth.

In [2]:
```
dataset = pd.read_csv('/kaggle/input/customer-segmentation-tutorial-in-python/Mall_Customers.csv')
X = dataset.iloc[:, [3, 4]].values
```

In [3]:
```
# Using the dendrogram to find the optimal number of clusters
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt
dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()
```



In [4]:
```
#From Dendrom, we understood that there are 5 clusters which we can get by drawing horizontal line near 200(Euclidean distance)
# Fitting Hierarchical Clustering to the dataset
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(X)
```

In [5]:
```
y_hc
```

Out[5]:
```
array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3,
       4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 1,
       4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 2, 0, 2, 0, 2,
       1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2])
```

In [6]:

```
# Visualising the clusters
plt.scatter(X[y_hc == 0, 0], X[y_hc == 0, 1], s = 100, c = 'red',
label = 'Carefull')
plt.scatter(X[y_hc == 1, 0], X[y_hc == 1, 1], s = 100, c = 'blue',
label = 'General')
plt.scatter(X[y_hc == 2, 0], X[y_hc == 2, 1], s = 100, c = 'green',
label = 'Target')
plt.scatter(X[y_hc == 3, 0], X[y_hc == 3, 1], s = 100, c = 'cyan',
label = 'Careless')
plt.scatter(X[y_hc == 4, 0], X[y_hc == 4, 1], s = 100, c = 'magenta',
label = 'Miser')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```