Chapter 4 : Part 2

**Hierarchical Clustering**

**4.2.1 Hierarchical Clustering**

Hierarchical clustering is another unsupervised machine learning algorithm, which is used to *group* the *unlabeled* *datasets* into a *cluster* and also known as hierarchical cluster analysis or HCA.

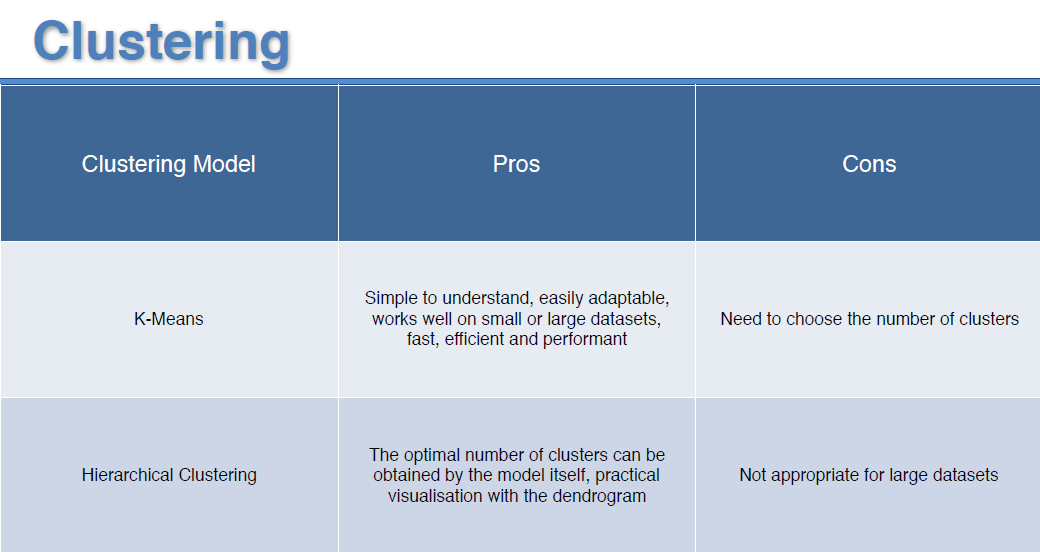
* Dendrogram: In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the dendrogram.
* Sometimes the results of ***K-Means Clustering*** and ***Hierarchical Clustering*** may look similar, but they both differ depending on how they work.
* There is no requirement to predetermine the ***number of clusters*** as we did in the K-Means algorithm.
* Agglomerative and Divisive clustering: The hierarchical clustering technique has two approaches:
* *Agglomerative*: Agglomerative is a ***bottom-up approach***, in which the algorithm starts with ***taking all data points as single clusters*** and ***merging*** them ***until one cluster is left***.
* *Divisive:* Divisive algorithm is the ***reverse*** of the ***agglomerative algorithm*** as it is a ***top-down approach***.
* Why hierarchical clustering?: As we already have other clustering algorithms such as ***K-Means Clustering***, then why we need hierarchical clustering?
* As we have seen in the ***K-means clustering*** that we need a ***predetermined*** ***number*** of ***clusters***, and
* It always tries to *create* the *clusters* of the *same* *size*.
* To solve these two challenges, we can opt for the ***hierarchical*** ***clustering*** ***algorithm*** because, in this algorithm, we don't need to have knowledge about the predefined number of clusters.

**4.2.2 Agglomerative Hierarchical clustering**

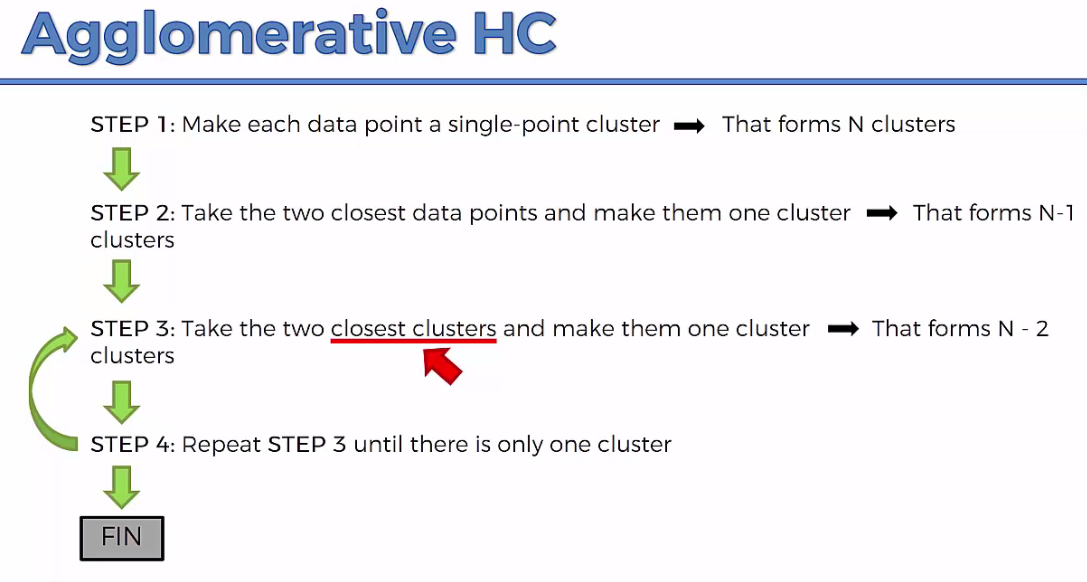
The Agglomerative Hierarchical Clustering algorithm is a popular example of ***HCA***. To group the datasets into clusters, it follows the Bottom-Up approach. It means, this algorithm considers ***each dat-point as a single cluster*** at the beginning, and then start COMBINING the *closest pair* of clusters together. It does this until all the clusters are merged into a single cluster that contains all the datasets.

* This Hierarchy Of Clusters is represented in the form of the DENDROGRAM.

**Clustering Pros & Cons**



* How Agglomerative HCA Work: The working of the ***AHC*** algorithm can be explained using the below steps:



* The way the hierarchical clustering algorithm works is that it *maintains a memory* of how we went through following process and that *memory* is *stored* in a DENDROGRAM.

|  |  |
| --- | --- |
| * Step-1: Create *each* *data* *point* as a *single* *cluster*. Let's say there are ***N data points***, so the number of ***clusters*** will also be ***N***. | Hierarchical Clustering in Machine Learning |
| * Step-2: Take *two* closest *data-points* or *clusters* and *merge* *them* to form one cluster. So, there will now be N-1 clusters. | Hierarchical Clustering in Machine Learning |
| * Step-3: Again, take the ***two closest clusters*** and ***merge*** them together to form one cluster. There will be ***N-2*** clusters. * Step-4: Repeat Step 3 until only *one* *cluster* left. So, we will get the following clusters. Consider the below images: | Hierarchical Clustering in Machine Learning |
| Hierarchical Clustering in Machine Learning Hierarchical Clustering in Machine Learning Hierarchical Clustering in Machine Learning | |
| * Step-5: Once all the clusters are combined into one big cluster, develop the ***Dendrogram*** to divide the clusters as per the problem. | |

**4.2.3 Distance between two clusters**

* Linkage Methods: The ***closest*** ***distance*** between the two clusters is ***crucial*** for the ***hierarchical*** ***clustering***. There are various ways to calculate the distance between two clusters, and these ways decide the rule for clustering. These measures are called Linkage methods. Some of the popular linkage methods are given below:

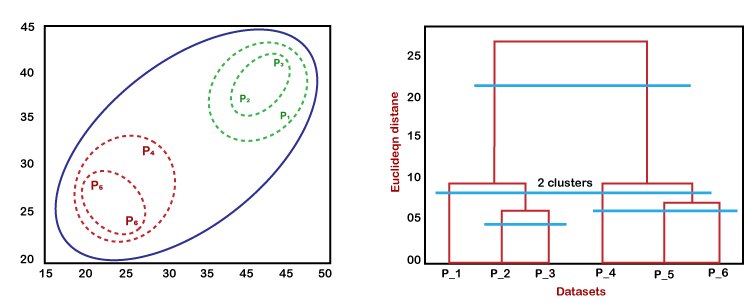
|  |  |
| --- | --- |
| 1. Single Linkage: It is the *Shortest Distance* between the *closest* *points* of the *clusters*. Consider the beside image: | Hierarchical Clustering in Machine Learning |
| 1. Complete Linkage: It is the *farthest* *distance* between the *two* *points* of two different *clusters*. It is one of the popular linkage methods as it forms tighter clusters than single-linkage. | Hierarchical Clustering in Machine Learning |
| 1. Average Linkage: It is the linkage method in which the *distance between each pair* of ***data-points*** is added up and then divided by the ***total*** ***number*** of ***data-points*** to calculate the average distance between two clusters. It is also one of the most popular linkage methods. | |
| 1. Centroid Linkage: It is the linkage method in which the ***distance*** ***between*** the ***centroid*** of the ***clusters*** is calculated. Consider the beside image: | Hierarchical Clustering in Machine Learning |

**4.2.4 Dendrogram in Hierarchical clustering**

The Dendrogram is a Tree-Like Structure that is mainly used to *store each step as a memory* that the *HC* *algorithm* *performs*. In the dendrogram plot, the

* *Y-axis* shows the *Euclidean* *distances* between the *data* *points*, and
* The *X-axis* shows *all the data points* of the given dataset.

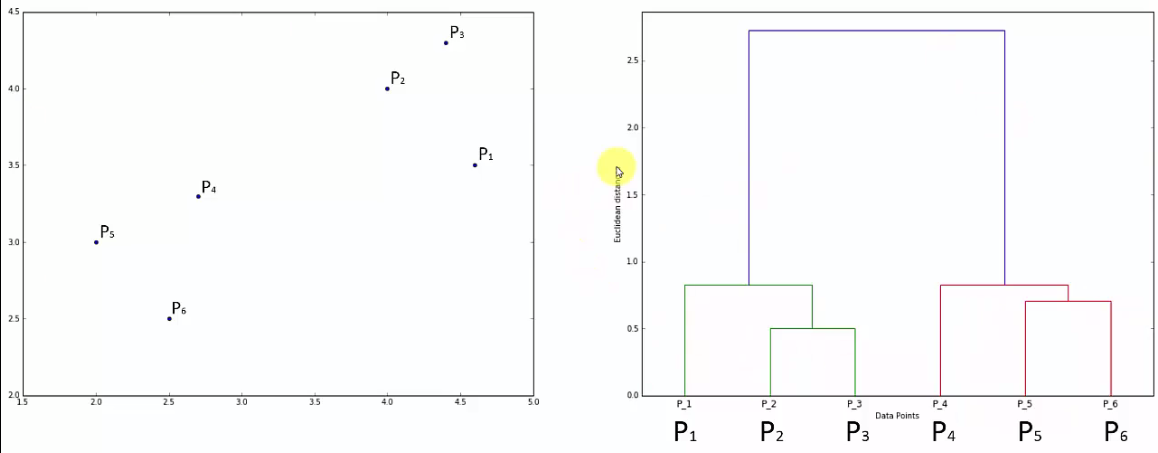
The working of the dendrogram can be explained using the below diagram:



* In the above diagram, the left part is showing how ***clusters*** are created in ***agglomerative*** ***clustering***, and the right part is showing the ***Corresponding*** ***Dendrogram***.

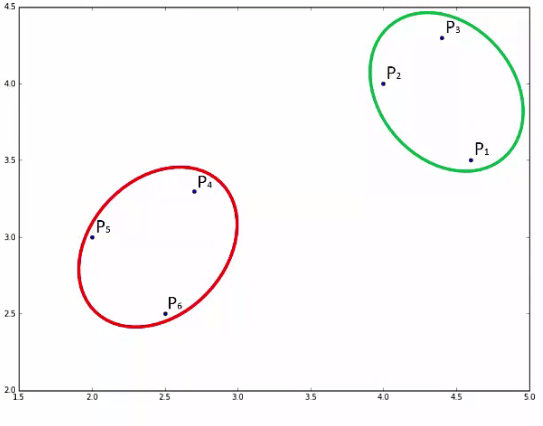
1. Firstly, the datapoints ***P2*** and ***P3*** combine together and form a *cluster*, correspondingly a *dendrogram* is *created*, which connects ***P2*** and ***P3*** with a ***rectangular*** ***shape***. The height is decided according to the *Euclidean* *distance* between the *data* *points*.
2. In the next step, ***P5*** and ***P6*** form a ***cluster***, and the corresponding ***dendrogram*** is created. It is higher than of previous, as the Euclidean distance between ***P5*** and ***P6*** is a *little bit greater* than the ***P2*** and ***P3***.
3. Again, two new dendrograms are created that combine ***P1***, ***P2***, and ***P3*** in one dendrogram, and ***P4***, ***P5***, and ***P6***, in another dendrogram.
4. At last, the ***final dendrogram*** is created that combines ***all the data points together***.

* We can *cut* the *dendrogram* tree structure at any level as per our requirement.
* Dendrogram contains the *memory* of the *hierarchical* *clustering* *algorithm*. You can understand in which order these classes were formed. Just by looking at the Dendrogram. Here I've got an example, this is the actual example generated by computer using HCA.

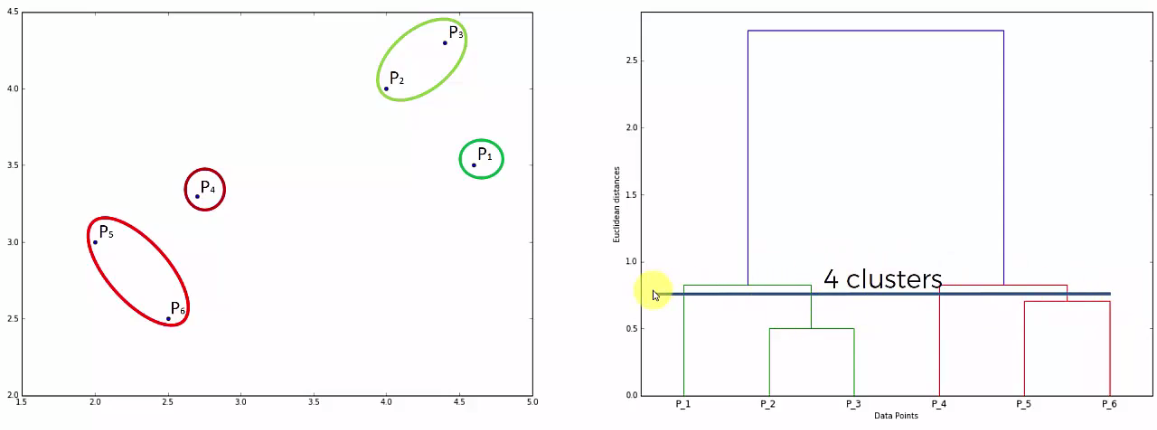


|  |  |
| --- | --- |
| * How do we get to that right number of clusters: Now we have to look at the horizontal levels and set thresholds so we can set heights thresholds or distance actually distance thresholds are also called dissimilarity thresholds because this vertical axis measures the Euclidean distance between points which also represents the *dissimilarity* between *them* or *points* or *clusters*. * So what we can do is *set a threshold* for all *dissimilarity* and we can say that we don't want dissimilarity to be greater than a certain level. |  |

* We were setting the dissimilarity threshold by saying that anything if we come across clusters that are above this *certain threshold* (i.e. above certain Euclidian distance) so we ***don't want within a cluster*** to have dissimilarity above this threshold. Then we end up with two clusters. The clysters are shown in the following figure:

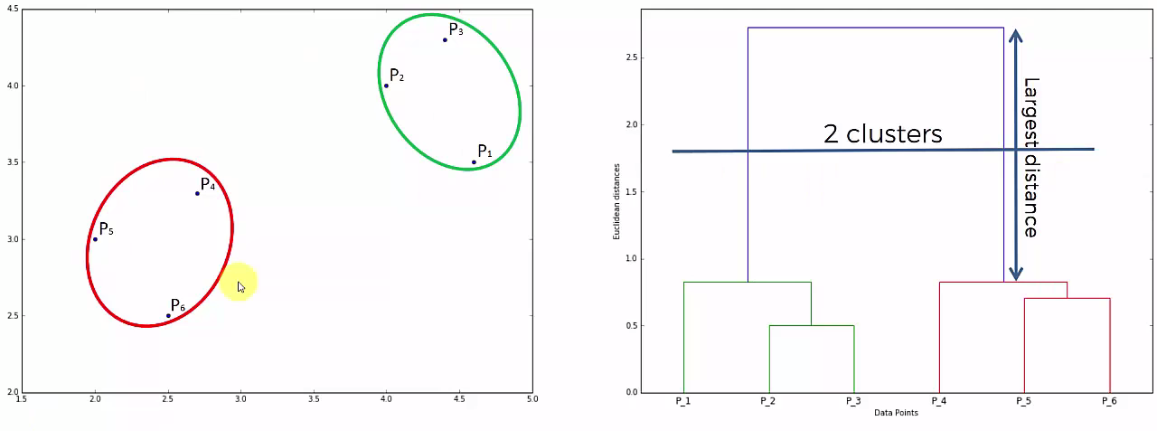


* What this is doing is it is not allowing any ***clusters*** that would have ***dissimilarity*** of ***greater*** than the *threshold value* *(1.7 in the figure)* within them.
* Calculating clusters numbers: You can quickly tell how many clusters you will have *at a certain threshold* by just looking at how many vertical lines this horizontal threshold actually crosses.
* So in above Dendrogram figure, the horizontal threshold line ***crosses two vertical line***, hence we get two clusters.
* In following case we have 4-clusters:

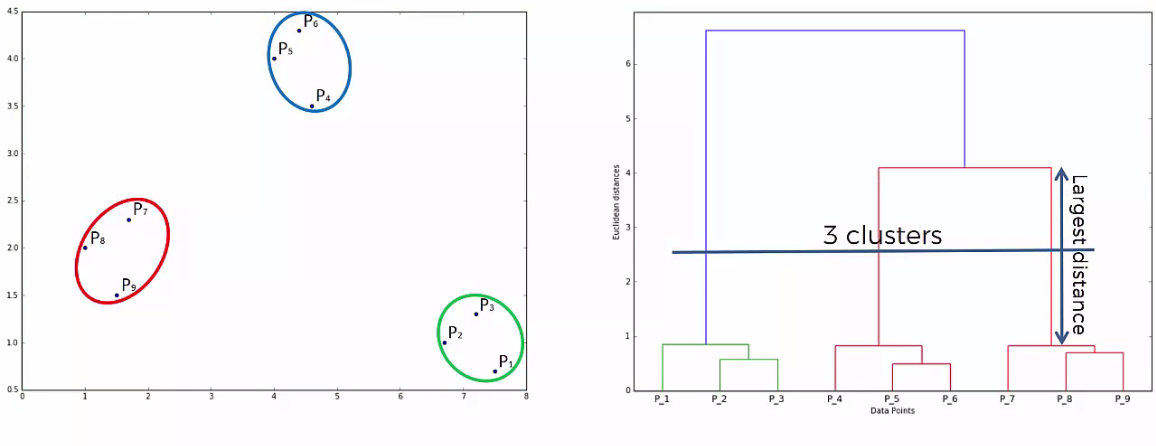


* Finding Optimal number of Clusters: Large vertical line means "Long distance/gap/void between the clusters". So we need to find a ***"threshold" that crosses maximum number "large vertical lines"***.
* We cannot consider the vertical lines that crosses "hypothetical extended horizontal line that forms dendrogram".

|  |  |
| --- | --- |
| * One of the standard approaches is just to look for the highest vertical distance that you can find in dendrogram. Any line that will not cross any horizontal lines. * For example: line at ***P2*** and ***P3*** can be considered but ***P1*** cannot be considered because it crosses ***hypothetical horizontal line (extended)*** that joins ***P2*** and ***P3***. Because it does not represents the distance between ***{P2, P3} cluster***, it actually distance between the single ***points P2***, ***P3***; and the actual distance between ***{P2, P3}*** and ***P1*** is the short vertical line (top on P2, P3). * So we have to find the ***largest vertical line that does not cross any of the existed horizontal lines that forms the dendrogram***. |  |



* One More Example:



**4.2.5 Python Implementation of Agglomerative Hierarchical Clustering**

We consider the same problem. The mall customer data.

* Problem Description: We have a dataset of ***Mall\_Customers***, which is the data of customers who visit the mall and spend there.
* In the given dataset, we have ***Customer\_Id***, ***Gender***, ***Age***, ***Annual Income ($)***, and ***Spending*** ***Score*** (which is the calculated value of how much a customer has spent in the mall, the more the value, the more he has spent). From this dataset, we need to calculate some patterns, as it is an unsupervised method, so we don't know what to calculate exactly.

#*Hierachical Clustering*

#*libraries*

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib**.**pyplot **as** plt

#*importing data*

dataSet = pd**.read\_csv**("Mall\_Customers.csv")

X = dataSet**.**iloc[:, [3, 4]]**.**values

#*There is no y for clustering*

#*Dendrogram :: Finding optimal noumber of clusters using "Dendrogram"*

**import** scipy**.**cluster**.**hierarchy **as** schr

dendrgm = schr**.dendrogram**(schr**.linkage**(X, method="ward"))

plt**.title**("Dendrogram")

plt**.xlabel**("Number of Clusters")

plt**.ylabel**("Euclidean distance")

plt**.show**()

#*creating cluster with optimal "n\_clusters". From "DENDROGRAM" we figured out that 5 is the optimal number of clusters*

**from** sklearn**.**cluster **import** AgglomerativeClustering

hrcl\_cluster\_genrt = **AgglomerativeClustering**(n\_clusters = 5, affinity="euclidean", linkage='ward')

y\_hrcl = hrcl\_cluster\_genrt**.fit\_predict**(X)

#*plotting the cluster*

plt**.scatter**(X[y\_hrcl **==** 0, 0], X[y\_hrcl **==** 0, 1], s = 100, c = "red", label="cluster 1")

plt**.scatter**(X[y\_hrcl **==** 1, 0], X[y\_hrcl **==** 1, 1], s = 100, c = "blue", label="cluster 2")

plt**.scatter**(X[y\_hrcl **==** 2, 0], X[y\_hrcl **==** 2, 1], s = 100, c = "green", label="cluster 3")

plt**.scatter**(X[y\_hrcl **==** 3, 0], X[y\_hrcl **==** 3, 1], s = 100, c = "cyan", label="cluster 4")

plt**.scatter**(X[y\_hrcl **==** 4, 0], X[y\_hrcl **==** 4, 1], s = 100, c = "pink", label="cluster 5")

#*No centroids in Hierarchical clustering*

#*plt.scatter(hrcl\_cluster\_genrt.cluster\_centers\_[:, 0], hrcl\_cluster\_genrt.cluster\_centers\_[:, 1], s=300, c="black", label = "Centroids")*

plt**.title**("Clusters Of Clients")

plt**.xlabel**("Annual income ($)")

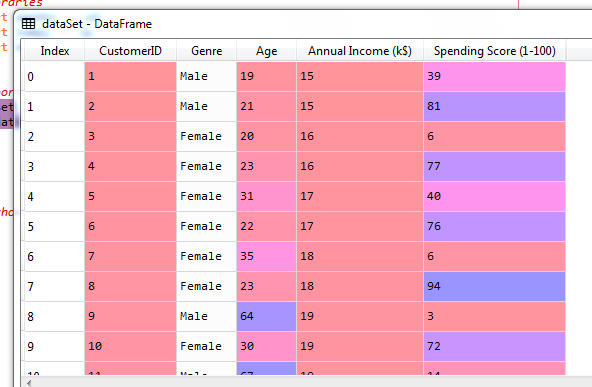
plt**.ylabel**("Spending score (1-100)")

plt**.legend**()

plt**.show**()

#*python prctc\_hrcl\_cltr.py*

* Data preprocessing: There is *no y dependent variable* in clustering.
* We are extracting only ***4th and 5th feature-column:*** *Annual Income* and *Spending Score*.



* Selecting Library and Class:

**from** sklearn**.**cluster **import** KMeans

* The Dendrogram: Now we will find the *optimal* *number* of clusters using the Dendrogram for our model. For this, we are going to use ***scipy*** library as it provides a function that will directly return the ***dendrogram*** for our code. Consider the below lines of code:

#*Dendrogram :: Finding optimal noumber of clusters using "Dendrogram"*

**import** scipy**.**cluster**.**hierarchy **as** schr

dendrgm = schr**.dendrogram**(schr**.linkage**(X, method="ward"))

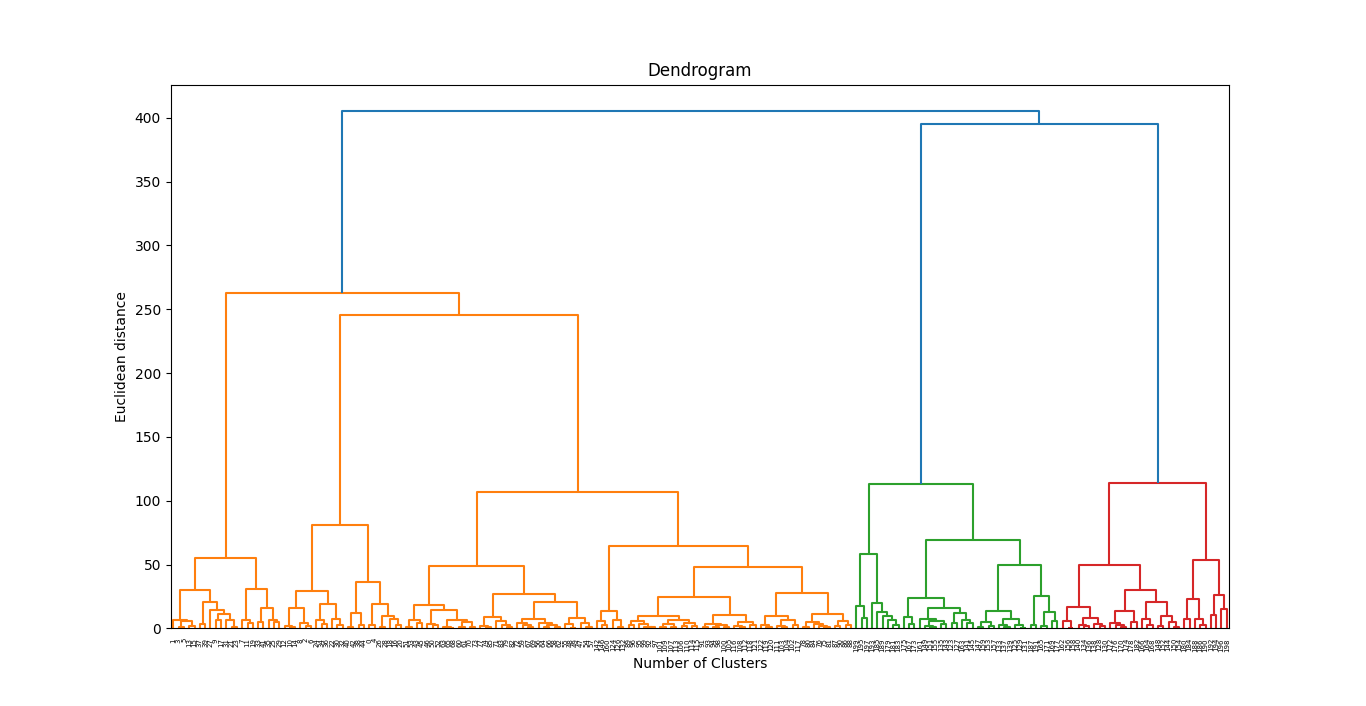
plt**.title**("Dendrogram")

plt**.xlabel**("Number of Clusters")

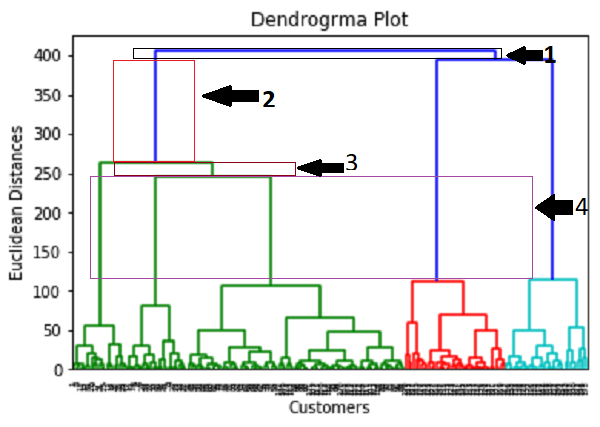
plt**.ylabel**("Euclidean distance")

plt**.show**()

* In the above lines of code, we have imported the ***hierarchy module*** of ***scipy*** library. This module provides us a method ***denrogram()***, which takes the ***linkage()*** as a parameter. The ***linkage*** function is used to define the ***distance*** ***between*** two ***clusters***, so here we have passed the ***X*** (matrix of features), and method "***ward***," the popular method of linkage in hierarchical clustering.
* The remaining lines of code are to describe the labels for the dendrogram plot.
* Output: By executing the above lines of code, we will get the below output:



* Optimal Number Of Clusters: Using this *Dendrogram*, we will now determine the *optimal* *number* of *clusters* for our model. For this, we will find the ***maximum vertical distance*** that ***does not cut any horizontal bar***. Consider the below diagram:



* In the above diagram, we have shown the ***vertical distances*** that are ***not cutting*** their ***horizontal bars***. As we can visualize, the ***4th*** distance is looking the maximum, so according to this, the number of clusters will be 5(the vertical lines in this range). We can also take the ***2nd*** number as it approximately equals the ***4th*** distance, but we will consider the 5 clusters because the same we calculated in the K-means algorithm.
* In the Dendrogram the largest distance is considered the ***Rightmost*** ***Blue*** ***Vertical*** ***Line***, but it cut by two horizontal lines approximately at ***270*** and ***247***. Hence we take the ***range*** from ***248*** to ***120***, because in this range *no* *horizontal* *lines* *appear*. However we can take the upper portion of this vertical-line, but as mention above we need to find the number near to k-means cluster.
* So, the ***optimal number*** of ***clusters*** will be ***5***, and we will train the model in the next step, using the same as we did in k-means clusters.
* Note: the ***vertical-range*** is taken in between the ***cuts*** of ***horizontal lines***.
* Training the hierarchical clustering model: As we know the required ***optimal number*** of clusters, we can now train our model.

**from** sklearn**.**cluster **import** AgglomerativeClustering

hrcl\_cluster\_genrt = **AgglomerativeClustering**(n\_clusters = 5, affinity="euclidean", linkage='ward')

y\_hrcl = hrcl\_cluster\_genrt**.fit\_predict**(X)

* In the above code, we have imported the ***AgglomerativeClustering*** class of cluster module of ***scikit*** learn library.
* Then we have created the object of this class named as ***hrcl\_cluster\_genrt***. The ***AgglomerativeClustering*** class takes the following parameters:
* ***n\_clusters=5:*** It defines the ***number*** of ***clusters***, and we have taken here ***5*** because it is the ***optimal*** ***number*** of clusters.
* ***affinity='euclidean':*** It is a ***metric*** used to ***compute*** the ***linkage***.
* ***linkage='ward':*** It defines the ***linkage*** ***criteria***, here we have used the "***ward***" ***linkage***. This method is the ***popular*** ***linkage*** ***method*** that we have already used for creating the ***Dendrogram***. It reduces the variance in each cluster.
* In the last line, we have created the dependent variable ***y\_hrcl*** to *fit or train* the model notice we used ***fit\_predict()*** as we did for ***k-means*** ***clustering***. It does train not only the model but also returns the clusters to which each data point belongs.
* After executing the above lines of code, if we go through the variable explorer option in our Sypder IDE, we can check the y\_pred (in our case ***y\_hrcl***) variable. We can compare the original dataset with the y\_pred variable. Consider the below image:



* As we can see in the above image, the ***y\_pred*** shows the ***clusters value***, which means the customer id ***1*** belongs to the ***5th cluster*** (as indexing starts from ***0***, so ***4*** means ***5th cluster***), the *customer id 2* belongs to *4th cluster*, and so on.
* Visualizing the Clusters: Here we will use the same lines of code as we did in ***k-means clustering***, except one change. Here we will ***not*** ***plot*** the ***centroid*** that we did in ***k-means***, because here we have used ***dendrogram*** to determine the ***optimal*** ***number*** of clusters. The code is given below:
* Notice that there is no centroid here.

#*plotting the cluster*

plt**.scatter**(X[y\_hrcl **==** 0, 0], X[y\_hrcl **==** 0, 1], s = 100, c = "red", label="cluster 1")

plt**.scatter**(X[y\_hrcl **==** 1, 0], X[y\_hrcl **==** 1, 1], s = 100, c = "blue", label="cluster 2")

plt**.scatter**(X[y\_hrcl **==** 2, 0], X[y\_hrcl **==** 2, 1], s = 100, c = "green", label="cluster 3")

plt**.scatter**(X[y\_hrcl **==** 3, 0], X[y\_hrcl **==** 3, 1], s = 100, c = "cyan", label="cluster 4")

plt**.scatter**(X[y\_hrcl **==** 4, 0], X[y\_hrcl **==** 4, 1], s = 100, c = "pink", label="cluster 5")

#*No centroids in Hierarchical clustering*

#*plt.scatter(hrcl\_cluster\_genrt.cluster\_centers\_[:, 0], hrcl\_cluster\_genrt.cluster\_centers\_[:, 1], s=300, c="black", label = "Centroids")*

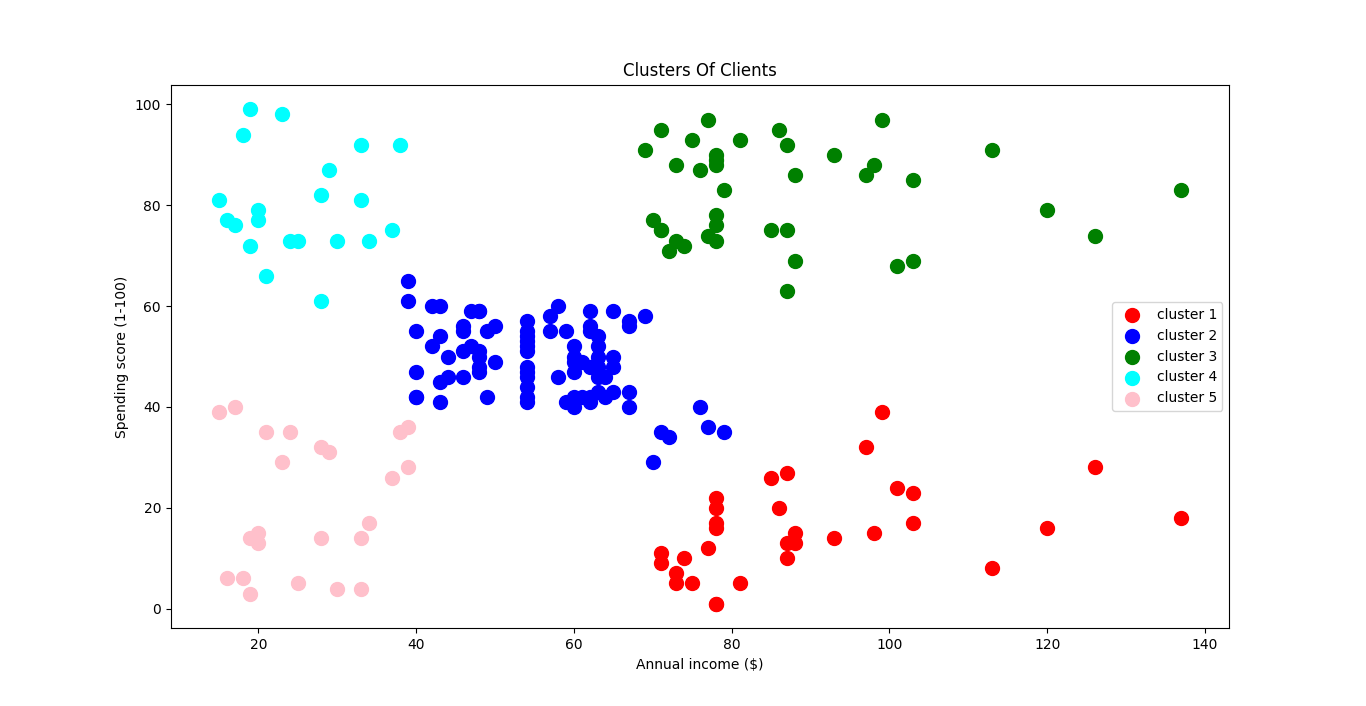
plt**.title**("Clusters Of Clients")

plt**.xlabel**("Annual income ($)")

plt**.ylabel**("Spending score (1-100)")

plt**.legend**()

plt**.show**()

****

* Categorize the customers: The output image is clearly showing the ***five different clusters*** with different colors. The clusters are formed between two parameters of the dataset; ***Annual*** ***income*** of customer and ***Spending***. We can change the colors and labels as per the requirement or choice. We can also observe some points from the above patterns, which are given below:

1. ***Cluster1*** shows the customer has a high income but low spending, so we can categorize them as Careful
2. ***Cluster2*** shows the customers with average salary and average spending so we can categorize these customers as Standard
3. ***Cluster3*** shows the customers with high income and high spending so they can be categorized as Target, and these customers can be the most profitable customers for the mall owner.
4. ***Cluster4*** shows the customers with low income with very high spending so they can be categorized as Careless.
5. ***Cluster5*** shows the low income and also low spending so they can be categorized as Sensible.

* Multi-Dimensional Clustering: For 3 feature variable we still can visualize the cluster in 3-D graphics, but more tan 3D we can't plot the clusters.
* However later we will learn a technique that allows us to reduce the dimensions of our data. So that you can plot the clusters.
* If you are doing clustering in more than two dimensions then don't execute the last code lines in this section to visualize the clusters. Because it's only for two dimensional clustering.