Introduction to Clustering Analysis

Algorithms:

- K-Means
- Agglomerative Hierarchical

Types of Clustering

$$\begin{tikzcd}
 & \text{Types of Clustering} \arrow[dl] \arrow[dr] & \\
\text{Flat (K-means)} & & \text{Hierarchical (Agglomerative, Divisive)}
\end{tikzcd}$$

K-Means

Clustering

- In unsupervised learning, we only have access to the features of an instance and not its label. The goal is to discover hidden patterns in the dataset.

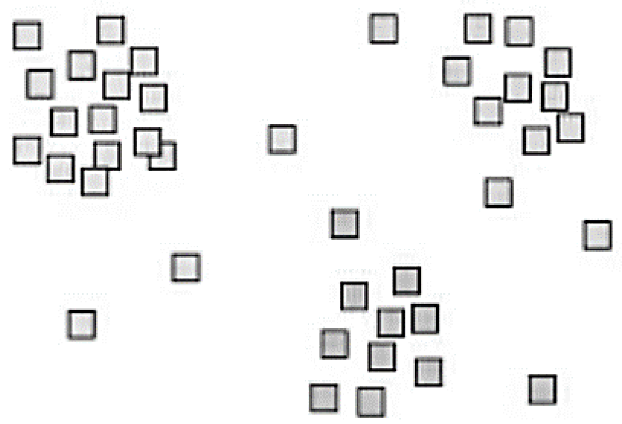
- Clustering goal: to aggregate instances into a few cohesive groups (Ex: segment customers into different groups based on their shopping. Detect outliers by detecting instances that are not similar to any known clusters).

K-Means Clustering

- Unsupervised model for identifying groups of data points without prior knowledge of existing classes.
- K-Means clustering splits the dataset into k number of clusters.
- Must first decide on the number of clusters (k) we want. Next, the centroids of the clusters must be chose **randomly** and the following steps performed iteratively:
  a) assign each training instance to a cluster based on its proximity to the cluster's centroid.
  b) compute the mean of all the data points assigned to a cluster and use that to update the centroid.
  c) After updating the centroids of all the clusters, reassign the points and repeat the updating and assigning steps until all the centroids stop changing.

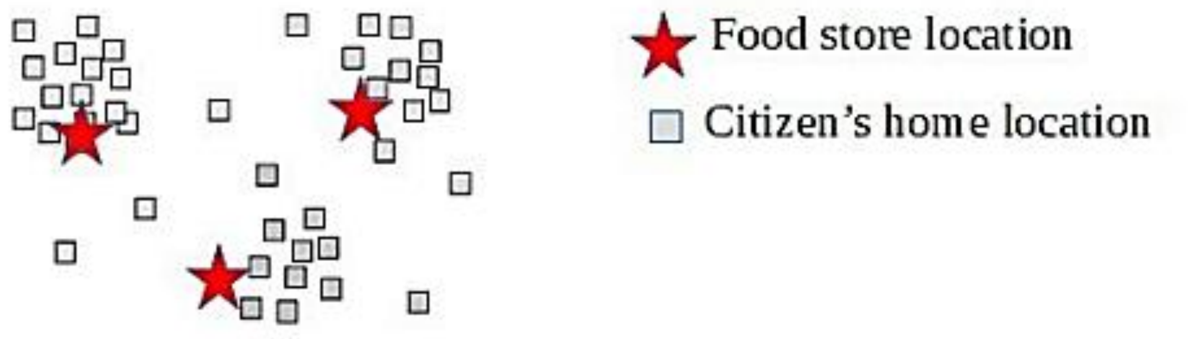
Example 1

- Citizens' houses in a city



Problem:

- Establish K food stores in a city.
- Each food store serves an area within a certain range.
- The food store locations correspond to the K clusters' centers (centroids).

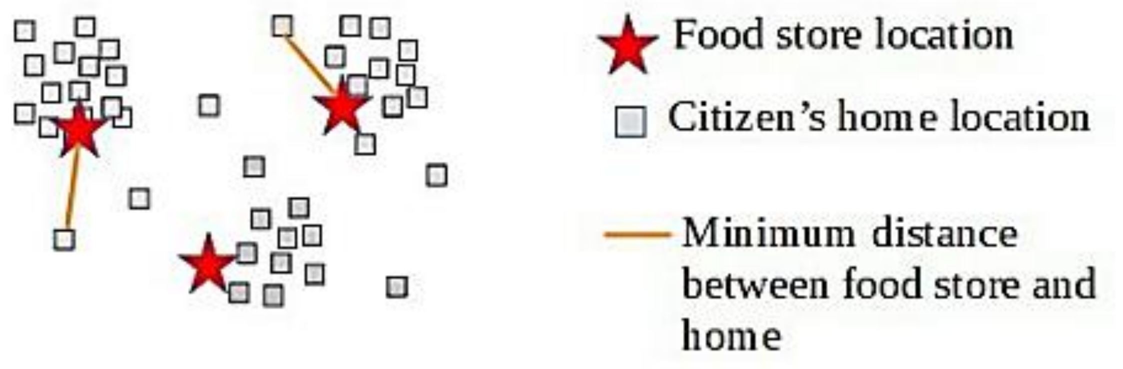


Procedure

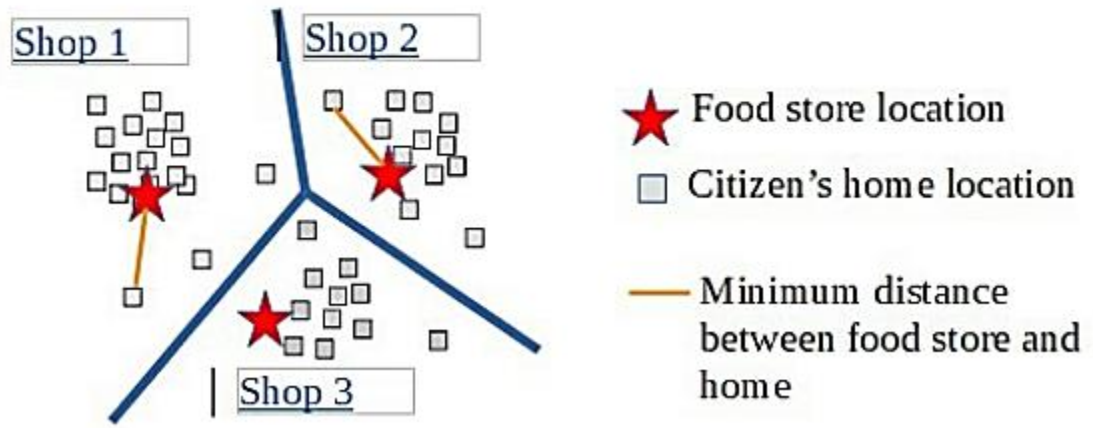
1. Initially choose a random location for the K = 3 food stores.
2. Calculate all the distances between first home and the K = 3 food stores. Choose the store which has the minimum distance.
3. Assign the house to this store (for example, the house is closer to Shop 2 so it becomes the first element of this cluster (2nd cluster).
4. Repeat the process (steps 2 and 3) for all the houses.

End of the Process:

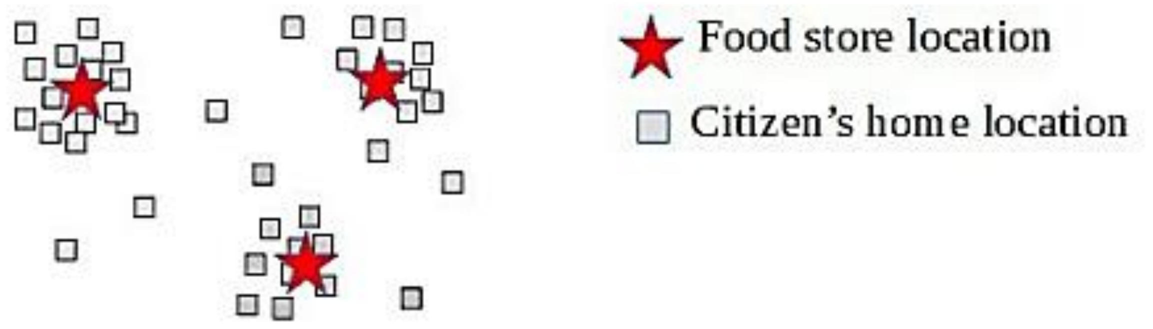
- Each citizen house belongs to their cluster (closest food store).
- 3 clusters are formed, based on the minimum location that each citizen has from each food store.



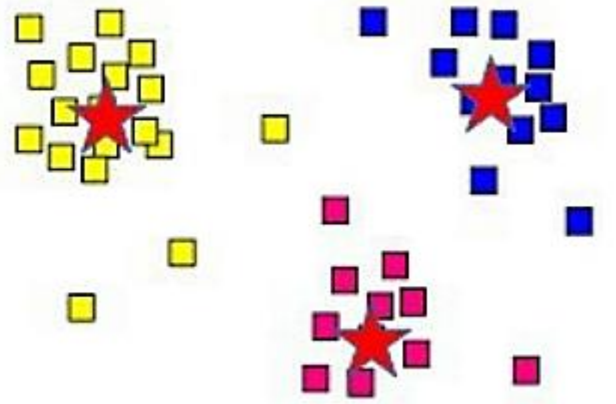
- The shop locations are not exactly in the center of each cluster. Find the mean value of the coordinates of all the data points that belong in each cluster.
- The means for Shop 1 cluster, Shop 2 cluster, Shop 3 cluster are calculated. We move the 3 centroids to the mean location.



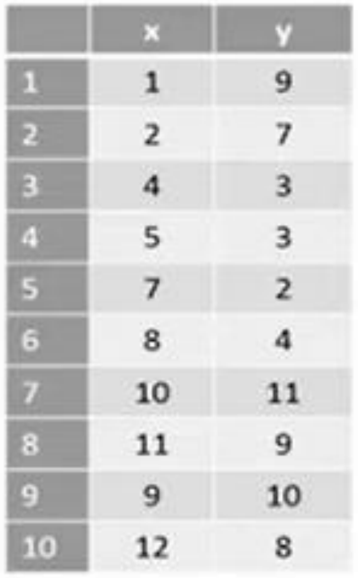
- The new location of the cluster centroids (shops) may change the current situation: a citizen may realize that there is a shop closer than the one he used to shop before.
- The process of finding the closest shop for all the citizens and form the clusters is repeated (steps 2 – 4).
- The process ends when we reach a satisfactory result.



Final Clusters!



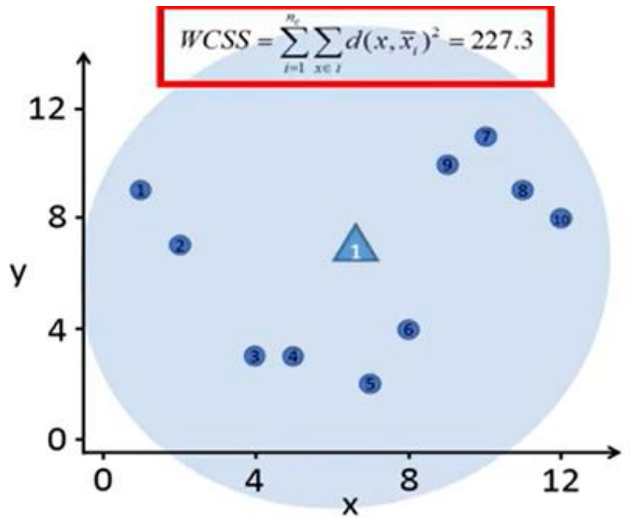
Inertia - WCSS



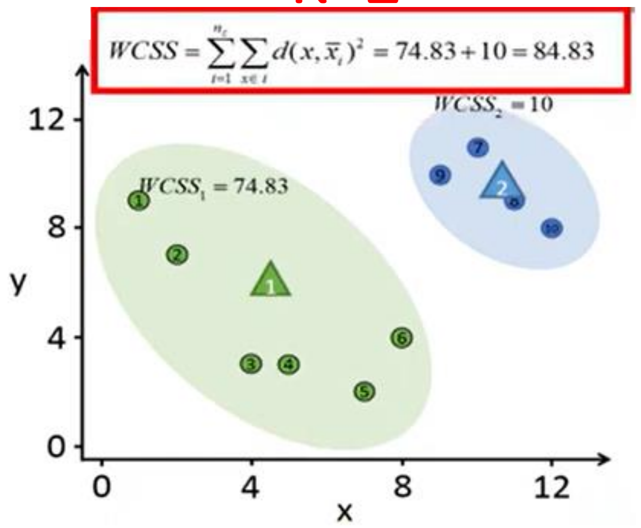
	x	y
1	1	9
2	2	7
3	4	3
4	5	3
5	7	2
6	8	4
7	10	11
8	11	9
9	9	10
10	12	8

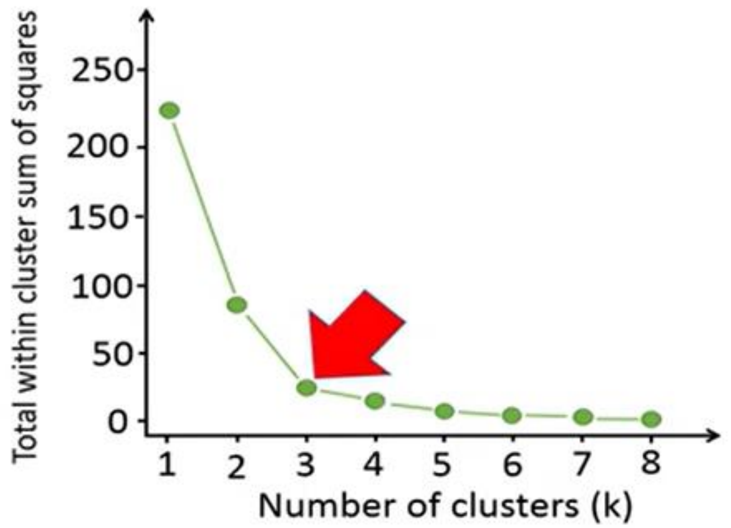
$$WCSS = \sum_{i=1}^{n_c} \sum_{x \in i} d(x, \bar{x}_i)^2$$

K=1

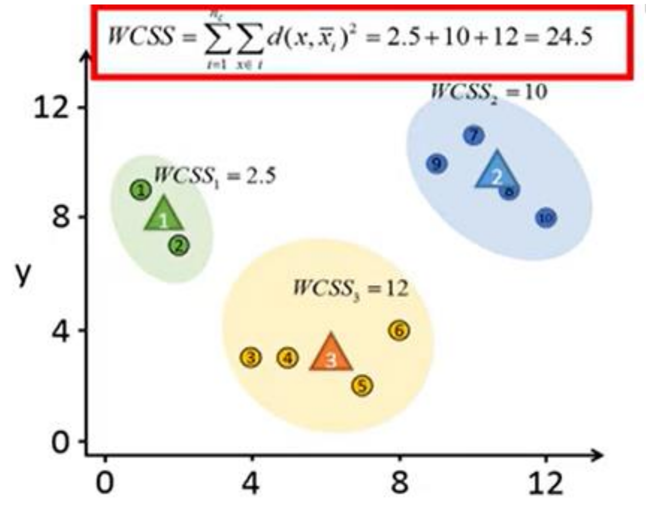


K=2

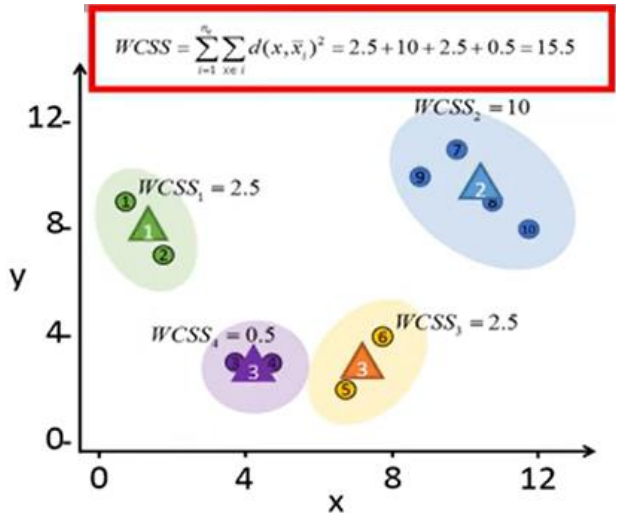




K=3



K=4



Example 1

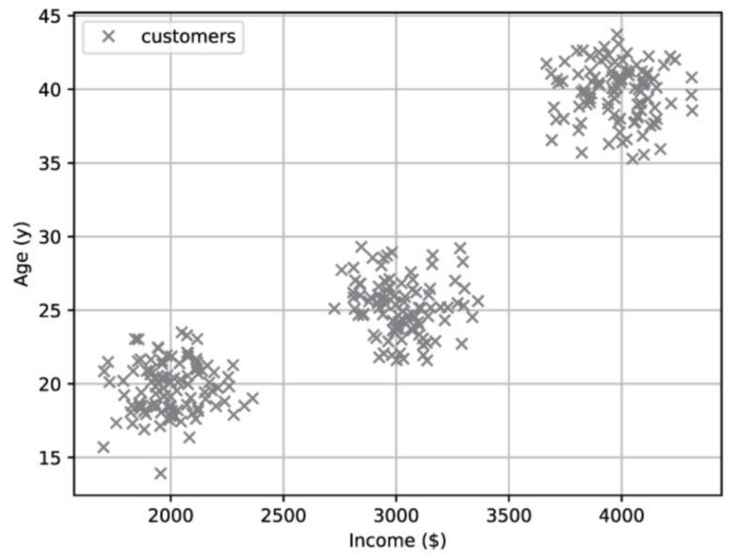
Find the Personas

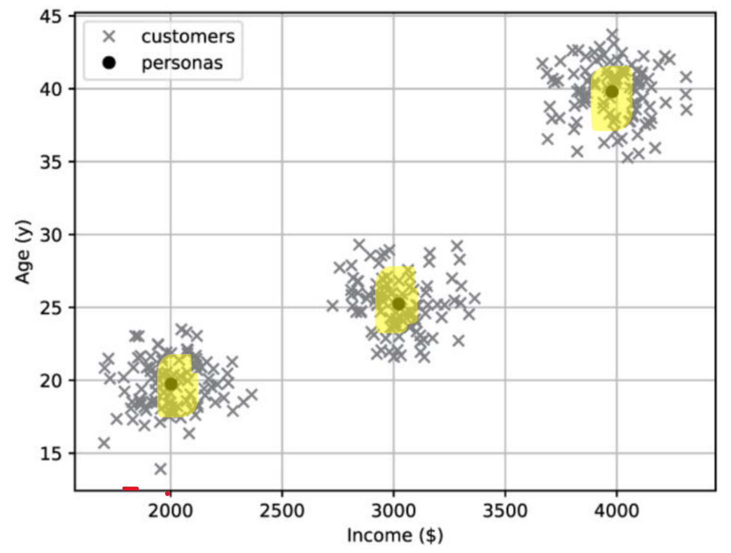
- Identify 3 types of personas with different types of incomes and ages (clustering).

- Given the data sets and an integer k, the K-Means algorithm finds k clusters of data such that the difference between the center of a cluster (called the centroid) and the data in the cluster is minimal.

- This is equivalent to finding the different personas.

- The K-Means algorithm extracts k clusters from multidimensional data. Behind the scenes, the algorithm iteratively recomputes cluster centers and reassigns each data value to its closest cluster center until it finds the optimal clusters.





Example 2

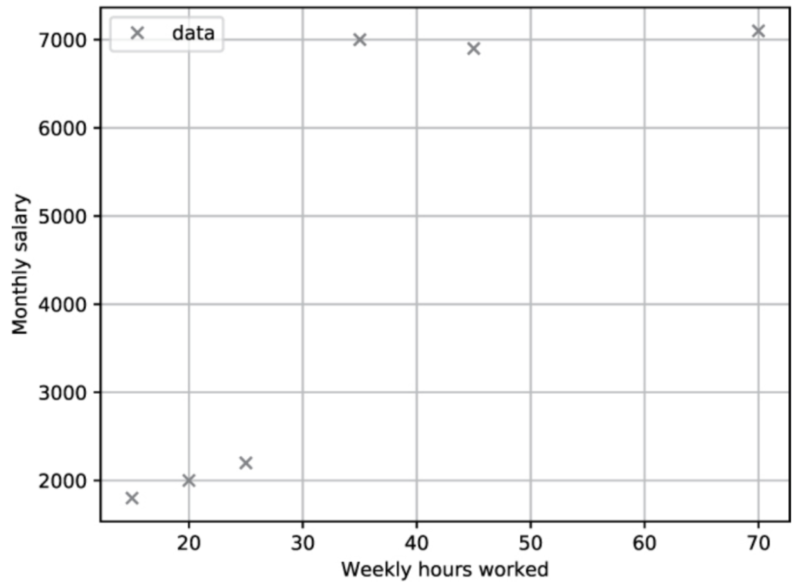
- We generate some 2D data (age and salary) for different individuals.

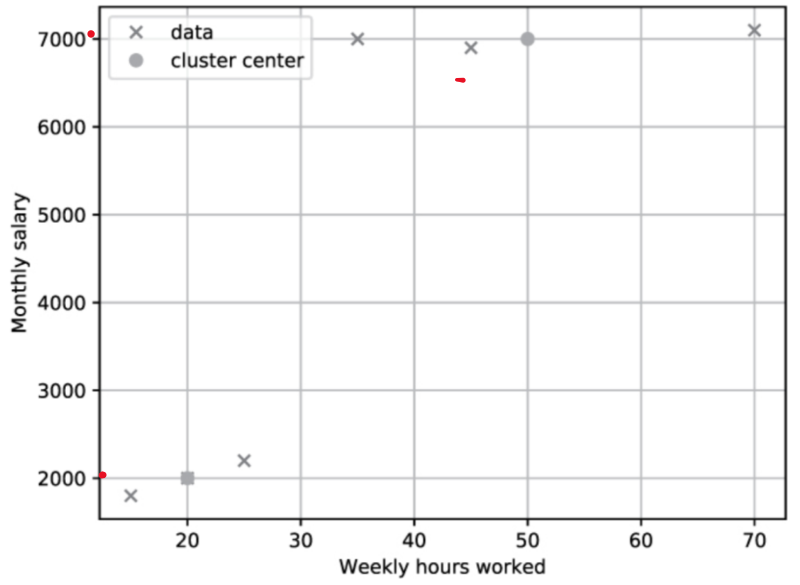
```
## Dependencies
from sklearn.cluster import KMeans
import numpy as np


## Data (Work (h) / Salary ($))
X = np.array([[35, 7000], [45, 6900], [70, 7100],
              [20, 2000], [25, 2200], [15, 1800]])


kmeans = KMeans(n_clusters=2).fit(X)


## Result & puzzle
cc = kmeans.cluster_centers_
print(cc)
```





- The K-Means algorithm identifies two separate cluster centers are:

(20, 2000) and (50, 7000)

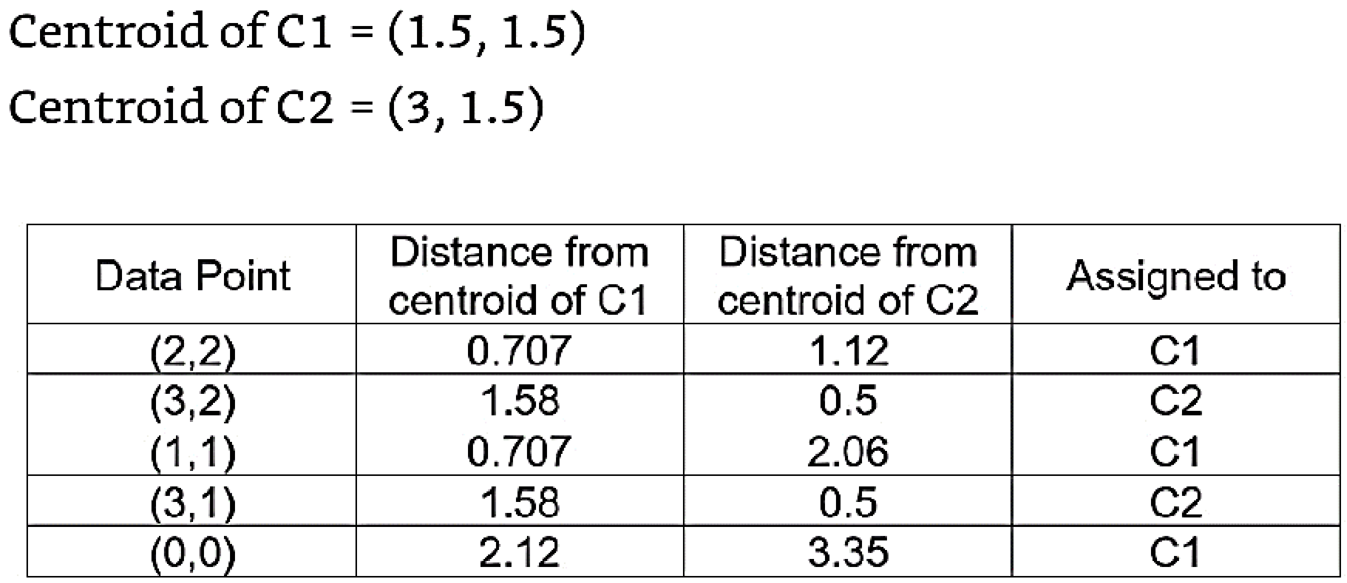
- The clusters correspond to two idealized employee personas:
- 1st Persona: works for 20 hours a week and earns $2000 per month
- 2nd Persona: works for 50 hours a week and earns $7000 per month.

- Clusters are not always ideal for finding similar data items.
- Many data sets do not show a clustered behavior. We can still leverage the distance information for machine learning and prediction.

- One answer is the **K-Nearest Neighbors algorithm.**

Example 3

- Let's use a small dataset with five points: (2,2), (3,2), (3,1), (1,1), (0,0).
- Say we want to cluster the points into 2 clusters so we need to select 2 centroids randomly. Suppose we choose (1.5, 1.5) and (3, 1.5) as the centroids and name the clusters C1 and C2.
- We assign each data point to the cluster with the nearest centroid. To determine how close a point is to a centroid, we commonly use the Euclidean distance formula. The table below shows the distances of each point from the two centroids.



Centroid of C1 = (1.5, 1.5)

Centroid of C2 = (3, 1.5)

Data Point	Distance from centroid of C1	Distance from centroid of C2	Assigned to
(2,2)	0.707	1.12	C1
(3,2)	1.58	0.5	C2
(1,1)	0.707	2.06	C1
(3,1)	1.58	0.5	C2
(0,0)	2.12	3.35	C1

- Based on the distances, we assign (2, 2), (1,1), and (0, 0) to C1, and (3, 2) and (3, 1) to C2.
- We then calculate the mean of the data points in each cluster and use that to update the centroids.

New Centroid of C1

$$= \left(\frac{2+1+0}{3}, \frac{2+1+0}{3}\right) = (1, 1)$$

New Centroid of C2

$$= \left(\frac{3+3}{2}, \frac{2+1}{2}\right) = (3, 1.5)$$

- Most of the points remain in the same cluster except for (2, 2).

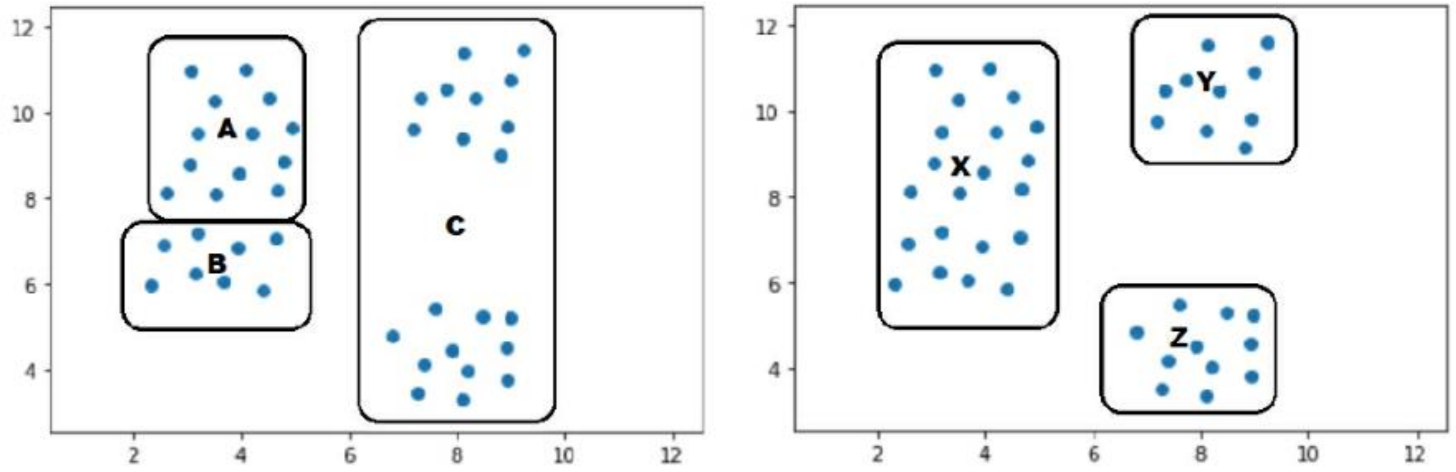
Data Point	Distance from new centroid of C1	Distance from new centroid of C2	Assigned to
(2,2)	1.41	1.12	C2
(3,2)	2.24	0.5	C2
(1,1)	0	2.06	C1
(3,1)	2	0.5	C2
(0,0)	1.41	3.35	C1

- We recalculate the mean of the data points in the two clusters and update the centroids. We repeat the process of updating the centroids and reassigning data points until the clusters no longer change. When that happens, the algorithm has converged and ends.

Different Results for Different Centroid Initialization and Choice of k

- As mentioned, we first need to choose the initial centroids.

- **Centroid initialization:** choosing the initial centroids. Poor initialization can lead to suboptimal solutions. Library sklearn implements the k-means algorithm and takes care of the centroid initialization using a technique called *kmeans++* which leads to good centroid initialization.



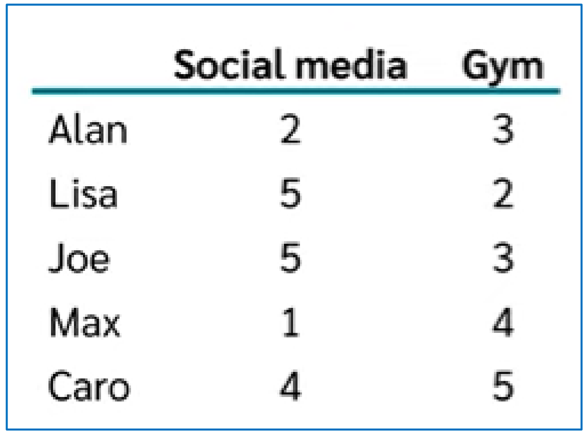
- After initial centroid initialization, we need to decide on the <u>number of clusters</u> by plotting the dataset on a scatter plot (only works if the dataset is simple and can be easily plotted).

- *Elbow method:* method involves running the k-means algorithm multiple times with different numbers of clusters and calculating the **inertia.** The goal is to find an optimal number of clusters where the decrease in inertia becomes less significant with added clusters.

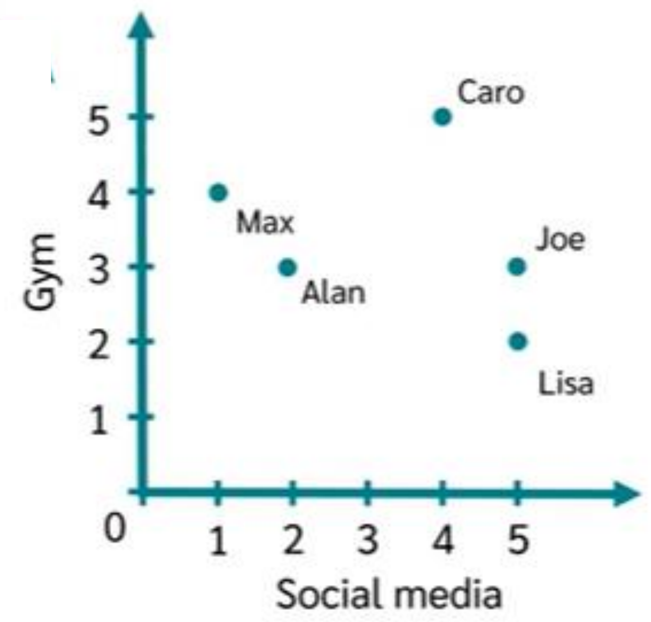
Hierarchical Clustering

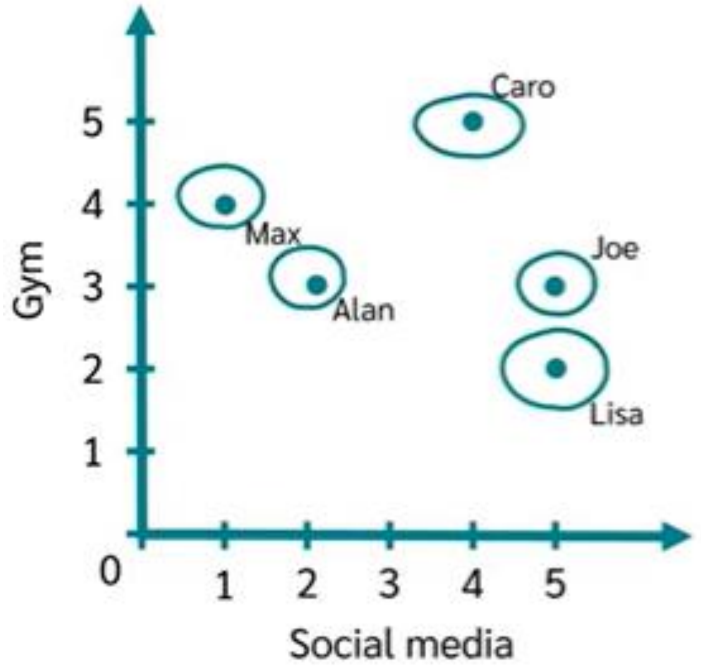
Example (courtesy of DataTab: https://www.youtube.com/watch?v=0m-rs2M7K-Y)

- Agglomerative Clustering



	Social media	Gym
Alan	2	3
Lisa	5	2
Joe	5	3
Max	1	4
Caro	4	5



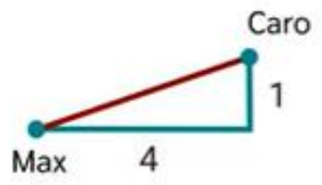


Distance Function

	Social media	Gym
Alan	2	3
Lisa	5	2
Joe	5	3
Max	1	4
Caro	4	5

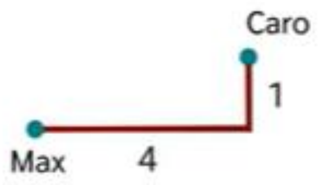
Euclidean distance

$$d = \sqrt{4^2 + 1^2} = 3{,}162$$



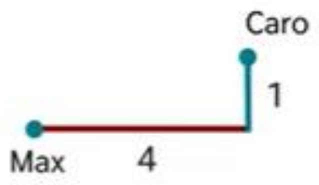
Manhattan distance

$$d = 4 + 1 = 5$$



Maximum distance

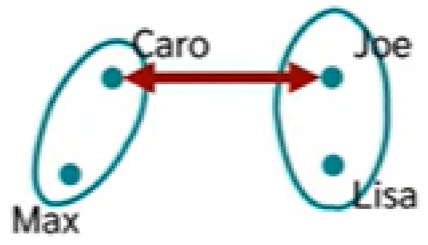
$$d = max(4\,,1) = 4$$



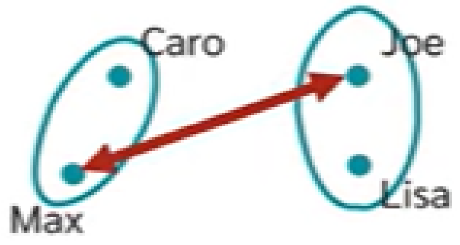
Linkage Function

- Linkage defines the merging criteria to compute the distance between sets of the observation data.
- Examples: "ward", "complete", "average", "single" methods.
- HC uses the linkage distance to recursive merge pairs of clusters of sample data.

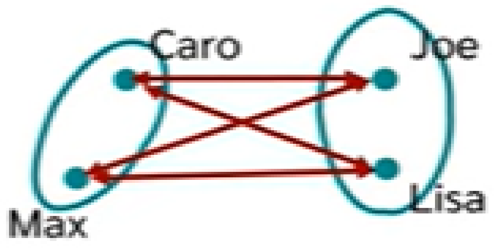
Single Linkage: distance between the closest elements in the clusters.



Complete Linkage: distance between the most distant elements in the clusters.

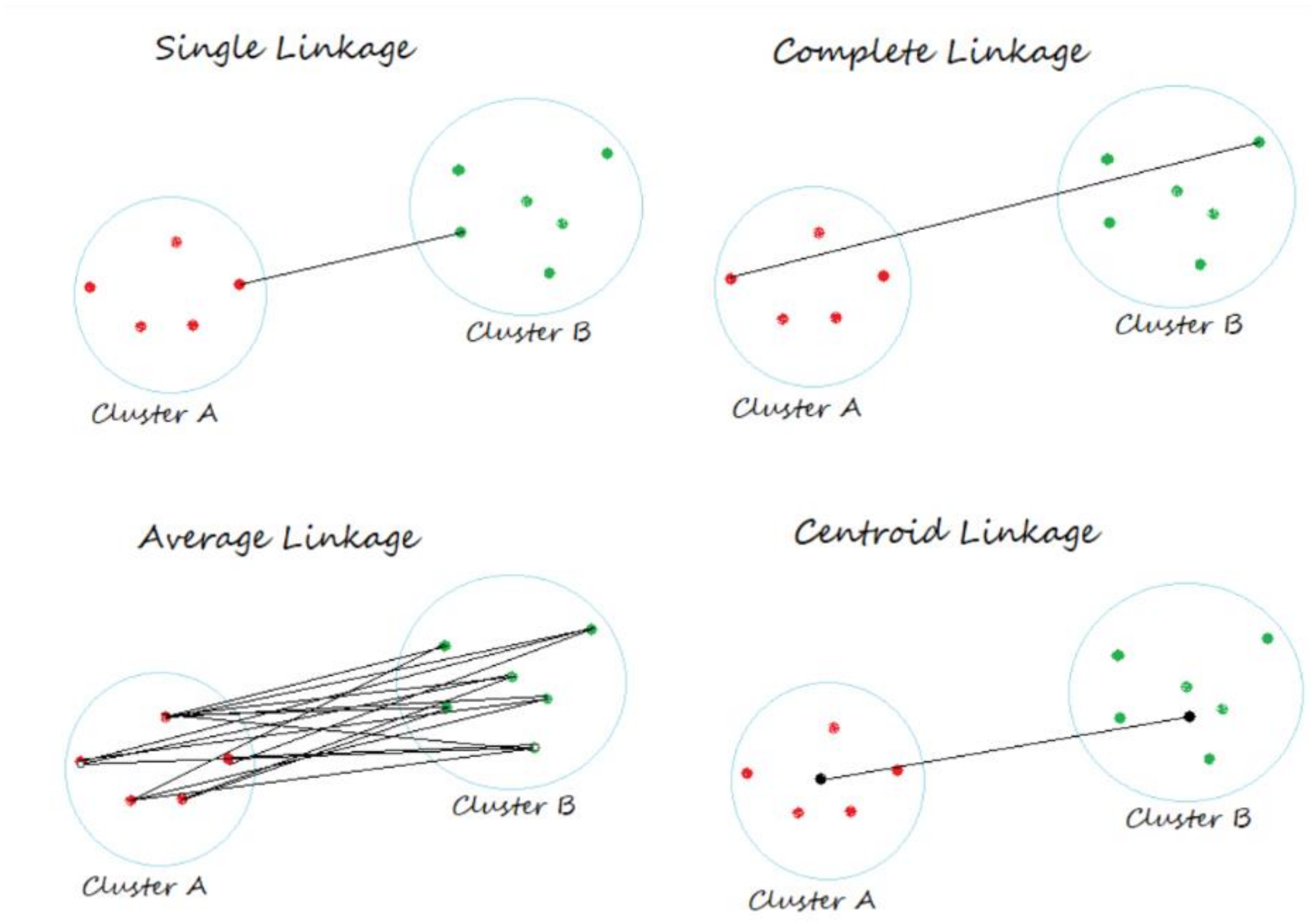


Average Linkage: average distance of all pairwise distances in the clusters.



Ward Linkage

Linkage Function (con'td)

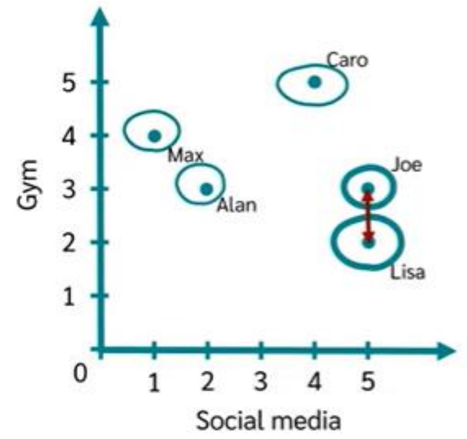


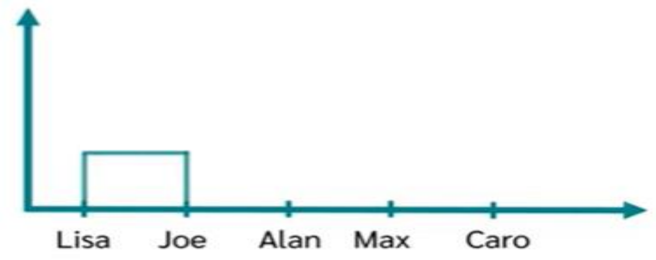
Distance Matrix

Scatterplot

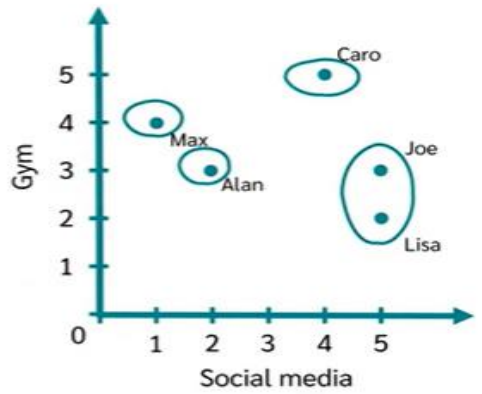
Dendrogram

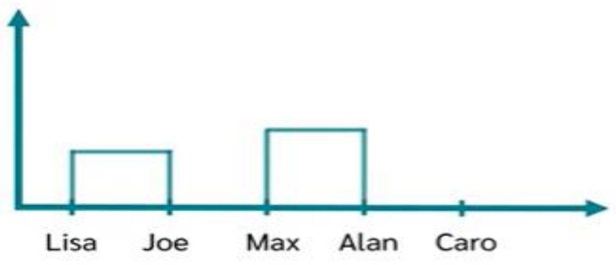
	Alan	Lisa	Joe	Max	Caro
Alan	0				
Lisa	3,16	0			
Joe	3,00	1,00	0		
Max	1,41	4,47	4,12	0	
Caro	2,83	3,16	2,24	3,16	0



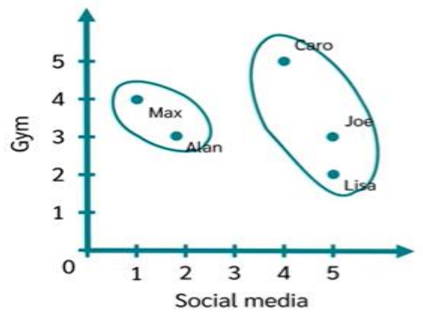


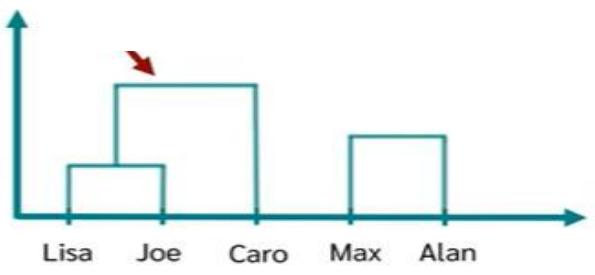
	Alan	Lisa, Joe	Max	Caro
Alan	0			
Lisa, Joe	3,00	0		
Max	1,41	4,12	0	
Caro	2,83	2,24	3,16	0



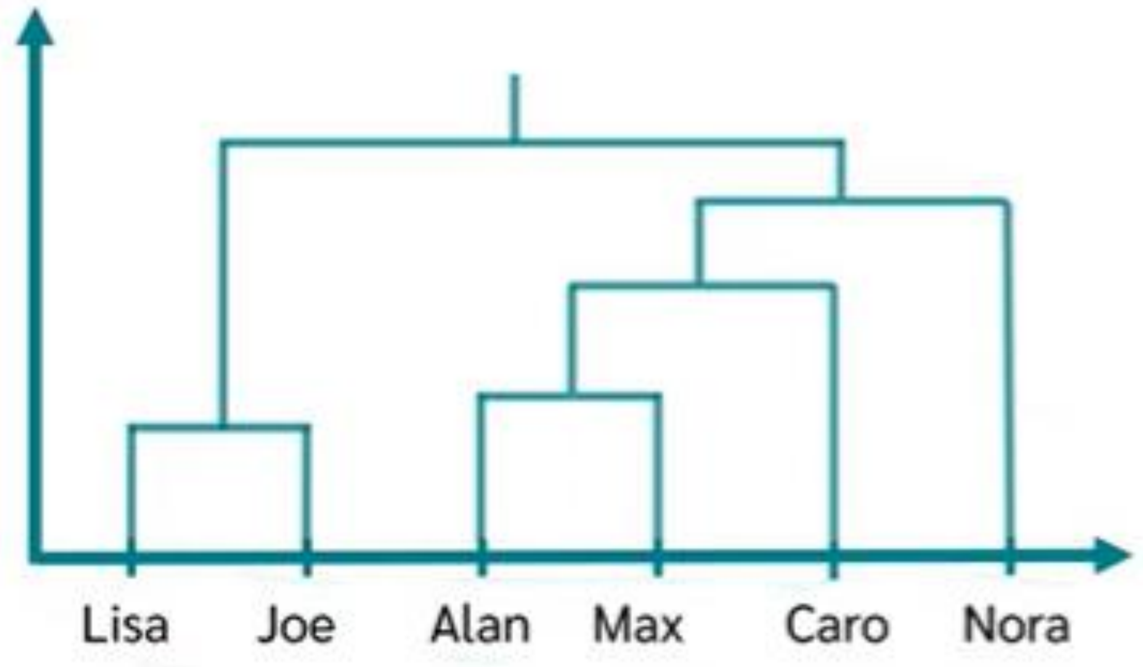


	Lisa, Joe	Max, Alan	Caro
Lisa, Joe	0		
Max, Alan	3,00	0	
Caro	2,24	2,83	0





Final Dendrogram



Python Code Example – creating the dendrogram

```
from scipy.cluster.hierarchy import dendrogram, linkage

dendos = linkage(features, 'single')

 annots = range(1, 11)

dendrogram(dendos, orientation='top', 10. labels=annots,
distance_sort='descending', show_leaf_counts=True)

plt.show()
```

Python Code Example: creating the model

```
from sklearn.cluster import AgglomerativeClustering

# training agglomerative clustering model

hc_model = AgglomerativeClustering(n_clusters=2,
affinity='euclidean', linkage='ward')

 hc_model.fit_predict(features)
```

Questions?