Question 1: Assignment Summary

Briefly describe the "Clustering of Countries" assignment that you just completed within 200-300 words. Mention the problem statement and the solution methodology that you followed to arrive at the final list of countries. Explain your main choices briefly (what EDA you performed, which type of Clustering produced a better result and so on)

Note: You don't have to include any images, equations or graphs for this question. Just text should be enough.

HELP International is an international humanitarian NGO that is committed to fighting poverty and providing people of backward countries with basic amenities during the time of natural calamities. They have now raised $10 million and the CEO wants to decide which countries are in dire need of aid. Main task is to cluster the countries by the various factors.

We started by first importing all the required libraries that will be used as part of clustering assignment. Next, we inspected the data, its data types, shape of the data frame and described its various statistics. Verified that data does not require any cleaning.

We transformed the columns export, import and health to actual values from the % of gdp per capita values. Exploratory Data Analysis was performed to identify data trends. This also included Univariate and bivariate analysis. Top/Bottom 10 countries were displayed for each metric depending on the final selection criteria. We also plotted heatmap and pair plot to identify the correlation between each variable.

Outlier Analysis was performed using the boxplot and IQR (Inter quartile range) and outliers were treated to remove/impute extreme data values using 1% and 99% as threshold values. The standardization/normalization of values was done using Standard Scalar so that all the metrics are given equal importance. Hopkins Statistics was calculated to determine if the clustering was possible.

We then performed clustering using K means algorithm. We identified the number of clusters using the Elbow curve and Silhouette Analysis. The Elbow curve gave us 3 as the optimal clusters and Silhouette Analysis gave us 4-5 clusters as the k value. We went ahead using 3 as the value for k. The reason being there were 2 clusters who had only 1 country each. Then we visualised the clusters using various metrics to identify which cluster has higher/lower variables. In the end we did cluster profiling to identify the top 5 countries which needed help.

Hierarchical clustering was performed and dendrograms were generated using Single and Complete linkage. This was used to identify the ideal number of clusters for hierarchical clustering. We took 5 as the number of clusters in hierarchical clustering. We identified the clusters and the visualised it with various metrics. In the end we did cluster profiling to identify the top 5 countries which needed help from the NGO.

Question 2: Clustering

a) Compare and contrast K-means Clustering and Hierarchical Clustering.

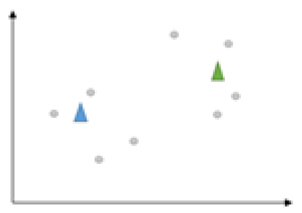
K-means

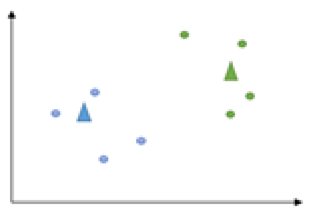
The first step of this algorithm is creating, among our unlabelled observations, c new observations, randomly located, called 'centroids'. The number of centroids will be representative of the number of output classes (which, remember, we do not know). Now, an iterative process will start, made of two steps:

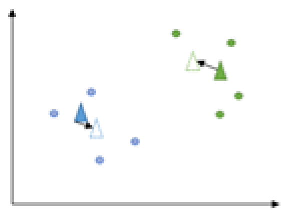
- First, for each centroid, the algorithm finds the nearest points (in terms of distance that is usually computed as Euclidean distance) to that centroid, and assigns them to its category;

- Second, for each category (represented by one centroid), the algorithm computes the average of all the points which has been attributed to that class. The output of this computation will be the new centroid for that class.

Every time the process is reiterated, some observations, initially classified together with one centroid, might be redirected to another one. Furthermore, after several reiterations, the change in centroids' location should be less and less important since the initial random centroids are converging to the real ones. This process ends when there is no more change in centroids' position.


1. Two centroids are randomly located among data


2. The algorithm finds the nearest observations for each centroid, and then attributes them to each centroid's class


3. New centroids, resulting from the average of each class observations, are computed, and then the process is repeated

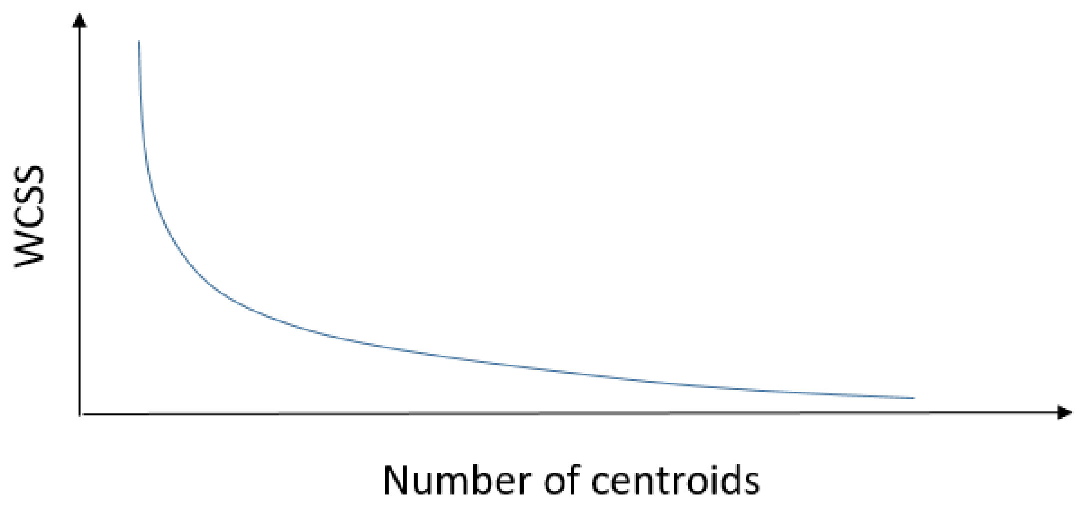
Now, how can we decide the number of centroids?

There are many methods that could be employed for this task. 'Elbow method' in this method the idea is that what we would like to observe within our clusters is a low level of variation, which is measured with the within-cluster sum of squares (WCSS):

$$WCSS = \sum_{xi \in c} (x_i - \bar{x})^2$$

And it is intuitive to understand that, the higher the number of centroids, the lower the WCSS. In particular, if we have as many centroids as the number of our observations, each WCSS will be equal to zero. However, if we remember the law of parsimony, we know that setting the highest number possible of centroids would be inconsistent.

The idea is picking that number of centroids after which the reduction in WCSS is irrelevant. The relation I've just described can be represented with the following graph:



The idea is that, if the plot is an arm, the elbow of the arm is the optimal number of centroids.

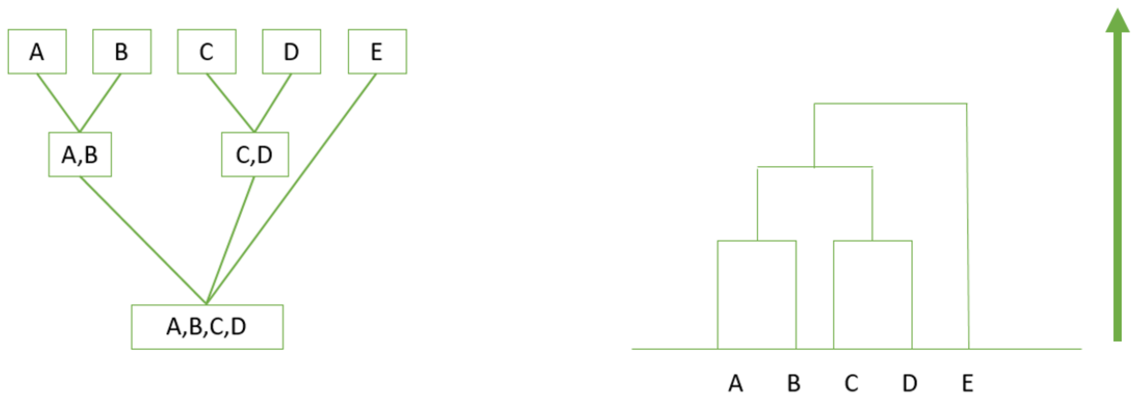
Hierarchical Clustering

This algorithm can use two different techniques:

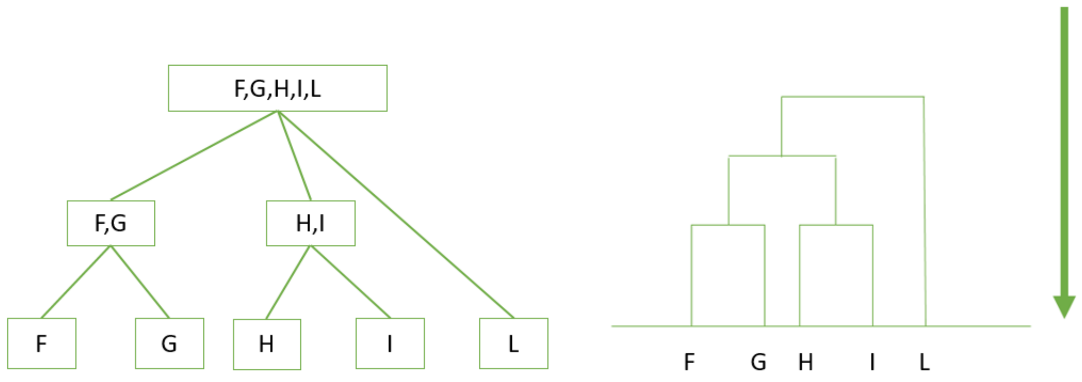
- Agglomerative
- Divisive

Those latter are based on the same ground idea, yet work in the opposite way: being K the number of clusters (which can be set exactly like in K-means) and n the number of data points, with n>K, agglomerative HC starts from n clusters, then aggregates data until it obtains K clusters; divisive HC, on the other hand, starts from just one cluster and then splits it depending, again, on similarities, until it obtains K clusters. Note that, when I say similarities, I'm referring to the distance among data points, which can be computed in different ways (I will dwell on this later on).

Let's visualize both the agglomerative and divisive techniques:



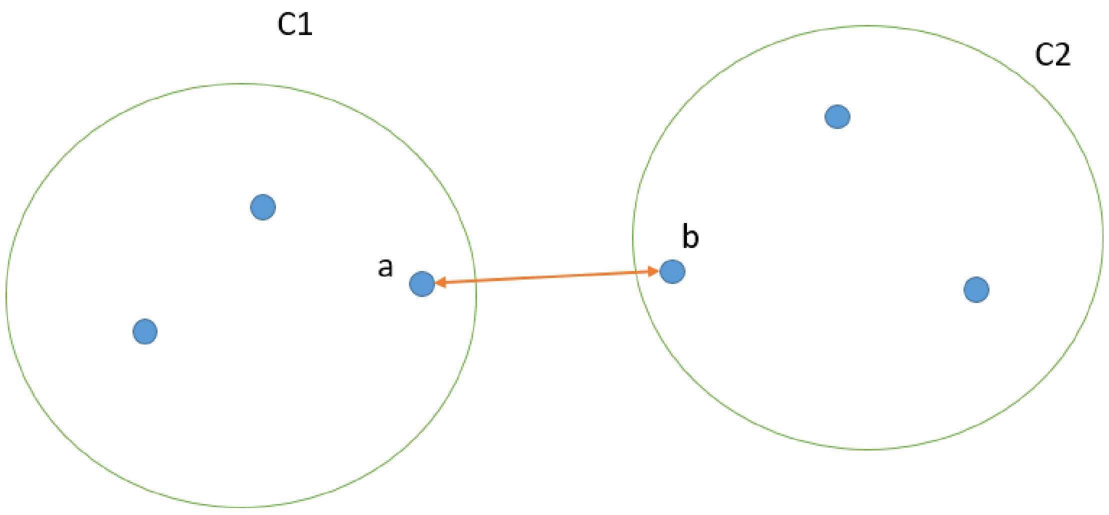
Agglomerative HC



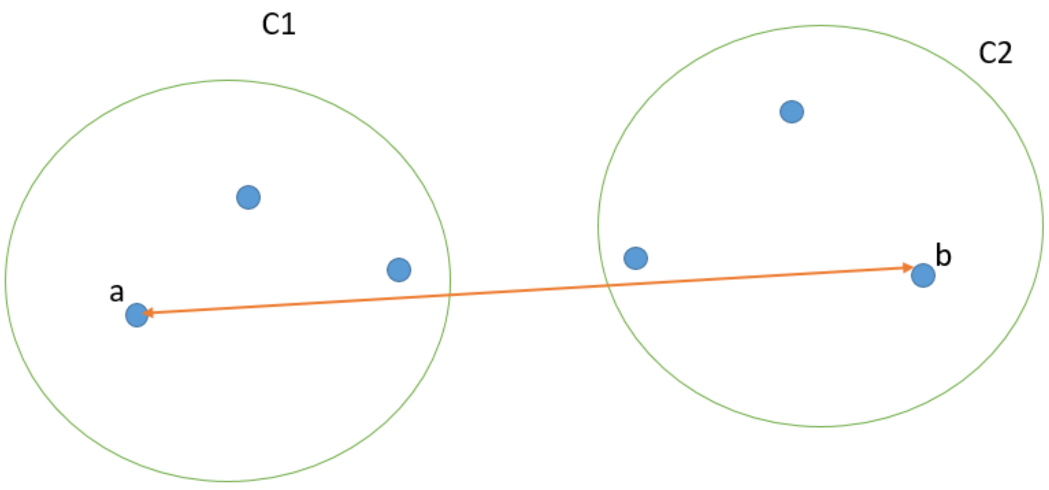
Divisive HC

As anticipated, the key element of discrimination here is similarity among data points. In mathematical terms, similarity mainly refers to distance, and it can be computed with different approaches. Here, I will propose three of them:

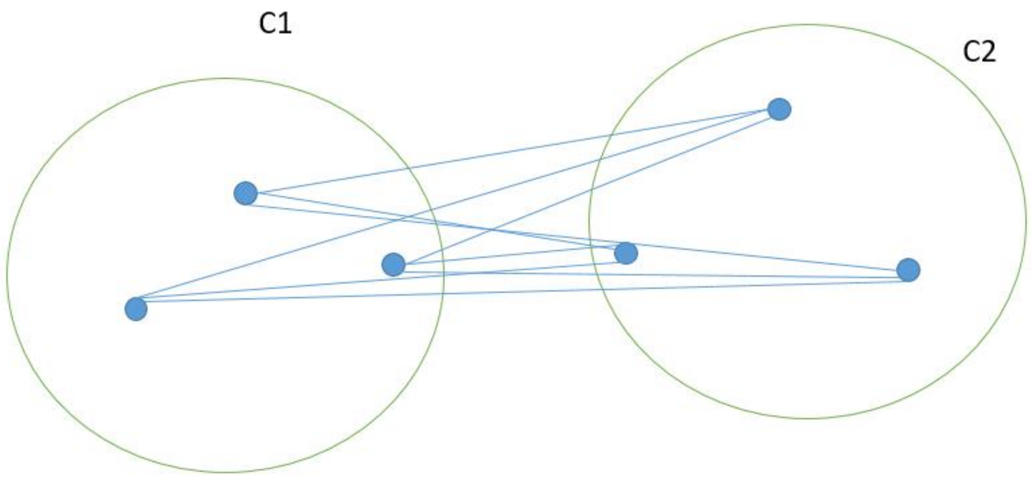
- Min: it states that, given two clusters C1 and C2, the similarity between them is equal to the minimum of similarity (translated: distance) between point a and b, such that a belongs to C1 and b belongs to C2.



- Max: it states that, given two clusters C1 and C2, the similarity between them is equal to the maximum of similarity between point a and b, such that a belongs to C1 and b belongs to C2.



- Average: it takes all the pairs of points, compute their similarities and then calculate the average of the similarities. That latter is the similarity between the clusters C1 and C2.



So, to recap, both the algorithms look for similarities among data and both use the same approaches to decide the number of clusters. Choosing the one rather than the other really depends on the kind of task you're facing.

Time Complexity
K-means is linear in the number of data objects i.e. O(n)O(n), where n is the number of data objects. The time complexity of most of the hierarchical clustering algorithms is quadratic i.e. O(n2)O(n2). Therefore, for the same amount of data, hierarchical clustering will take quadratic amount of time. Imagine clustering 1 million records?

Shape of Clusters
K-means works well when the shape of clusters are hyper-spherical (or circular in 2 dimensions). If the natural clusters occurring in the dataset are non-spherical then probably K-means is not a good choice.

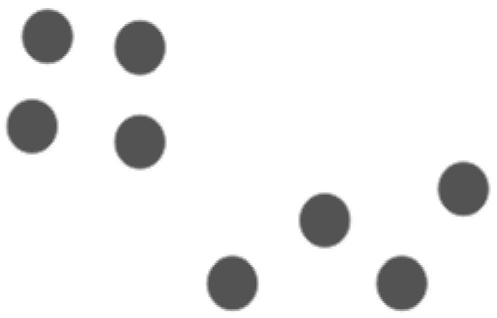
Repeatability
K-means starts with a random choice of cluster centers, therefore it may yield different clustering results on different runs of the algorithm. Thus, the results may not be repeatable and lack consistency. However, with hierarchical clustering, you will most definitely get the same clustering results.

Off course, K-means clustering requires prior knowledge of K (or number of clusters), whereas in hierarchical clustering you can stop at whatever level (or clusters) you wish.

b) Briefly explain the steps of the K-means clustering algorithm.

The first property of clusters – it states that the points within a cluster should be similar to each other. So, our aim here is to minimize the distance between the points within a cluster. K-means is a centroid-based algorithm, or a distance-based algorithm, where we calculate the distances to assign a point to a cluster. In K-Means, each cluster is associated with a centroid. The main objective of the K-Means algorithm is to minimize the sum of distances between the points and their respective cluster centroid.

Let's now take an example to understand how K-Means actually works:



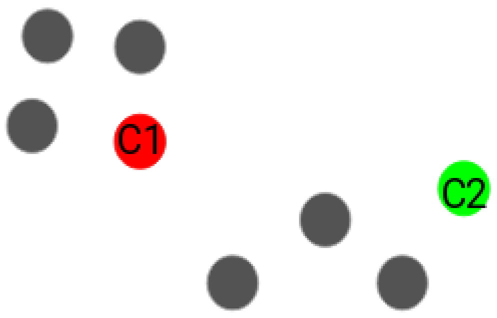
We have these 8 points and we want to apply k-means to create clusters for these points. Here's how we can do it.

Step 1: Choose the number of clusters k

The first step in k-means is to pick the number of clusters, k.

Step 2: Select k random points from the data as centroids

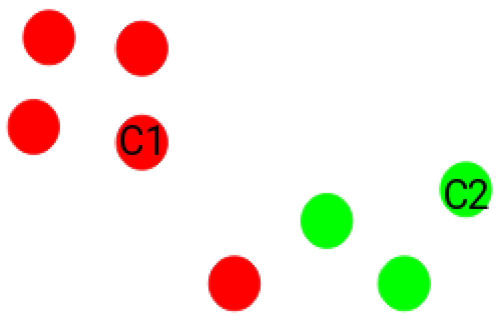
Next, we randomly select the centroid for each cluster. Let's say we want to have 2 clusters, so k is equal to 2 here. We then randomly select the centroid:



Here, the red and green circles represent the centroid for these clusters.

Step 3: Assign all the points to the closest cluster centroid

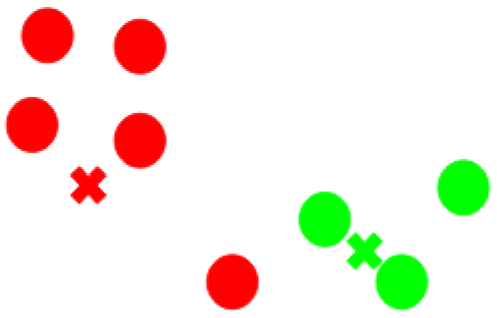
Once we have initialized the centroids, we assign each point to the closest cluster centroid:



Here you can see that the points which are closer to the red point are assigned to the red cluster whereas the points which are closer to the green point are assigned to the green cluster.

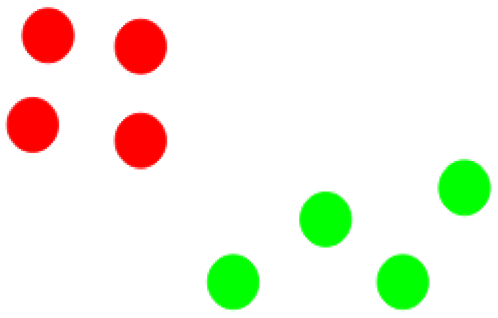
Step 4: Recompute the centroids of newly formed clusters

Now, once we have assigned all of the points to either cluster, the next step is to compute the centroids of newly formed clusters:



Here, the red and green crosses are the new centroids.

Step 5: Repeat steps 3 and 4. We then repeat steps 3 and 4:



The step of computing the centroid and assigning all the points to the cluster based on their distance from the centroid is a single iteration.

Stopping Criteria for K-Means Clustering

There are essentially three stopping criteria that can be adopted to stop the K-means algorithm:

1. Centroids of newly formed clusters do not change
2. Points remain in the same cluster
3. Maximum number of iterations are reached

We can stop the algorithm if the centroids of newly formed clusters are not changing. Even after multiple iterations, if we are getting the same centroids for all the clusters, we can say that the algorithm is not learning any new pattern and it is a sign to stop the training.

Another clear sign that we should stop the training process if the points remain in the same cluster even after training the algorithm for multiple iterations.

Finally, we can stop the training if the maximum number of iterations is reached. Suppose if we have set the number of iterations as 100. The process will repeat for 100 iterations before stopping.

c) How is the value of 'k' chosen in K-means clustering? Explain both the statistical as well as the business aspect of it.

Determining the **optimal number of clusters** in a data set is a fundamental issue in partitioning clustering, such as k-means clustering, which requires the user to specify the number of clusters k to be generated.
Unfortunately, there is no definitive answer to this question. The optimal number of clusters is somehow subjective and depends on the method used for measuring similarities and the parameters used for partitioning.
A simple and popular solution consists of inspecting the dendrogram produced using hierarchical clustering to see if it suggests a particular number of clusters. Unfortunately, this approach is also subjective.
These methods include direct methods and statistical testing methods:

1. Direct methods: consists of optimizing a criterion, such as the within cluster sums of squares or the average silhouette. The corresponding methods are named elbow and silhouette methods, respectively.
2. Statistical testing methods: consists of comparing evidence against null hypothesis. An example is the gap statistic.
   In addition to elbow, silhouette and gap statistic methods, there are more than thirty other indices and methods that have been published for identifying the optimal number of clusters.

   The Elbow Method

   This is probably the most well-known method for determining the optimal number of clusters. It is also a bit naive in its approach. Calculate the **Within-Cluster-Sum of Squared** Errors (WSS) for **different values of k**, and choose the k for which WSS becomes first starts to diminish. In the plot of WSS-versus-k, this is visible as an **elbow.**

   Within-Cluster-Sum of Squared Errors sounds a bit complex. Let's break it down:

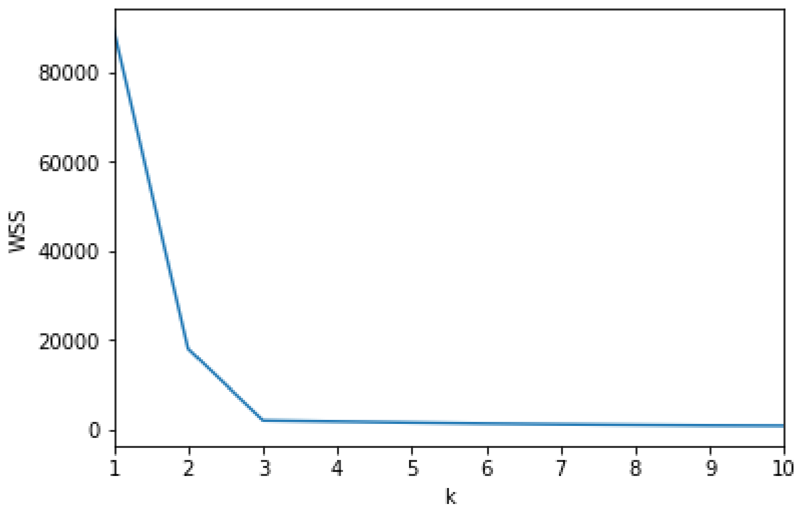
   - The Squared Error for each point is the square of the distance of the point from its representation i.e. its predicted cluster center.

   - The WSS score is the sum of these Squared Errors for all the points.

- Any distance metric like the Euclidean Distance or the Manhattan Distance can be used.

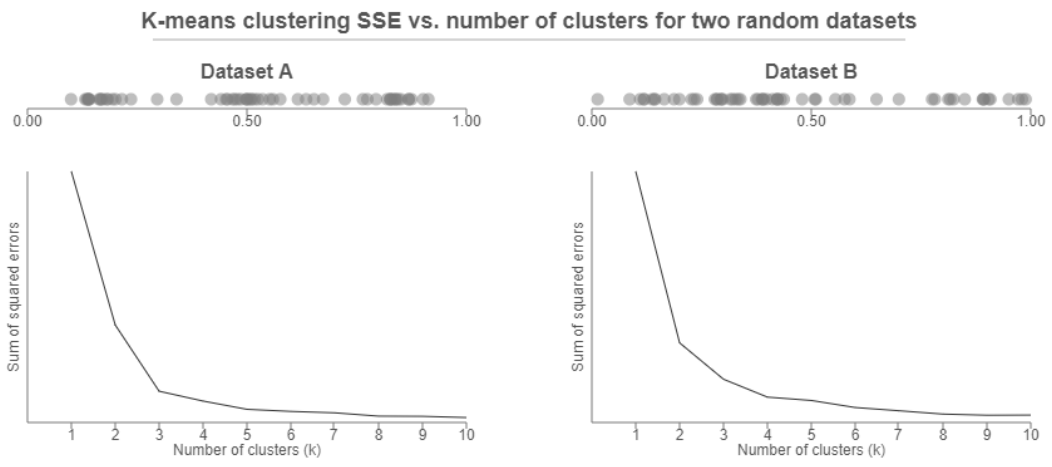
Let us implement this in Python using the **sklearn** library and our own function for calculating WSS for a range of values for k.

We obtain the following plot for WSS-vs-k for our dataset.



As expected, the **plot looks like an arm with a clear elbow at k = 3.**

Unfortunately, we do not always have such clearly clustered data. This means that the elbow may not be clear and sharp.



Source: bl.ocks.org/

For Dataset A, the elbow is clear at k = 3. However, this choice is ambiguous for Dataset B. We could choose k to be either 3 or 4.

In such an ambiguous case, we may use the Silhouette Method.

The Silhouette Method

The silhouette value measures how similar a point is to its own cluster (cohesion) compared to other clusters (separation).

The range of the Silhouette value is between +1 and -1. A **high value is desirable** and indicates that the point is placed in the correct cluster. If many points have a negative Silhouette value, it may indicate that we have created too many or too few clusters. The Silhouette Value **s(i)** for each data point **i** is defined as follows:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{ if } |C_i| > 1$$

and

$$s(i) = 0, \text{ if } |C_i| = 1$$

Note: s(i) is defined to be equal to zero if **i** is the only point in the cluster. This is to prevent the number of clusters from increasing significantly with many single-point clusters. Here, **a(i)** is the measure of similarity of the point **i** to its own cluster. It is measured as the average distance of **i** from other points in the cluster.

For each data point $i \in C_i$ (data point i in the cluster C_i), let

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j)$$

Similarly, **b(i)** is the measure of dissimilarity of **i** from points in other clusters.

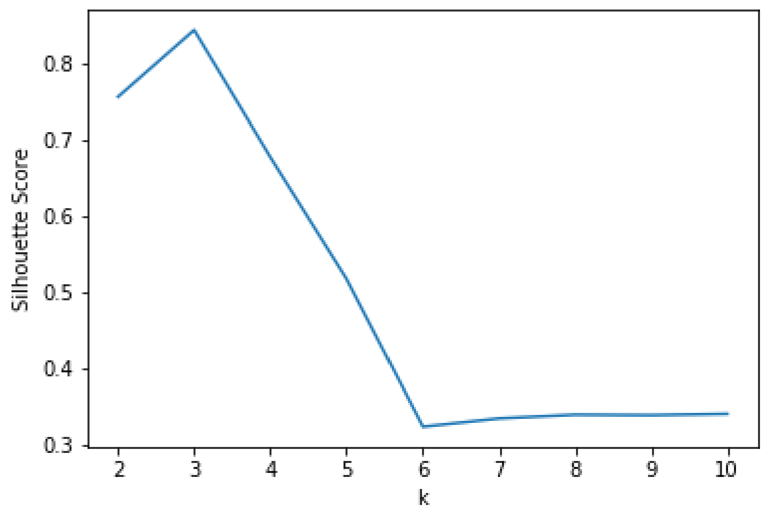
For each data point $i \in C_i$, we now define

$$b(i) = \min_{i \neq j} \frac{1}{|C_j|} \sum_{j \in C_j} d(i, j)$$

d(i, j) is the distance between points **i** and **j**. Generally, **Euclidean Distance** is used as the distance metric. The Silhouette score can be easily calculated in Python using the metrics module of the **sklearn** library.

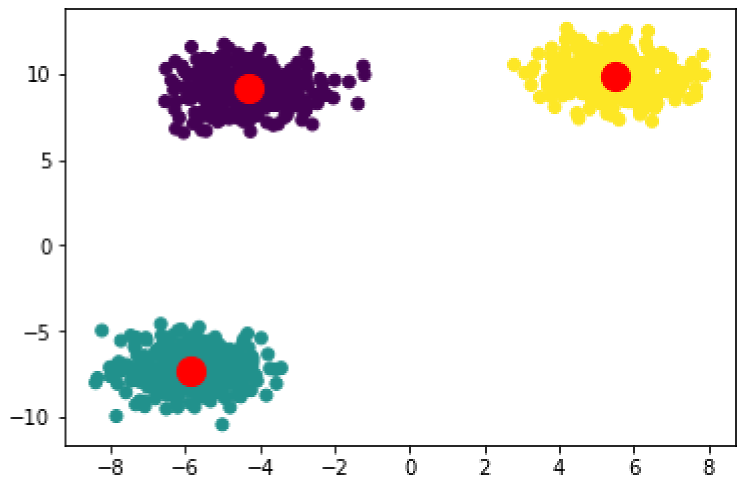
I mentioned before that a high Silhouette Score is desirable. The Silhouette Score reaches its **global maximum at the optimal k**. This should ideally appear as a peak in the Silhouette Value-versus-k plot.

Here is the plot for our own dataset:



There is a clear peak at k = 3. Hence, it is optimal.

Finally, the data can be **optimally** clustered into 3 clusters as shown below.



The Elbow Method is more of a decision rule, while the Silhouette is a metric used for validation while clustering. Thus, it can be used in combination with the Elbow Method. Therefore, the Elbow Method and the Silhouette Method are not alternatives to each other for finding the optimal K. Rather they are tools to be used together for a more confident decision.

d) Explain the necessity for scaling/standardisation before performing Clustering.

Standardization (sometimes called data normalization or feature scaling) refers to the process of rescaling the values of the variables in your data set so they share a common scale. Often performed as a pre-processing step, particularly for cluster analysis, standardization may be important if you are working with data where each variable has a different unit (e.g., inches, meters, tons and kilograms), or where the scales of each of your variables are very different from one another (e.g., 0-1 vs 0-1000).

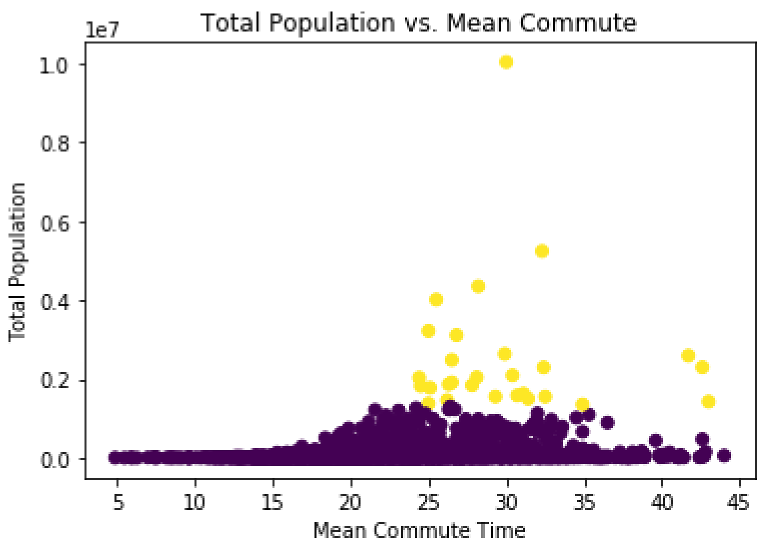
The reason this importance is particularly high in cluster analysis is because groups are defined based on the distance between points in mathematical space.

When you are working with data where each variable means something different, (e.g., age and weight) the fields are not directly comparable. One year is not equivalent to one pound, and may or may not have the same level of importance in sorting a group of records. In a situation where one field has a much greater range of value than another (because the field with the wider range of values likely has greater distances between values), it may end up being the primary driver of what defines clusters. Standardization helps to make the relative weight of each variable equal by converting each variable to a unitless measure or relative distance.

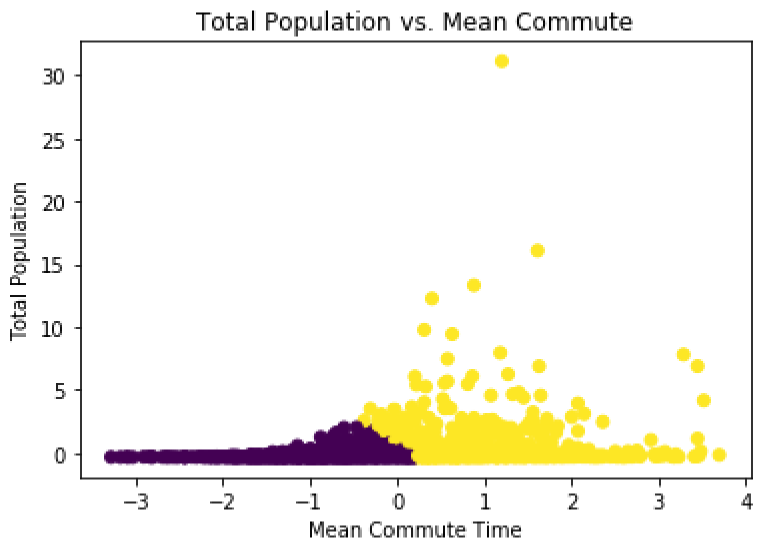
What follows is a couple examples demonstrating how standardization may impact a clustering solution, using the 2015 US Census Demographic Dataset, downloaded from Kaggle. This dataset includes different demographic variables for counties in the United States, including population, race, income, poverty, commute distance, commute method, as well as variables describing employment.

In our first example, we are interested in performing cluster analysis on Total Population and Mean Commute Time. We would like to use these two variables to split all of the counties into two groups. The units (number of people vs. minutes) and the range of values (85 - 10038388 people vs. 5 - 45 minutes) of these attributes are very different. It is also worth noting that Total Population is a sum, and Mean Commute Time is an average.

When we create clusters with the raw data, we see that Total Population is the primary driver of dividing these two groups. There is an apparent population threshold used to divide the data into two clusters:

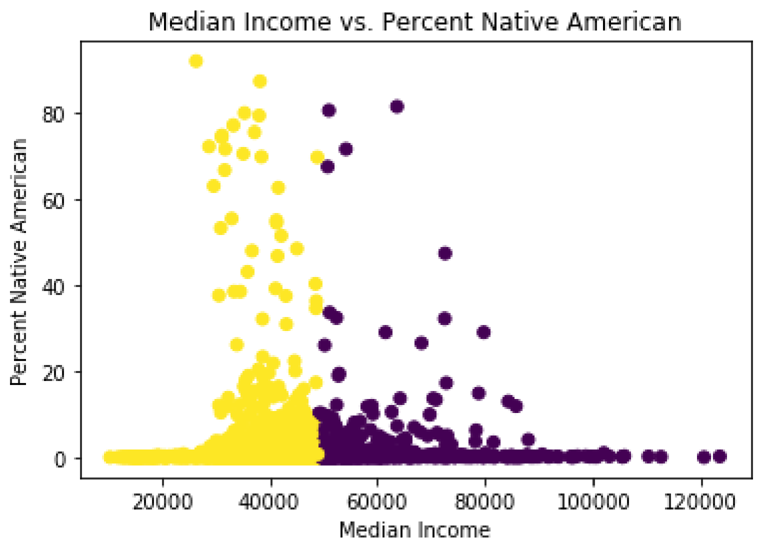


However, after standardization, both Total Population and Mean Commute seem to have an influence on how the clusters are defined.

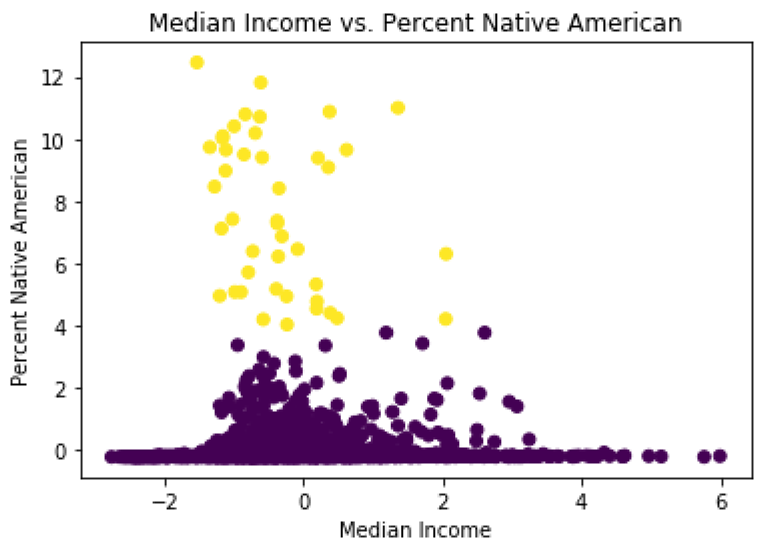


In this next example, we are interested in clustering on Median Income and Percent of the Population that is Native American (by county). Median Income is measured in dollars and represents the "middle" income for a household in a given county, and Native American is a percentage of the total population for that county. Again, the units and ranges of these variables are very different from one another.

When we perform cluster analysis with these two variables without first standardizing, we see that the clusters are primarily split on Income. Income, being measured in dollars, has greater separation in points than percentages, therefore it is the dominant variable.



When we standardize the data prior to performing cluster analysis, the clusters change. We find that with more equal scales, the Percent Native American variable more significantly contributes to defining the clusters.



Standardization prevents variables with larger scales from dominating how clusters are defined. It allows all variables to be considered by the algorithm with equal importance.

e) Explain the different linkages used in Hierarchical Clustering.

Single-Linkage

Single-linkage (nearest neighbour) is the shortest distance between a pair of observations in two clusters. It can sometimes produce clusters where observations in different clusters are closer together than to observations within their own clusters. These clusters can appear spread-out.

Complete-Linkage

Complete-linkage (farthest neighbour) is where distance is measured between the farthest pair of observations in two clusters. This method usually produces tighter clusters than single-linkage, but these tight clusters can end up very close together. Along with average-linkage, it is one of the more popular distance metrics.

Average-Linkage

Average-linkage is where the distance between each pair of observations in each cluster are added up and divided by the number of pairs to get an average inter-cluster distance. Average-linkage and complete-linkage are the two most popular distance metrics in hierarchical clustering.

Centroid-Linkage

Centroid-linkage is the distance between the centroids of two clusters. As the centroids move with new observations, it is possible that the smaller clusters are more similar to the new larger cluster than to their individual clusters causing an inversion in the dendrogram. This problem doesn't arise in the other linkage methods because the clusters being merged will always be more similar to themselves than to the new larger cluster.