How does k-means clustering work?

The goal of the k-means algorithm is to partition the data into k groups based on feature similarities. K is a predefined property of a k-means clustering model. Each of the k clusters are specified by a centroid (center of a cluster) and each data sample belongs to the cluster with the nearest centroid. During training, the algorithm iteratively updates the k centroids based on the data provided. Specifically, it involves the following steps:

1. **Specifying k**: The algorithm needs to know how many clusters to generate as an end result.
2. **Initializing centroids**: The algorithm starts with randomly selecting k samples from the dataset as centroids.
3. **Assigning clusters**: Now that we have k centroids, samples that share the same closest centroid constitute one cluster. K clusters are created as a result. Note that, **closeness** is usually measured by the **Euclidean distance**. Other metrics can also be used, such as the **Manhattan distance** and **Chebyshev distance**, which are listed in the following table:

Given two 2-dimension data points (x_1, y_1) and (x_2, y_2)

Distance metric	Calculation
Euclidean distance	$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$
Manhattan distance	$\lvert x_1 - x_2 \rvert + \lvert y_1 - y_2 \rvert$
Chebyshev distance	$\max(\lvert x_1 - x_2 \rvert, \lvert y_1 - y_2 \rvert)$

4. **Updating centroids**: For each cluster, we need to recalculate its center point, which is the mean of all the samples in the cluster. K centroids are updated to be the means of corresponding clusters. This is why the algorithm is called **k-means**.
5. **Repeating step 3 and 4**: It keeps repeating assigning clusters and updating centroids until the model is converged where the centroids stop moving or move small enough, or enough iterations have been taken.

The outputs of a trained k-means clustering model include the following:

- The cluster ID of each training sample, ranging from 1 to k
- K centroids, which can be used to cluster new samples—the new sample will belong to the cluster of the closest centroid

K-Mean Clustering

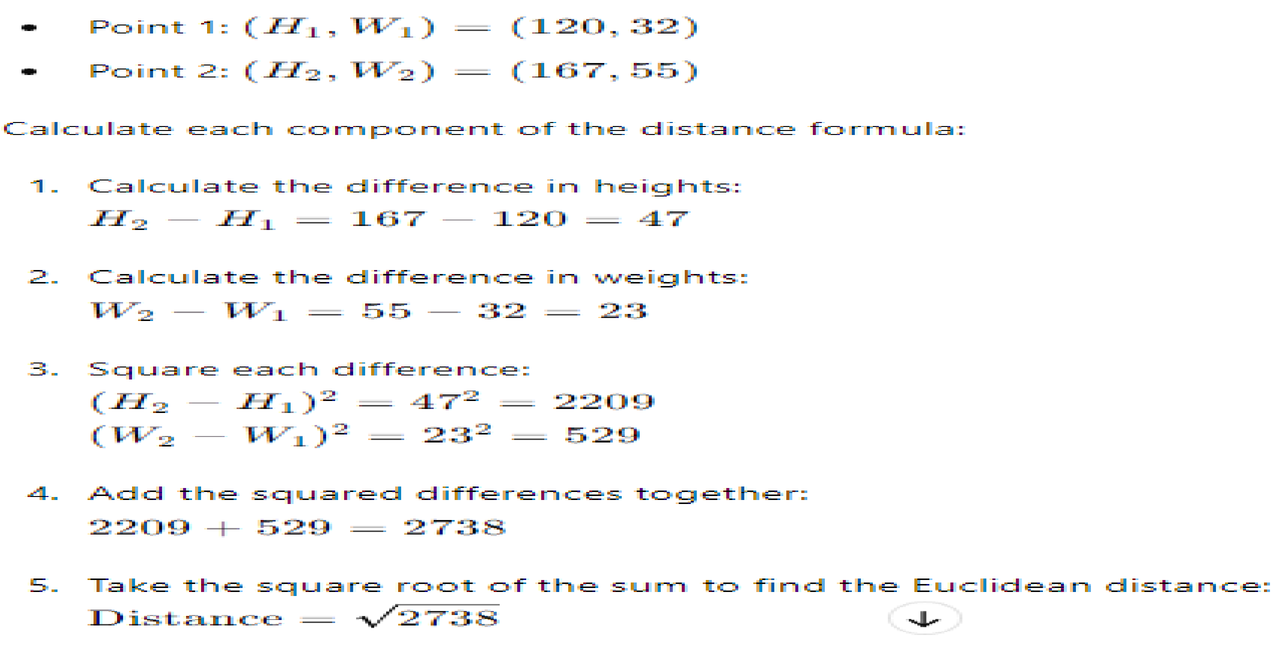
It allows grouping the data according to the existing similarities among them in k clusters, given as input to the algorithm.
Example: Let's imagine we have 5 objects (say 5 people) and for each of them we know two features (height and weight). We want to group them into k=2 clusters. Our dataset will look like this:

	Height(H)	Weight(W)
Person 1	167	55
Person 2	**120**	**32**
Person 3	**113**	**33**
Person 4	175	76
Person 5	108	25

First of all, we have to initialize the value of the centroids for our clusters.
For instance, let's choose **Person 2 and Person 3** as the two centroids c1 and c2, so that **c1=(120,32) and c2=(113,33).**
Now we compute the euclidian distance between each of the two centroids and each point in the data. If you did all the calculations, you should have come up with the following numbers:

	Distance of object from c1	**Distance of object from c2**
Person 1	52.3	58.3
Person 2	0	7.1
Person 3	7.1	0
Person 4	70.4	75.4
Person 5	13.9	9.4

- To find the Euclidean distance between the points (120, 32) and (167, 55), where the first coordinate represents height (H) and the second coordinate represents weight (W), we use the Euclidean distance formula:



- Point 1: $(H_1, W_1) = (120, 32)$
- Point 2: $(H_2, W_2) = (167, 55)$

Calculate each component of the distance formula:

1. Calculate the difference in heights:
   $H_2 - H_1 = 167 - 120 = 47$
2. Calculate the difference in weights:
   $W_2 - W_1 = 55 - 32 = 23$
3. Square each difference:
   $(H_2 - H_1)^2 = 47^2 = 2209$
   $(W_2 - W_1)^2 = 23^2 = 529$
4. Add the squared differences together:
   $2209 + 529 = 2738$
5. Take the square root of the sum to find the Euclidean distance:
   $\text{Distance} = \sqrt{2738}$

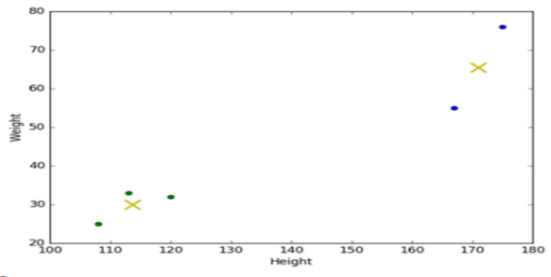
Given points: **Distance ≈ 52.3**

Therefore, the Euclidean distance between the points (120, 32) and (167, 55) is approximately 52.36 units. This distance represents the straight-line distance between the two points in the two-dimensional space defined by height and weight.
At this point, we will assign each object to the cluster it is closer to (that is taking the minimum between the two computed distances for each object). We can then arrange the points as follows:

Person 1 → cluster 1
Person 2 → cluster 1
Person 3 → cluster 2
Person 4 → cluster 1
Person 5→ cluster 2

Let's iterate, which means to **redefine the centroids by calculating the mean of the members o**f each of the two clusters.
So c'1 = ((167+120+175)/3, (55+32+76)/3) = **(154, 54.3)**
and c'2 = ((113+108)/2, (33+25)/2) = **(110.5, 29)**
Then, we calculate the distances again and re-assign the points to the **new centroids.**

We repeat this process until the centroids don't move anymore (or the difference between them is under a certain small threshold). The final result is given in the figure below. You can see the two different clusters labelled with two different colours and the position of the centroids, given by the crosses.



The k-means algorithm is implemented in the scikit-learn package.
The basic concept of k-means stands on mathematical calculations (means, euclidian distances). But what if our data is non-numerical or, in other words, categorical, e.g. ID code, date of birth etc., instead of their heights and weights. We could think of transforming our categorical values in numerical values and eventually apply kmeans. But beware: k-means uses numerical distances, so it could consider close two really distant objects that merely have been assigned two close numbers.

k-modes is an extension of k-means. Instead of distances it uses dissimilarities (that is, quantification of the total mismatches between two objects: the smaller this number, the more similar the two objects). And instead of means, it uses modes. A mode is a vector of elements that minimizes the dissimilarities between the vector itself and each object of the data. We will have as many modes as the number of clusters we required, since they act as centroids