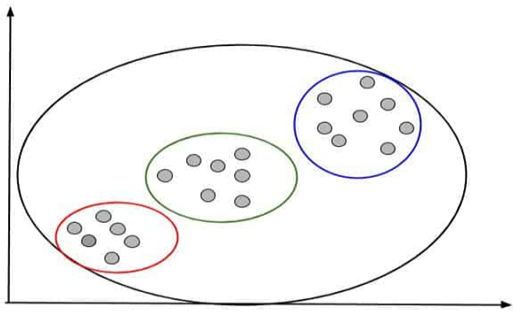
**Exercise-9**

**AIM: Implementation of Hierarchical Agglomerative Clustering Algorithm**

**DESCRIPTION:**

**Hierarchal Clustering:** Hierarchical Clustering groups similar objects into one cluster. The final cluster in the Hierarchical cluster combines all clusters into one cluster.

**An example** of Hierarchical clustering is **Dendrogram.** Hierarchical clustering cluster the **data points based on its similarity.** Hierarchical clustering continues clustering until one single cluster left. you can see in this image. Hierarchical clustering combines all three smaller clusters into one final cluster.

### Type of Hierarchical Clustering

Hierarchical Clustering is of 2 types-

1. **Agglomerative Hierarchical Clustering.**
2. **Divisive Hierarchical Clustering.**

#### 1. Agglomerative Hierarchical Clustering.

Agglomerative Hierarchical Clustering uses a **bottom-up approach**to form clusters. That means it starts from single data points. Then it clusters the closer data points into one cluster. The same process repeats until it gets one single cluster.

### Steps to Perform Hierarchical Clustering:

## Step 1- Make each data point a single cluster. Suppose that forms n clusters.

## 

## Step 2- Take the 2 closet data points and make them one cluster. Now the total

## clusters become n-1.

## 

## Step 3-Take the 2 closet clusters and make them one cluster. Now the total clusters

## become n-2.

## 

## Step 4- Repeat Step 3 until only one cluster is left.

## 

## Python Implementation of Aggloramative Hierarchical Clustering Algorithm:

We have a dataset of **Mall\_Customers\_dataset.csv**, 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.

The **steps to be followed for the implementation** are given below:

* Data Pre-processing
* Finding the optimal number of clusters using the elbow method
* Training the K-means algorithm on the training dataset
* Visualizing the clusters

**1. Data Pre-processing:**

**(a) Importing Libraries:** firstly, we will import the libraries for our model, which is part of data pre-processing. The code is given below:

**# importing libraries**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

**(b) Importing the Dataset:** Next, we will import the dataset that we need to use. So here, we are using the **Mall\_Customer\_data.csv** dataset. It can be imported using the below code:

**# Importing the dataset**

dataset = pd.read\_csv('Mall\_Customers\_dataset.csv')

**(c ) Extracting Independent Variables:** Here we don't need any dependent variable for data pre-processing step as it is a clustering problem, and we have no idea about what to determine. So we will just add a line of code for the matrix of features.

**x = dataset. iloc [:, [3, 4]].values**

### Step-2: We have loaded dataset. Now it’s time to find the optimal number of clusters. And for that we need to create a Dendrogram

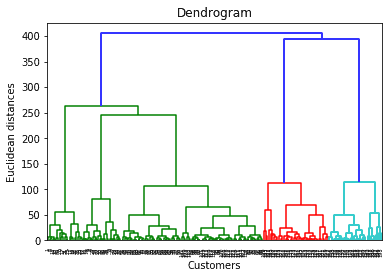
### Create Dendrogram to find the Optimal Number of Clusters

**scipy.cluster.hierarchy as sch dendro = sch.dendrogram(sch.linkage(X, method = 'ward'))**

### Here in the code “sch” is the short code for scipy.cluster.hierarchy.”

**“dendro”** is the variable name. It may be anything. And **“Dendrogram”** is the function name.

**So, after implementing this code, we will get our Dendrogram.**



As I discussed that cut the **horizontal line with longest line** that **traverses maximum distance** up and down **without intersecting the merging points**. In that dendrogram, the optimal number of clusters are 5.

### Step- 3: Training the Aggloramative Hierarchical Clustering algorithm on the training dataset:

**from sklearn.cluster import AgglomerativeClustering**

**hc = AgglomerativeClustering(n\_clusters = 5, affinity = 'euclidean', linkage = 'ward')**

**y\_hc = hc.fit\_predict(X)**

### Step-4: Visualizing the Clusters:

The last step is to visualize the clusters. As we have 5 clusters for our model, so we will visualize each cluster one by one. To visualize the clusters will use scatter plot using **plt.scatter() function** of matplotlib.

**#visulaizing the clusters**

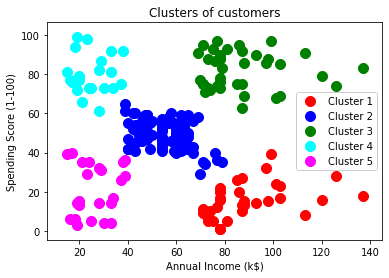
plt.scatter(X[y\_hc == 0, 0], X[y\_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1') plt.scatter(X[y\_hc == 1, 0], X[y\_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2') plt.scatter(X[y\_hc == 2, 0], X[y\_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3') plt.scatter(X[y\_hc == 3, 0], X[y\_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') plt.scatter(X[y\_hc == 4, 0], X[y\_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5') plt.title('Clusters of customers') plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')

plt.legend()

plt.show()

In above **lines of code,** we have written code for each clusters, ranging from 1 to 5. The first coordinate of the **plt.scatter, i.e., x[y\_hc == 0, 0]** containing the x value for the showing the matrix of features values, and the y\_predict is ranging from 0 to 1.



**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:

* **Cluster1 shows the customers with average salary** and **average spending** so we can categorize these customers as
* **Cluster2 shows the customer has a high income** but **low spending**, so we can categorize them as careful.
* **Cluster3** shows the **low income** and also **low spending** so they can be categorized as sensible.
* **Cluster4** shows the customers with **low income** with **very high spending** so they can be categorized as careless.
* **Cluster5** 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.

**PROGRAM:**

**# importing libraries**

import numpy as nm

import matplotlib.pyplot as plt

import pandas as pd

**# Loading the dataset**

dataset = pd.read\_csv('Mall\_Customers\_dataset.csv')

print(dataset)

x = dataset.iloc[:, [3, 4]].values

print(x)

**#Create Dendrogram to find the Optimal Number of Clusters**

import scipy.cluster.hierarchy as sch

dendro = sch.dendrogram(sch.linkage(X, method = 'ward'))

plt.title('Dendrogram')

plt.xlabel('Customers')

plt.ylabel('Euclidean distances')

plt.show()

**#Fitting Agglomerative 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)

y\_hc

**#Visualise the clusters**

plt.scatter(X[y\_hc == 0, 0], X[y\_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1')

plt.scatter(X[y\_hc == 1, 0], X[y\_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')

plt.scatter(X[y\_hc == 2, 0], X[y\_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3')

plt.scatter(X[y\_hc == 3, 0], X[y\_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')

plt.scatter(X[y\_hc == 4, 0], X[y\_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')

plt.title('Clusters of customers')

plt.xlabel('Annual Income (k$)')

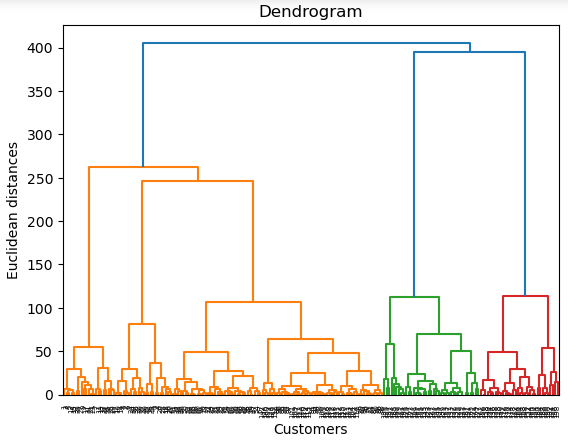
plt.ylabel('Spending Score (1-100)')

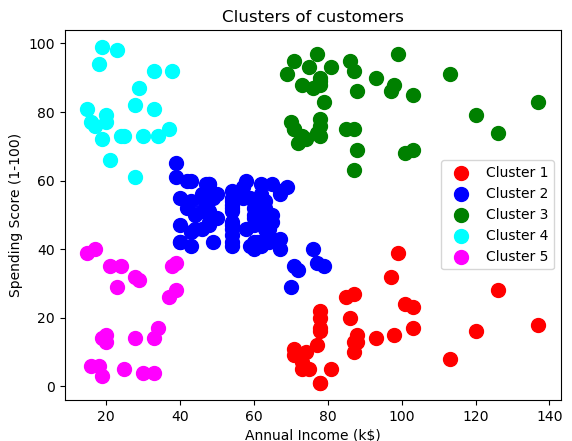
plt.legend()

plt.show()

**INPUT/OUTPUT:**

**# Creating Dendrogram**

****

**# Optimal No. of Clusters**

**CONCLUSION: Program is executed successfully without any error.**