

🧑 IBM - DATA SCIENCE - MACHINE LEARNING WITH PYTHON 🐍🧠

CLUSTERING

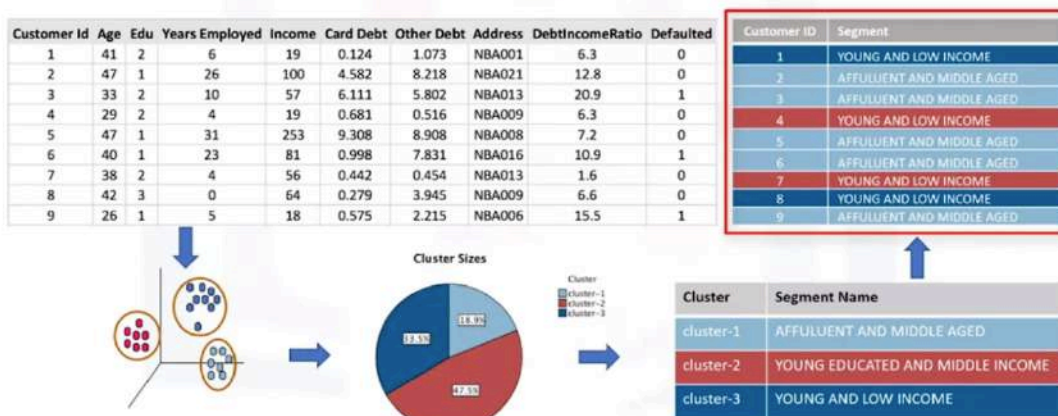
Intro to Clustering

La joda es separar en particiones datos que tengan características similares.

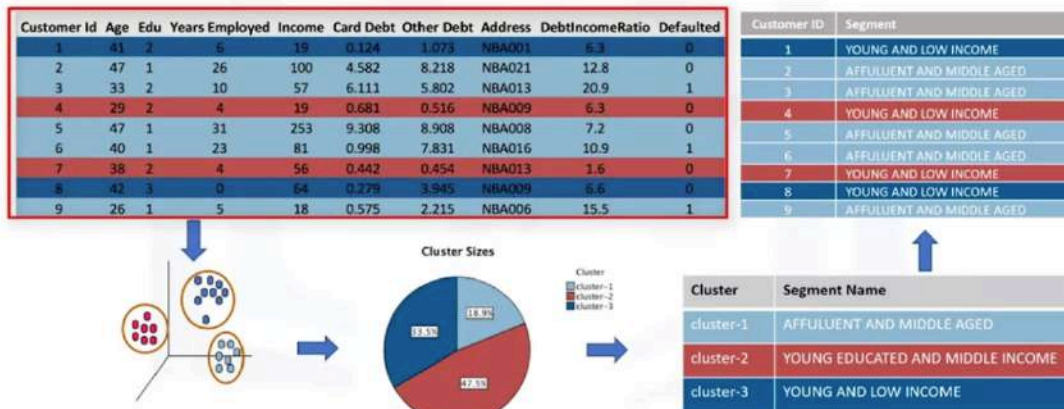
Necesitamos un analítical approach para separar a los wachs que tengan características similares.

Para esto es clave usar clustering

Clustering for segmentation



Clustering for segmentation



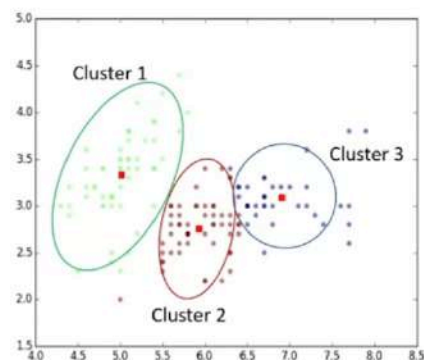
Imagínate que una vez que puedes meter a cada cliente en un cluster, puedes cruzarlo con lo que suele comprar ese cluster y vender mejores experiencias para cada segmento.

Definamos clustering: Encontrar clusters en un dataset de forma no supervisada... pero ¿qué es un cluster?

What is clustering?

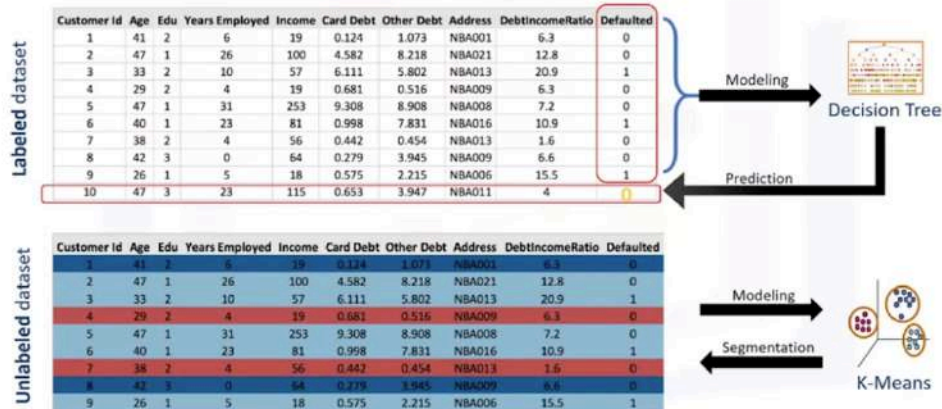
What is a cluster?

A group of objects that are **similar to other objects** in the cluster, and **dissimilar to data points** in other clusters.



La pregunta es... ¿cuál es la diferencia entre el clustering y classification?

Clustering Vs. classification



Clasificación necesita de data que este labeled. Clustering usa unlabeled. De forma generalizada, la clasificación es una forma supervisada de aprendizaje donde cada instancia de entrenamiento pertenece a una clase particular.

En clustering, por el otro lado, la data no esta labeled y el proceso es no supervisado. Por ejemplo, podemos usar K means para agrupar customers similares segun si comparten un atributo similar, como podria ser la edad.

Clustering applications

- **RETAIL/MARKETING:**
 - Identifying buying patterns of customers
 - Recommending new books or movies to new customers
- **BANKING:**
 - Fraud detection in credit card use
 - Identifying clusters of customers (e.g., loyal)
- **INSURANCE:**
 - Fraud detection in claims analysis
 - Insurance risk of customers

Clustering applications

- **PUBLICATION:**
 - Auto-categorizing news based on their content
 - Recommending similar news articles
- **MEDICINE:**
 - Characterizing patient behavior
- **BIOLOGY:**
 - Clustering genetic markers to identify family ties

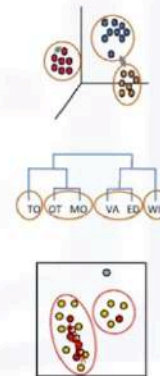
Why clustering?

- Exploratory data analysis
- Summary generation
- Outlier detection
- Finding duplicates
- Pre-processing step

Algunos algoritmos de clustering:

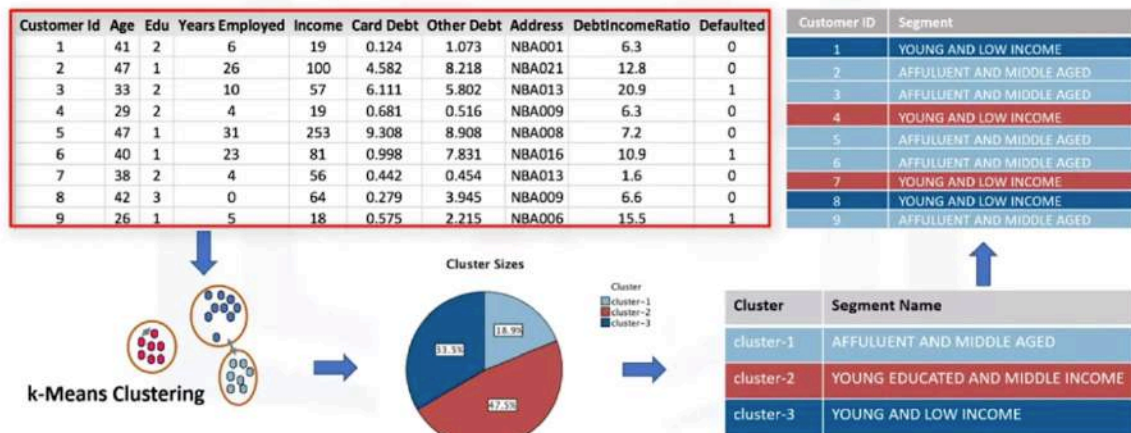
Clustering algorithms

- Partitioned-based Clustering
 - Relatively efficient
 - E.g. k-Means, k-Median, Fuzzy c-Means
- Hierarchical Clustering
 - Produces trees of clusters
 - E.g. Agglomerative, Divisive
- Density-based Clustering ★
 - Produces arbitrary shaped clusters
 - E.g. DBSCAN



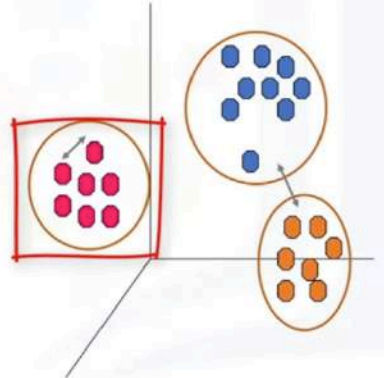
K-Means Algorithm

What is k-Means clustering?



k-Means algorithms

- Partitioning Clustering
- K-means divides the data into **non-overlapping** subsets (clusters) without any cluster-internal structure
- Examples within a cluster are very similar
- Examples across different clusters are very different



En vez de una métrica de similaridad, podemos usar una de disimilaridad. La distancia de los samples entre ellos nos puede servir para darle forma a los clusters.

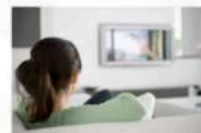
Kmeans intenta minimizar las distancias internas en el cluster y maximizar las externas. La pregunta es como podemos calcular la disimilaridad entre samples (o la distancia)? El bato aca vuelve a tirar la distancia euclidiana

1-dimensional similarity/distance



Customer 1

Age
54



Customer 2

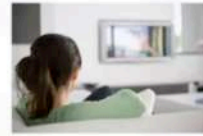
Age
50

$$\text{Dis}(x_1, x_2) = \sqrt{\sum_{i=0}^n (x_{1i} - x_{2i})^2}$$

Multi-dimensional similarity/distance



Customer 1		
Age	Income	education
54	190	3



Customer 2		
Age	Income	education
50	200	8

$$\text{Dis}(x_1, x_2) = \sqrt{\sum_{i=0}^n (x_{1i} - x_{2i})^2}$$
$$= \sqrt{(54 - 50)^2 + (190 - 200)^2 + (3 - 8)^2} = 11.87$$

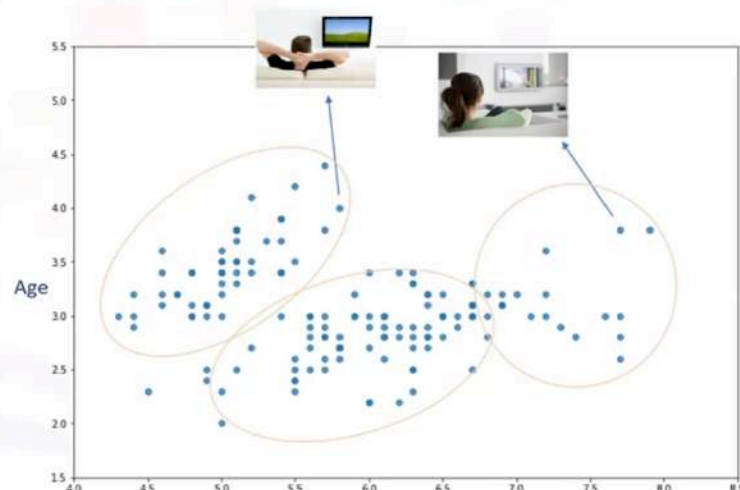
Dice que obvio, tenemos que normalizar nuestro feature set para que esto de bien. → Sino, diferencias de escalas o unidades podrian arruinar nuestras predicciones.

La distancia es la que va a darle forma a los clusters, psique es importante entender nuestro dataset y ver que tipo de distancia deberíamos usar. Ya que tambien podríamos usar nose, diferencias de cosenos, y otras.

Como funciona el clustering en kmeans?

How does k-Means clustering work?

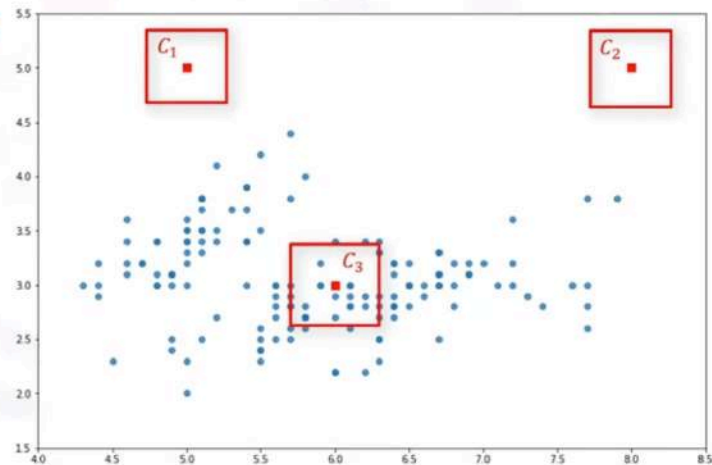
Customer ID	Age	Income
1	3	4
2	2	6
3	3.5	2
...



En el primer paso, deberíamos determinar el numero de clusters. El concepto clave del algoritmo kmeans es que toma de forma random un punto central para cada cluster. Esto significa que vamos a tener que inicializar K, que va a representar la cantidad de centroides elegidos de forma random. Elegir el valor de K es un problema difícil en K means, lo vamos a discutir mas adelante. Por ahora vamos con un ejemplo de K = 3

k-Means clustering – initialize k

1) Initialize $k=3$
centroids randomly

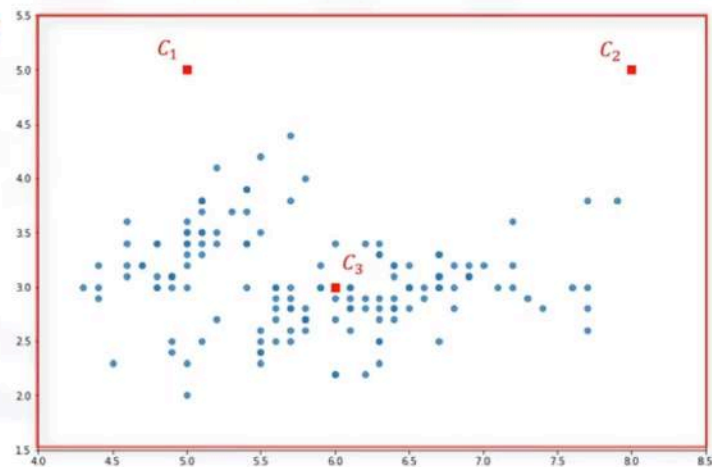


Los centroids se pueden elegir de forma random o bien elegir 3 puntos observados del dataset. En el ejemplo de arriba (y abajo) son elegidos de forma random.

k-Means clustering – initialize k

1) Initialize $k=3$
centroids randomly

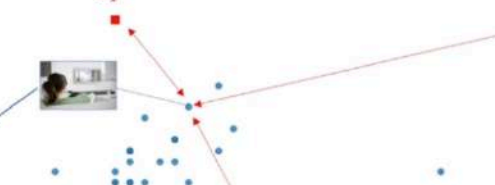
$C_1 = [8., 5.]$
 $C_2 = [5., 5.]$
 $C_3 = [6., 3.]$



Después del paso de inicialización, tenemos que asignar a cada uno de los customers al centro mas cercano. Para esto, calculamos la distancia de cada punto al centroide.

2) Distance calculation

C_1	C_2	C_3
$d(p_1, c_1)$	$d(p_1, c_2)$	$d(p_1, c_3)$
$d(p_2, c_1)$	$d(p_2, c_2)$	$d(p_2, c_3)$
$d(p_3, c_1)$	$d(p_3, c_2)$	$d(p_3, c_3)$
$d(p_4, c_1)$	$d(p_4, c_2)$	$d(p_4, c_3)$
$d(p, \dots, c_1)$	$d(p, \dots, c_2)$	$d(p, \dots, c_3)$
$d(pn, c_1)$	$d(pn, c_2)$	$d(pn, c_3)$
$d(p, \dots, c_1)$	$d(p, \dots, c_2)$	$d(p, \dots, c_3)$
$d(p, \dots, c_1)$	$d(p, \dots, c_2)$	$d(p, \dots, c_3)$
$d(pn, c_1)$	$d(pn, c_2)$	$d(pn, c_3)$
$d(p, \dots, c_1)$	$d(p, \dots, c_2)$	$d(p, \dots, c_3)$
$d(p, \dots, c_1)$	$d(p, \dots, c_2)$	$d(p, \dots, c_3)$
$d(pn, c_1)$	$d(pn, c_2)$	$d(pn, c_3)$
$d(p, \dots, c_1)$	$d(p, \dots, c_2)$	$d(p, \dots, c_3)$
$d(p, \dots, c_1)$	$d(p, \dots, c_2)$	$d(p, \dots, c_3)$
$d(pn, c_1)$	$d(pn, c_2)$	$d(pn, c_3)$

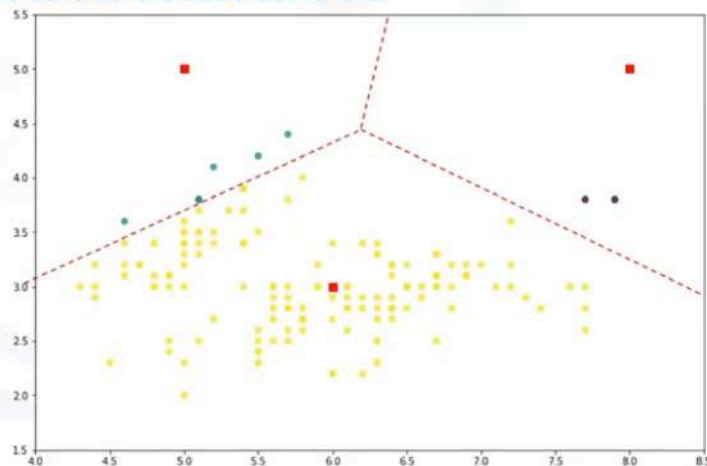


A scatter plot showing a distribution of blue data points in a 2D feature space. The x-axis ranges from 4.0 to 8.0, and the y-axis ranges from 1.5 to 5.5. Three clusters are identified and labeled with red text and arrows: C_1 (top-left), C_2 (top-right), and C_3 (bottom-center). Each cluster is represented by a red square. A small inset image in the top-left corner shows a person's head and shoulders, likely representing the input data for the clustering process.

De base podemos decir que no resulto muy bien nuestro algoritmo, y esto se debe a que seleccionamos los centroides de forma random:

3) Assign each point to the closest centroid

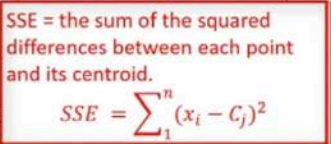
C_1	C_2	C_3
$d(p1, c1)$	$d(p1, c2)$	$d(p1, c3)$
$d(p2, c1)$	$d(p2, c2)$	$d(p2, c3)$
$d(p3, c1)$	$d(p3, c2)$	$d(p3, c3)$
$d(p4, c1)$	$d(p4, c2)$	$d(p4, c3)$
$d(p \dots, c1)$	$d(p \dots, c2)$	$d(p \dots, c3)$
$d(pn, c1)$	$d(pn, c2)$	$d(pn, c3)$
$d(p \dots, c1)$	$d(p \dots, c2)$	$d(p \dots, c3)$
$d(p \dots, c1)$	$d(p \dots, c2)$	$d(p \dots, c3)$
$d(pn, c1)$	$d(pn, c2)$	$d(pn, c3)$
$d(p \dots, c1)$	$d(p \dots, c2)$	$d(p \dots, c3)$
$d(pn, c1)$	$d(pn, c2)$	$d(pn, c3)$
$d(p \dots, c1)$	$d(p \dots, c2)$	$d(p \dots, c3)$
$d(pn, c1)$	$d(pn, c2)$	$d(pn, c3)$
$d(p \dots, c1)$	$d(p \dots, c2)$	$d(p \dots, c3)$
$d(pn, c1)$	$d(pn, c2)$	$d(pn, c3)$



El modelo tendría un error muy grande:

3) Assign each point to the closest centroid

3) Assign each point to the closest centroid

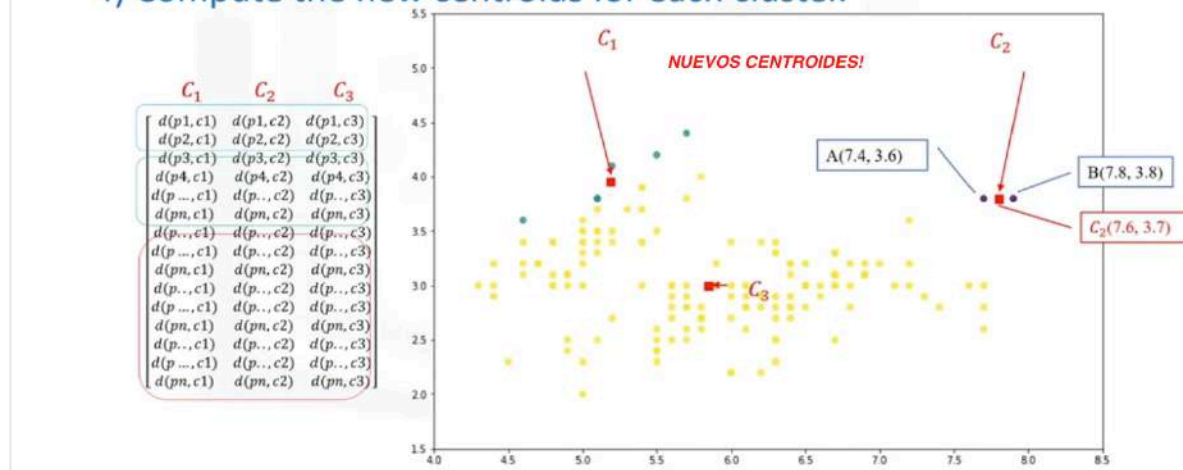


En el siguiente paso, cada centroide va a ser actualizado para ser la MEDIA de los puntos en cada cluster.

Los centroides de cada uno de los 3 clusters se convierten en la nueva media.

4) Compute the new centroids for each cluster.

4) Compute the new centroids for each cluster.



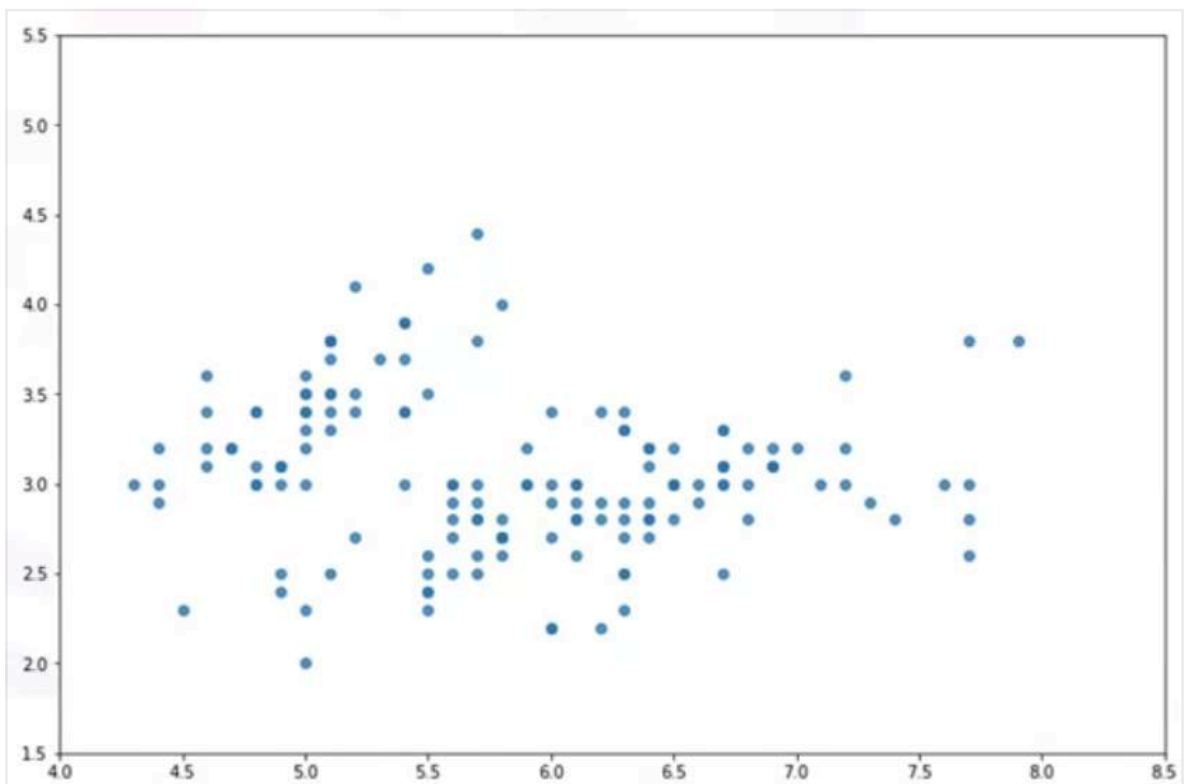
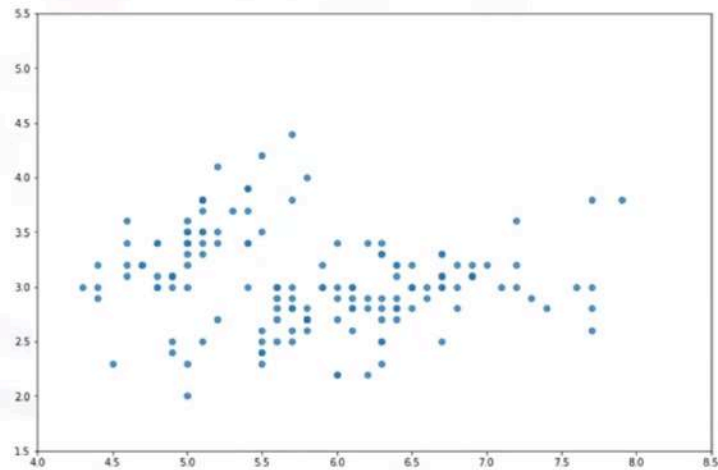
Este proceso se sigue repitiendo, recalcando el error y volviendo a mover los centroides hasta que estos no se muevan mas. Osea que vamos a haber encontrado la forma optima del centroide. Notar que cuando movemos los centroides, vamos cambiando la forma de los clusters. Ya que al moverse el centroide, cambia para cada punto cual es el centroide

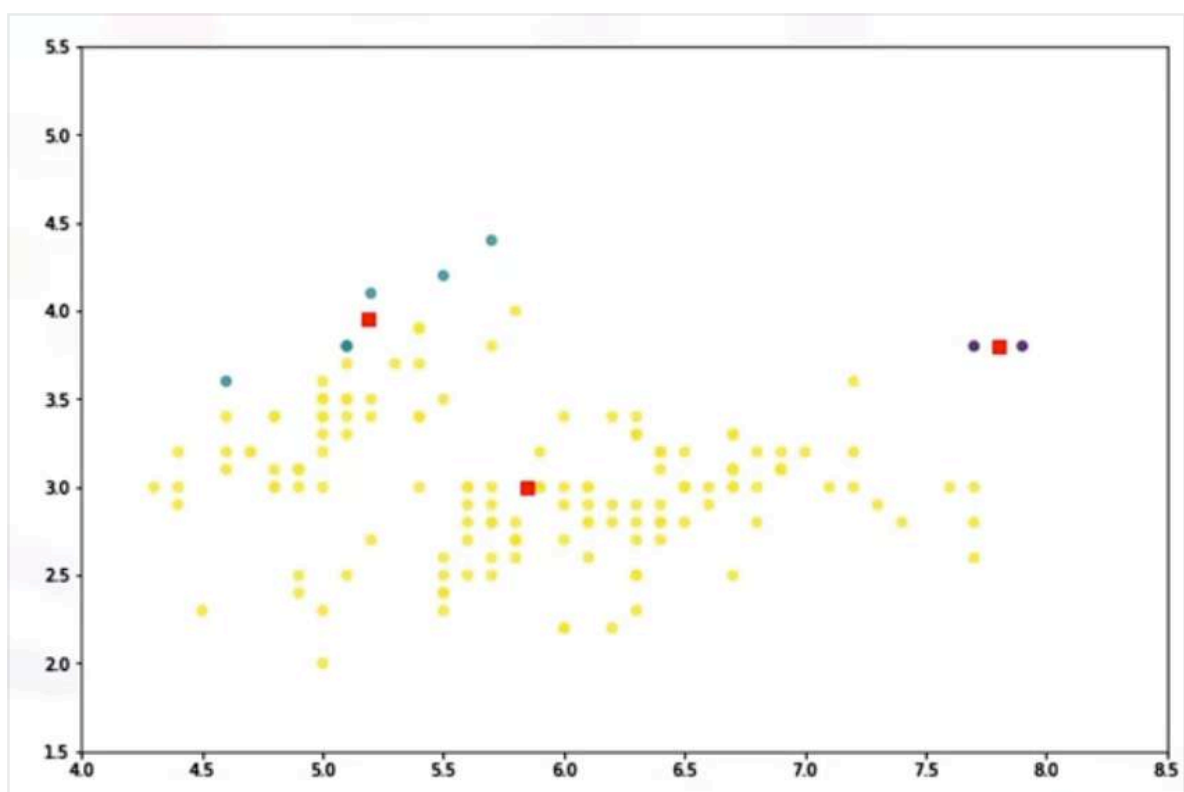
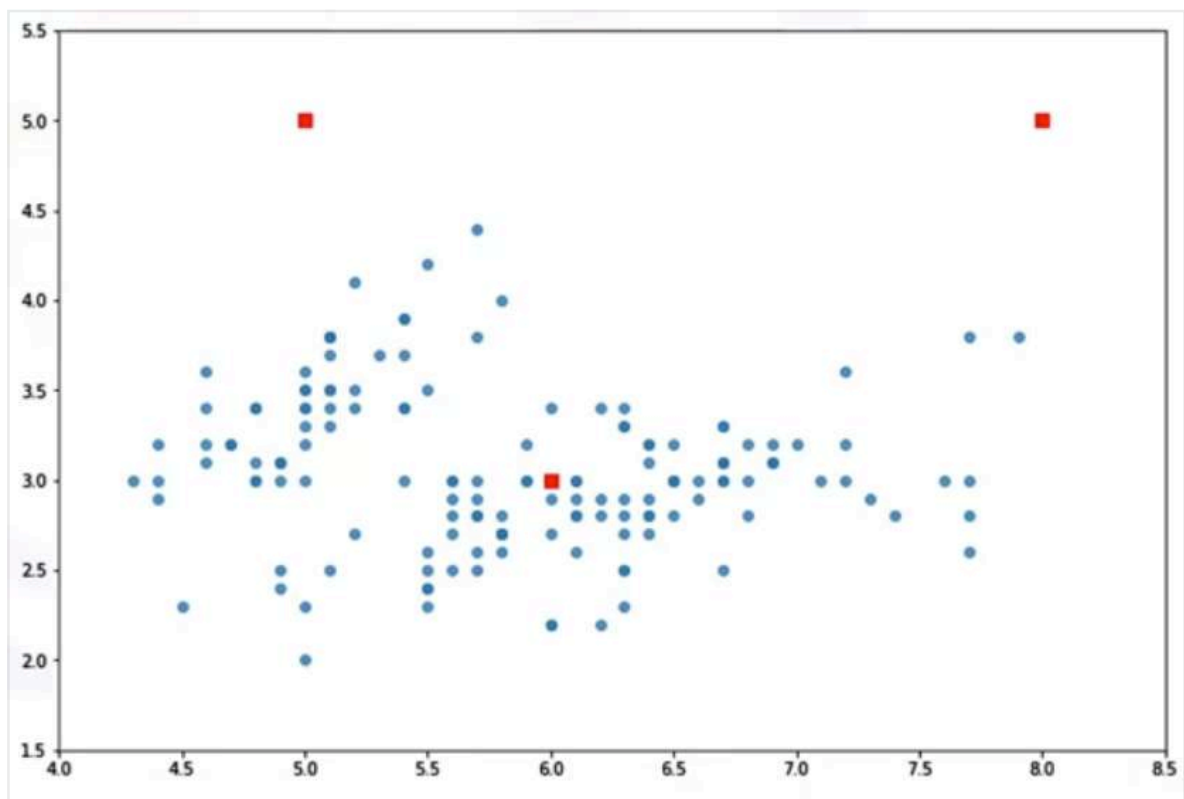
mas cercano a si mismo.

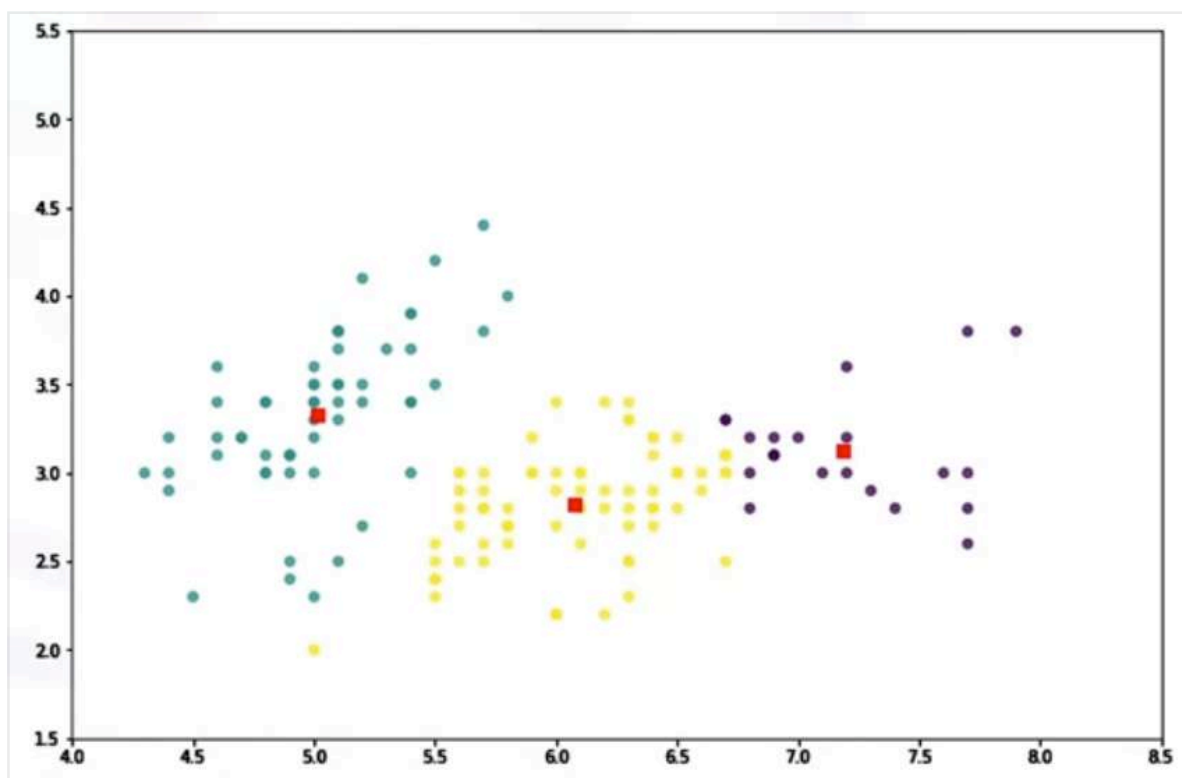
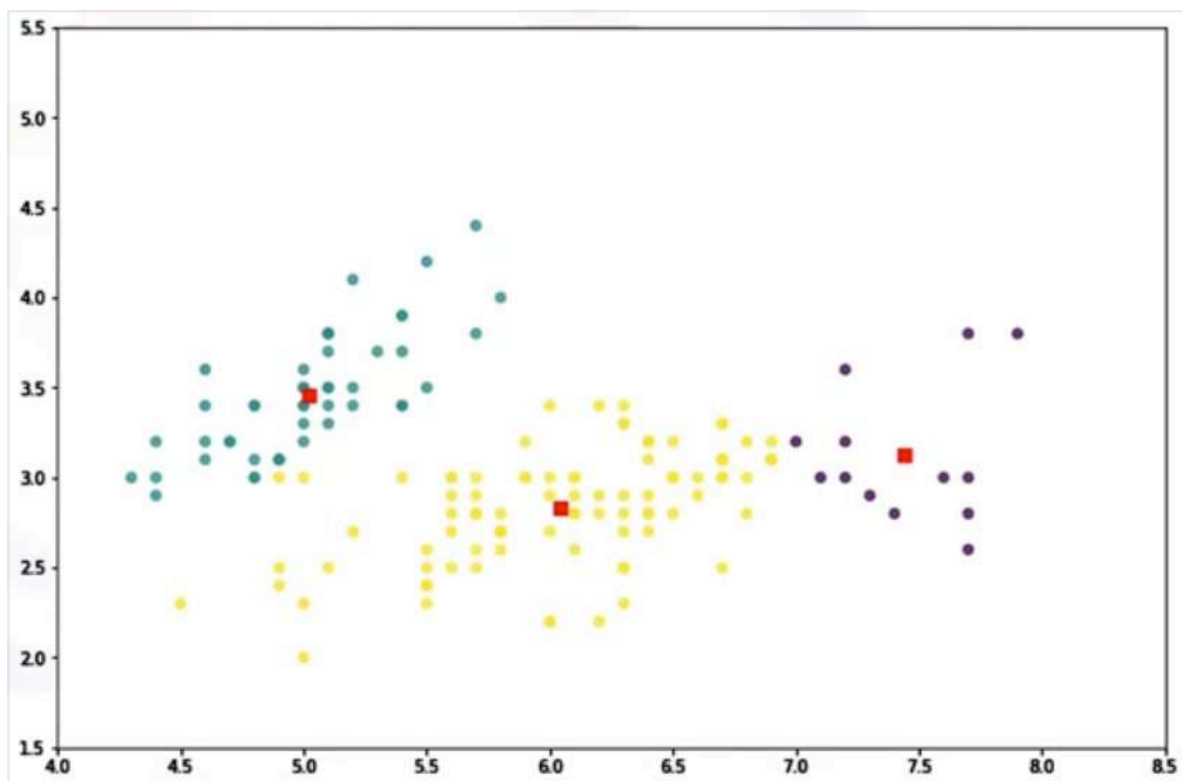
Es un algoritmo iterativo, entonces tenemos que repetir las vueltas hasta que el mismo converja.

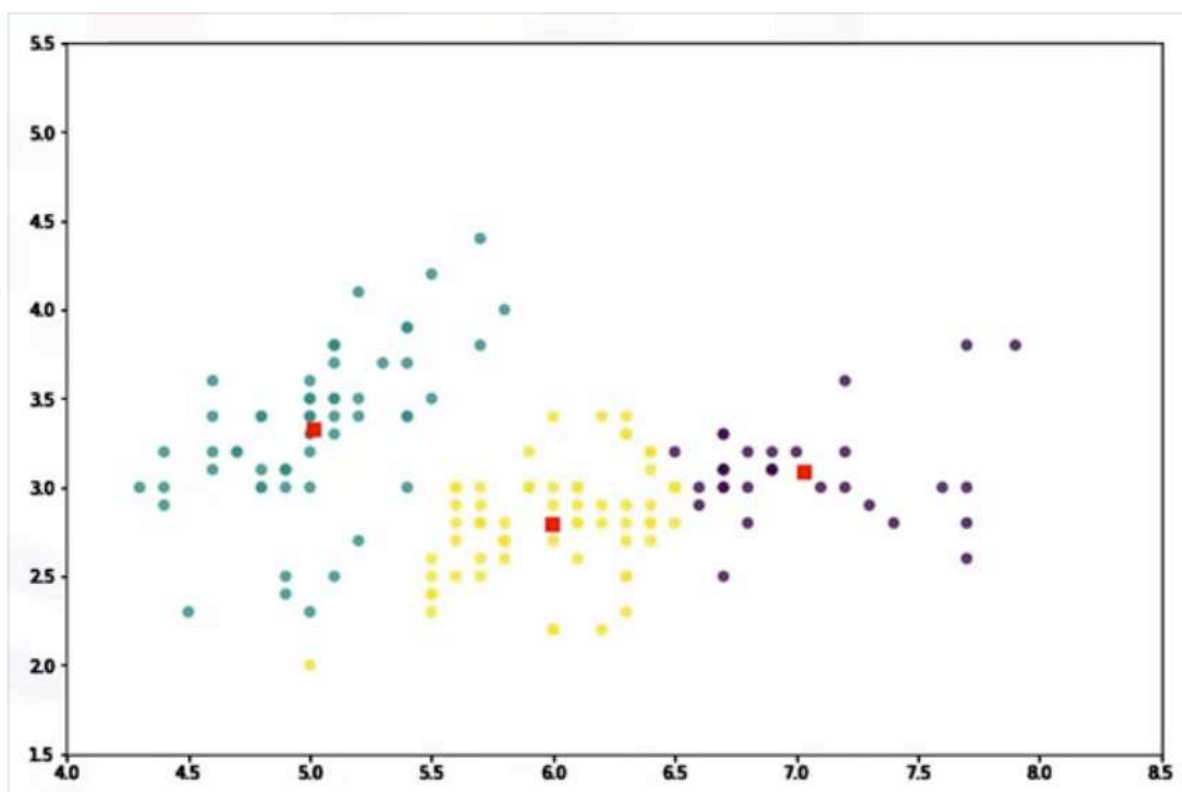
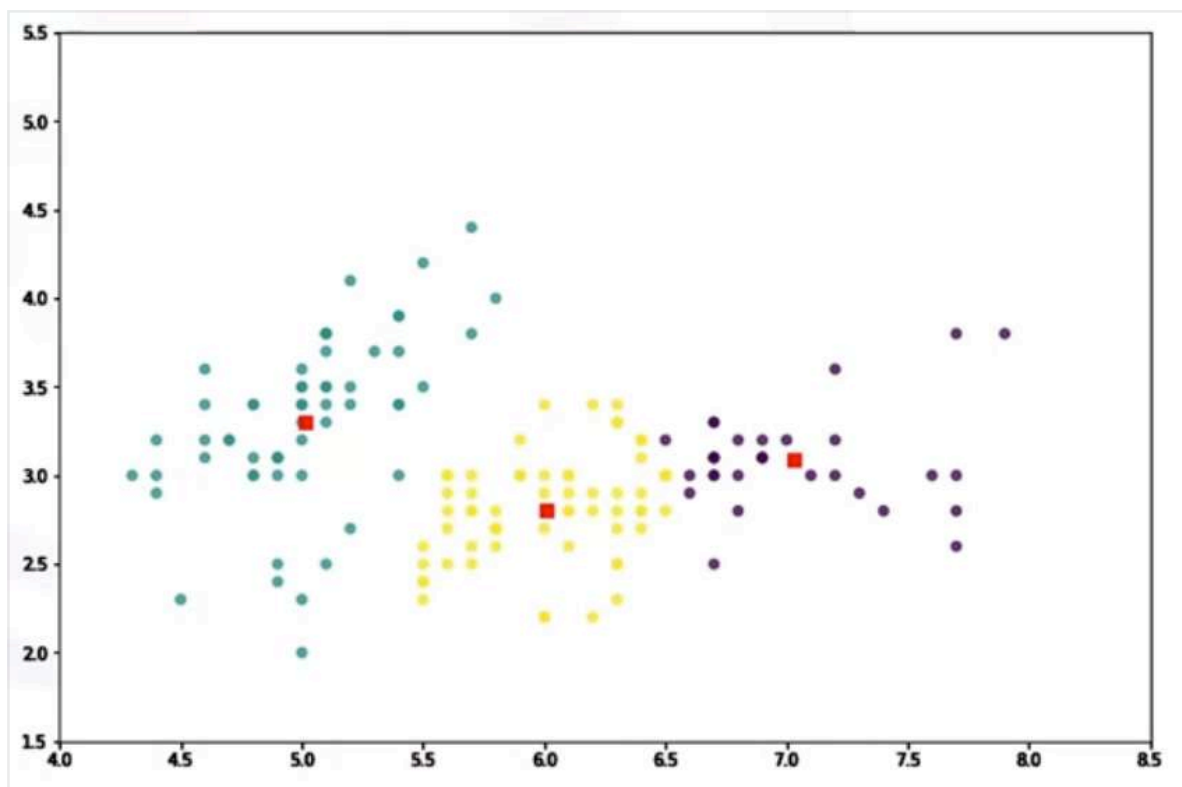
k-Means clustering – repeat

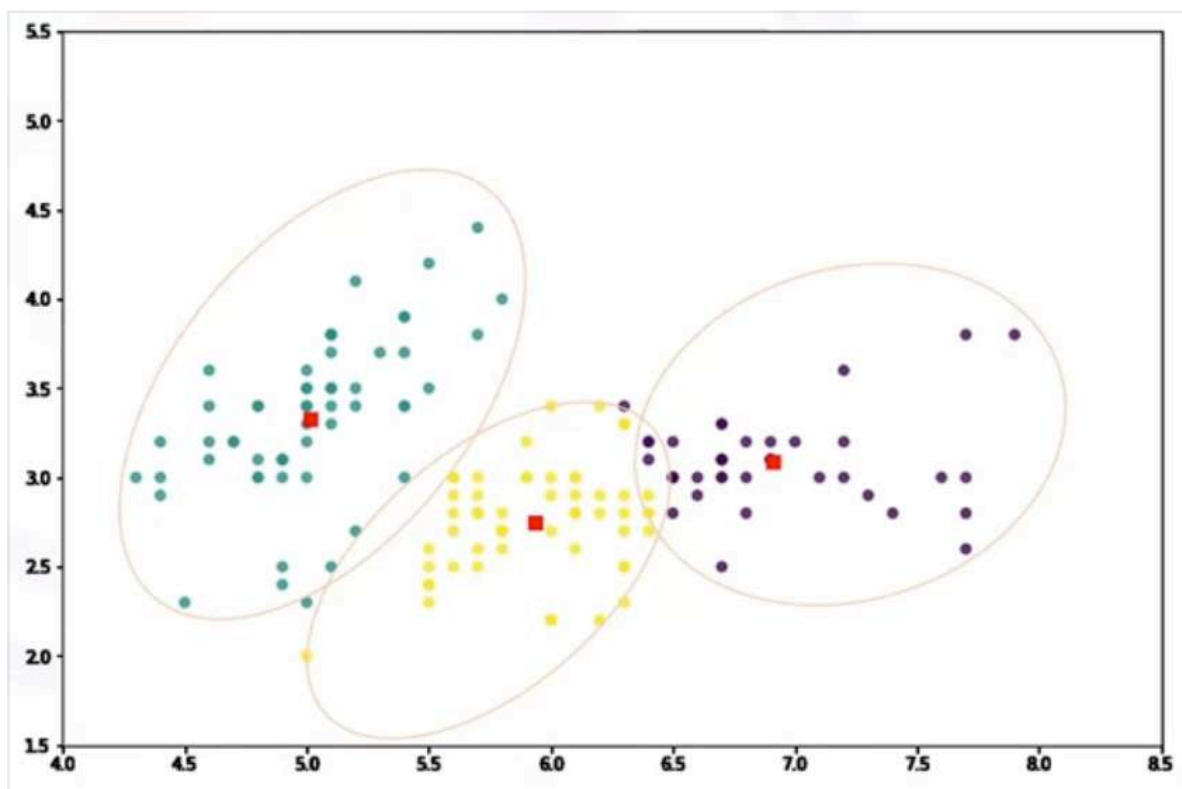
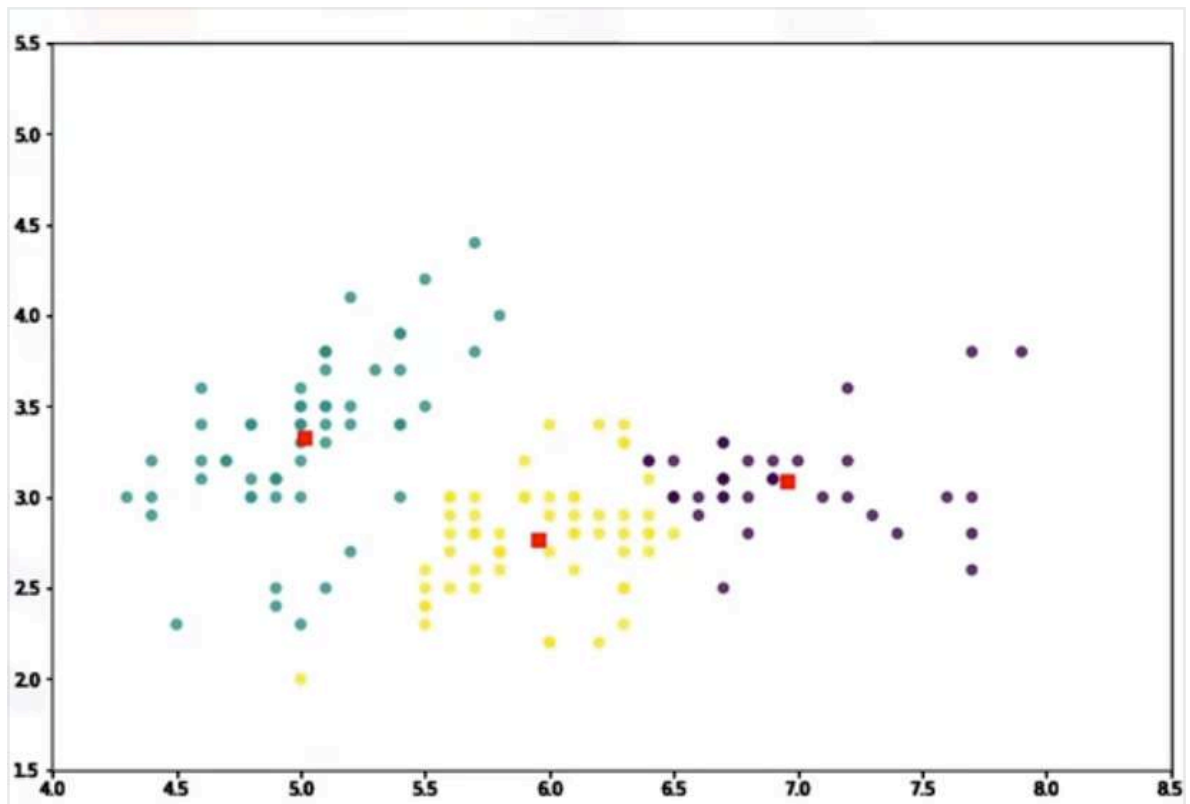
5) Repeat until there are no more changes.











Sin embargo, como es un algoritmo heurístico (?) no hay ninguna certeza de que converja al óptimo global. Y el resultado va a depender en gran medida de los clusters iniciales. Osea que nos va a estar dando un óptimo local, no necesariamente el mejor posible.

Para resolver este problema, es común correr el proceso muchas veces con condiciones iniciales diferentes. Como el algoritmo es por lo general

muy rápido, no debería haber problema con correrlo muchas veces.

More on k-Means

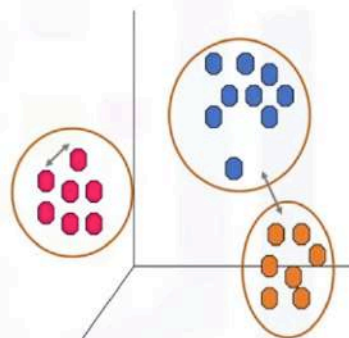
Characteristics y accuracy.

k-Means clustering algorithm

1. Randomly placing k centroids, one for each cluster.
2. Calculate the distance of each point from each centroid.
3. Assign each data point (object) to its closest centroid, creating a cluster.
4. Recalculate the position of the k centroids.
5. Repeat the steps 2-4, until the centroids no longer move.

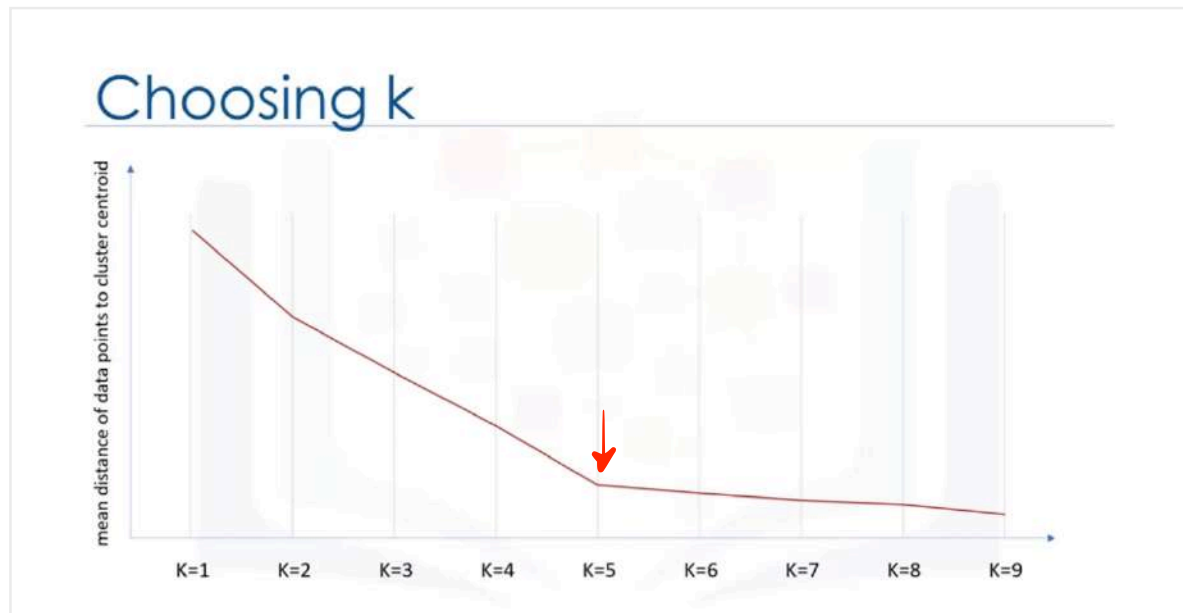
k-Means accuracy

- External approach
 - Compare the clusters with the ground truth, if it is available.
- Internal approach
 - Average the distance between data points within a cluster.

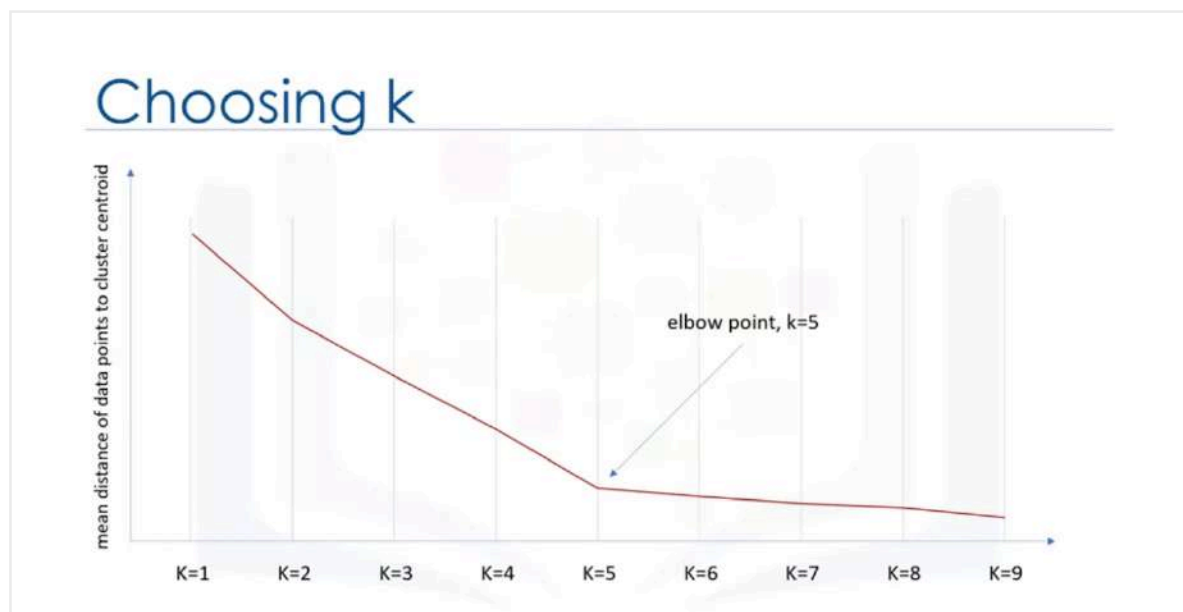


Para elegir K , se suele ir corriendo el algoritmo y ver como evoluciona alguna métrica de accuracy para el clustering, como la distancia promedio entre los puntos contra el centroide, esto indica que tan densos son nuestros clusters o bien hasta que punto minimizamos el error de los

clusters.



El problema es que SIEMPRE, aumentando el numero de clusters, se va a reducir la distancia de los puntos a los centroides. Por lo que no me permite elegir un K tan piola.... Lo que se suele hacer es tomar el ELBOW POINT que es donde la pendiente cambia drásticamente:



Esa es la K correcta para el clustering. Este método se conoce como el Elbow Method.

k-Means recap

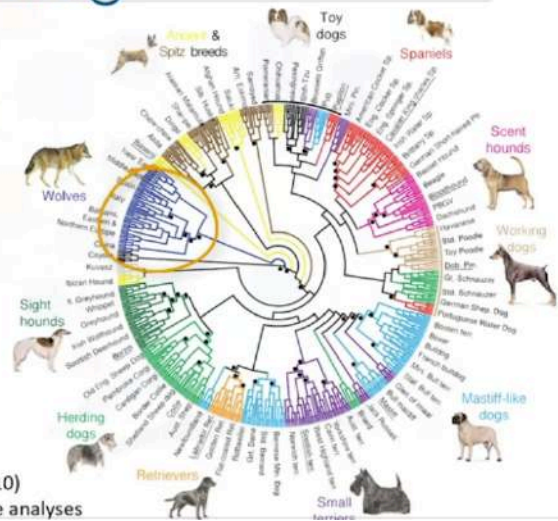
- Med and Large sized databases (*Relatively efficient*)
- Produces sphere-like clusters
- Needs number of clusters (k)

HIERARCHICAL CLUSTERING

Intro to Hierarchical Clustering

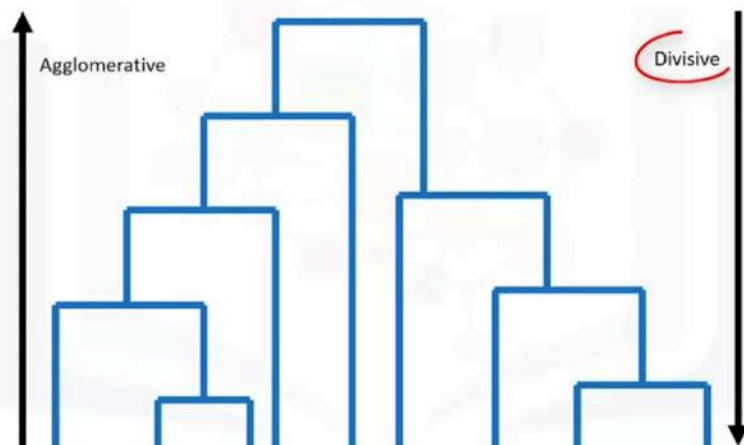
Hierarchical clustering

Hierarchical clustering algorithms build a hierarchy of clusters where **each node is a cluster** consists of the clusters of its daughter nodes.



Source: von Holdt B.M. et al. (2010)
Genome-wide SNP and haplotype analyses

Hierarchical clustering



El agglomerative es el mas popular en ds y es en lo que nos vamos a centrar en este video.

Agglomerative clustering

TO OT MO VA ED WI

	TO	OT	VA	MO	WI	ED
TO		351	3363	505	1510	2699
OT			3543	167	1676	2840
VA				3690	1867	819
MO					1824	2976
WI						1195
ED						




Tomamos los dos clusters con menor distancia entre si y los agrupamos. En este ejemplo son Ottawa y Montreal. Tenemos que mercera estas dos ciudades cuando construimos el cluster:

	TO	OT/MO	VA	WI	ED
TO		351	3363	1510	2699
OT/MO			3543	1676	2840
VA				1867	819
WI					1195
ED					

Por lo tanto, las distancias de todos los demás clusters a este Cluster tambien se actualizan. Pero como? Como calculamos la distancia desde Winnipeg hasta nuestro nuevo cluster mergeado. Hay muchos approaches pero por ejemplo podríamos seleccionar la distancia al centro de este nuevo cluster.

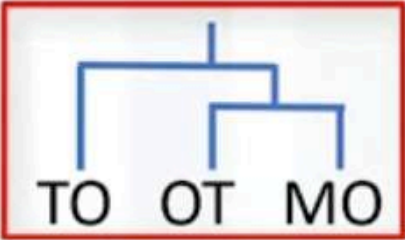
Agglomerative clustering



	TO	OT/MO	VA	WI	ED
TO		351	3363	1510	2699
OT/MO			3543	1676	2840
VA				1867	819
WI					1195
ED					



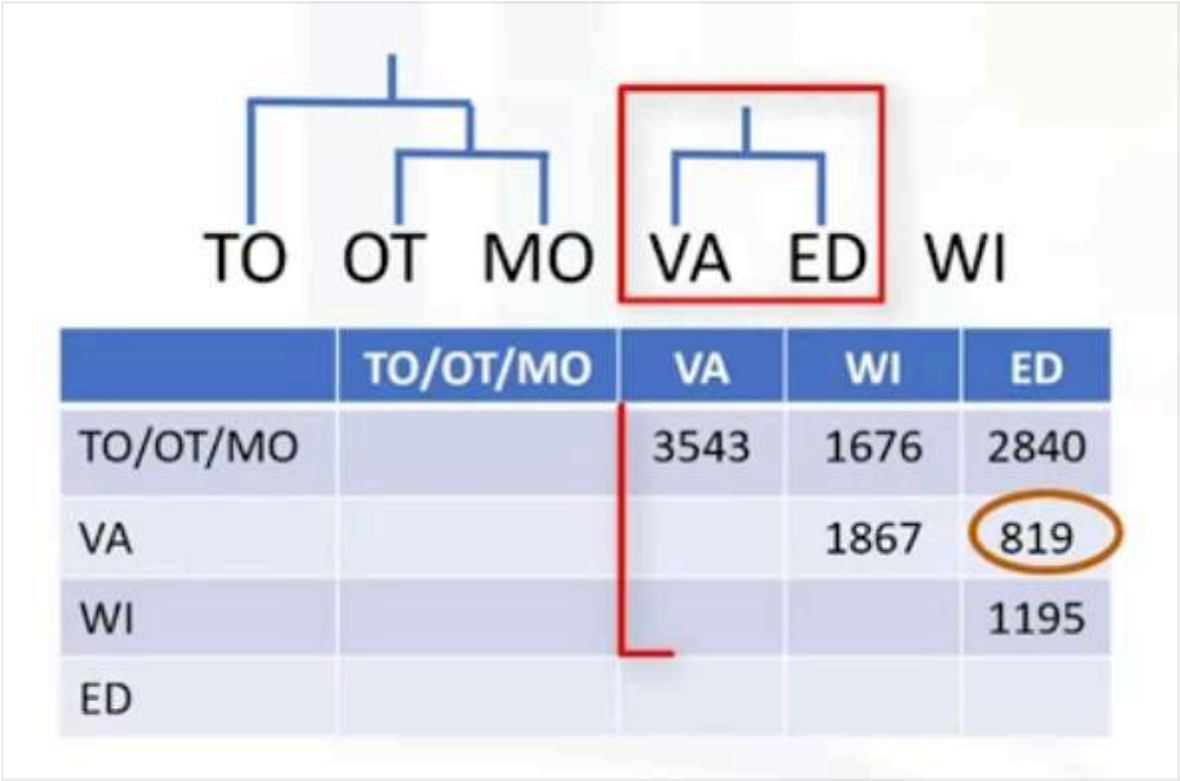
Volvemos a repetir el proceso, vemos que ahora nuestro cluster doble es el y Toronto son los dos clusters mas cercanos de la tabla:



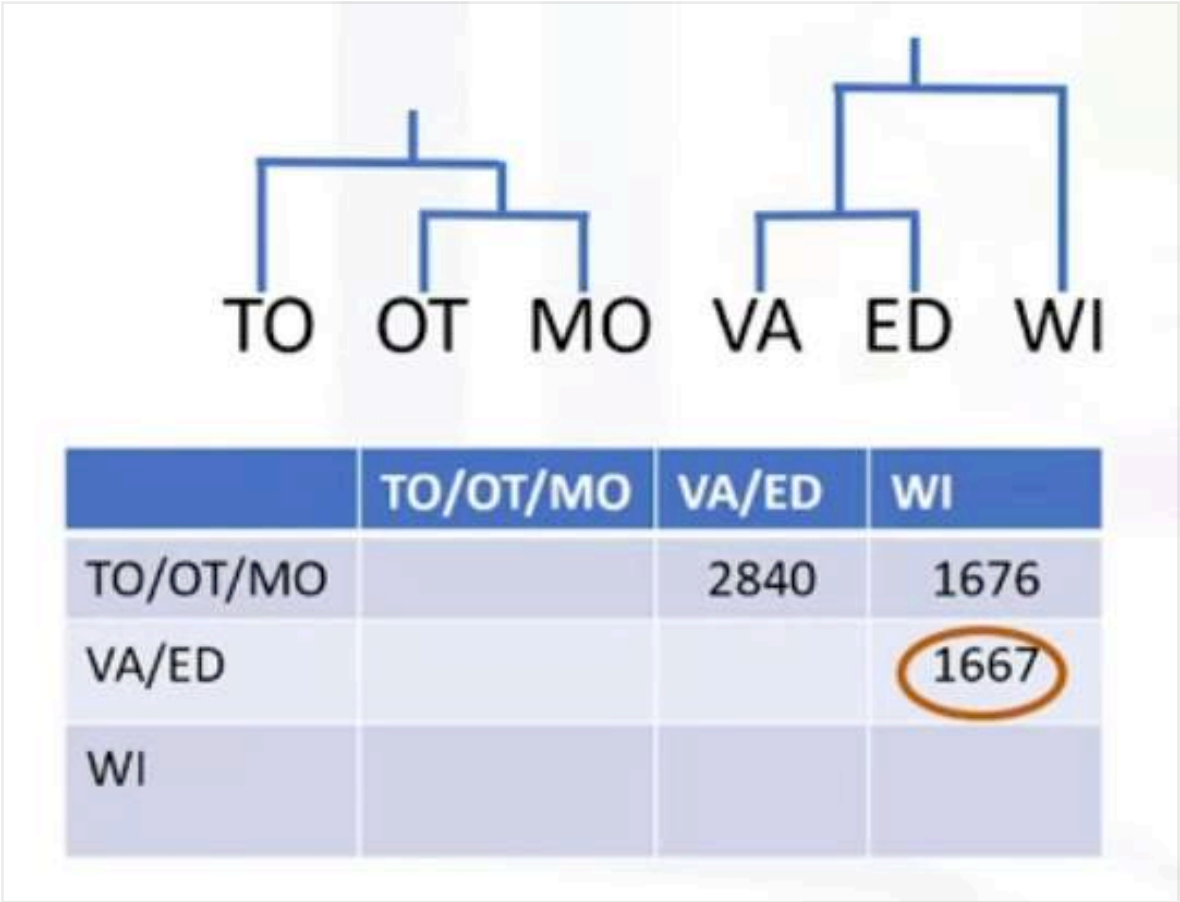
	TO	OT/MO	VA	WI	ED
TO		351	3363	1510	2699
OT/MO			3543	1676	2840
VA				1867	819
WI					1195
ED					

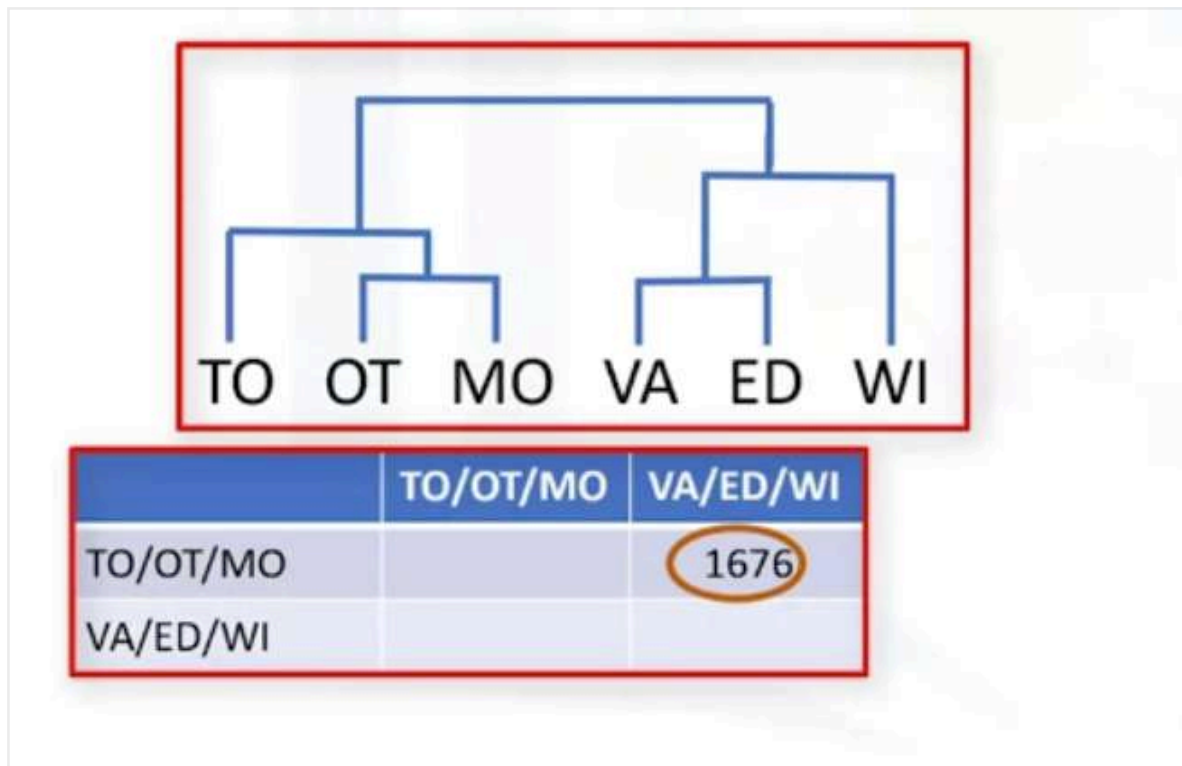
entonces los agrupamos.

En el siguiente paso, la distancia mas corta es entre ED y VA:



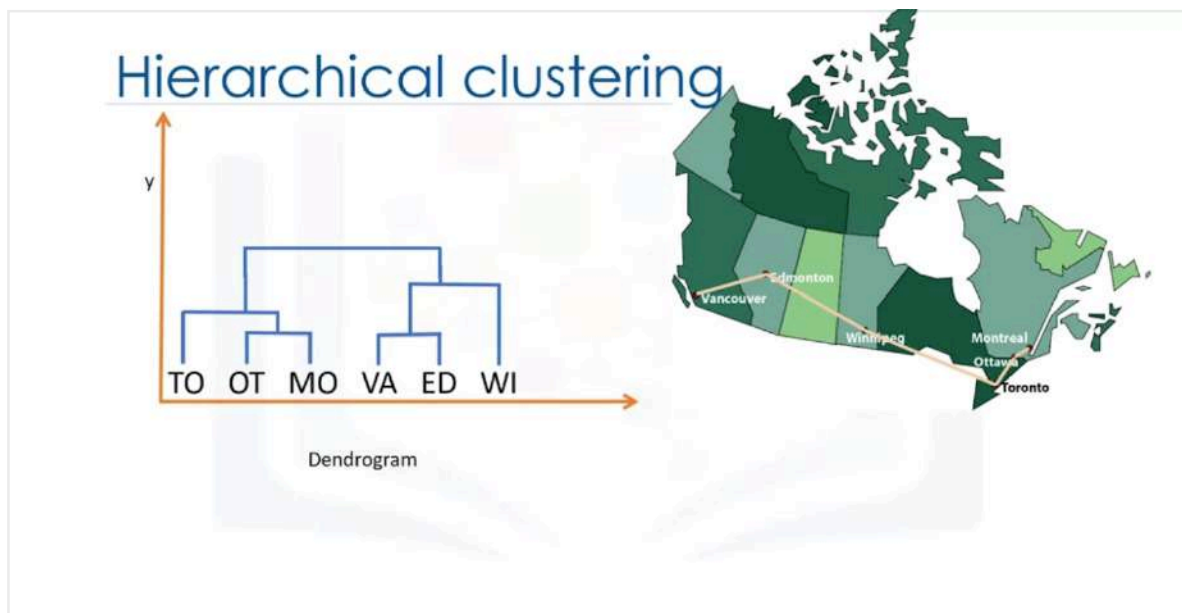
Asi seguimos...





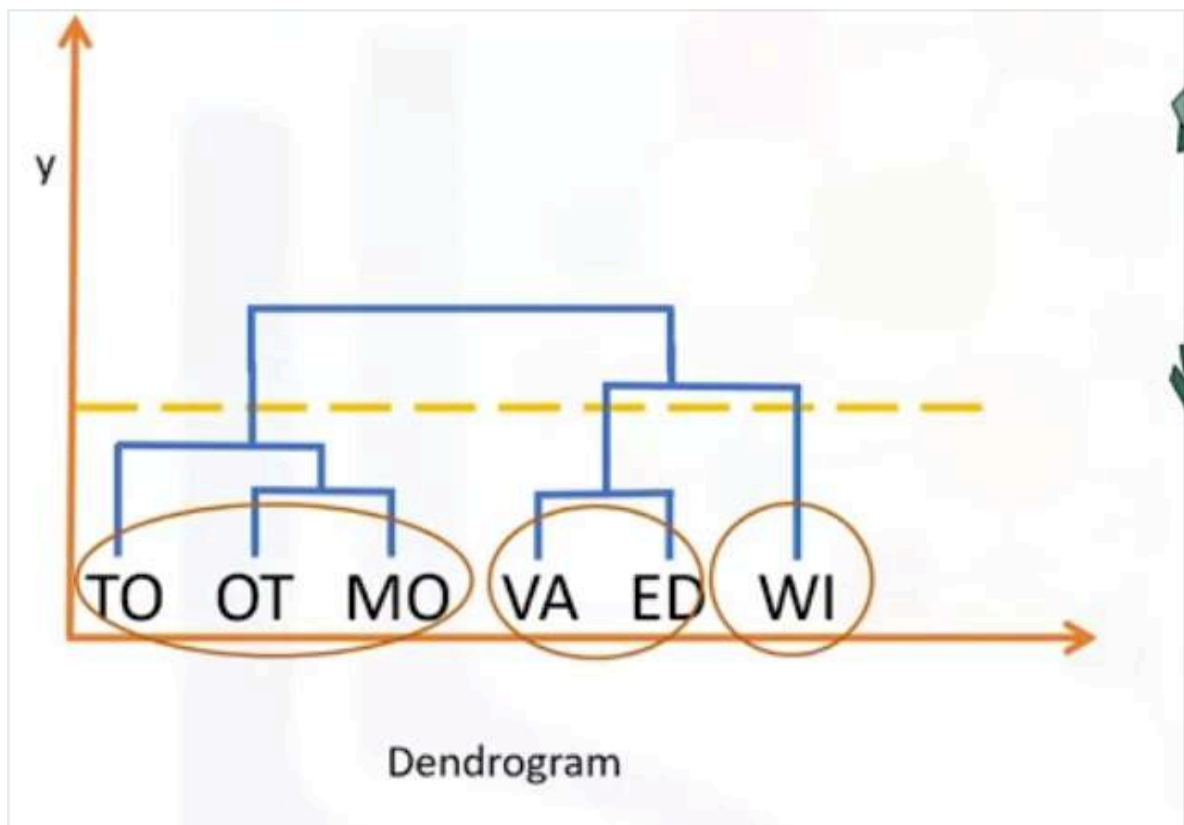
repetimos hasta que todos los clusters son unidos y se completa el árbol.

Hierarchical clustering se suele mostrar en un Dendrograma, como en siguiente slide:



Cada merge es representado por una línea horizontal. La coordenada Y (altura de la union) es el grado de similaridad de los dos clusters siendo unidos.

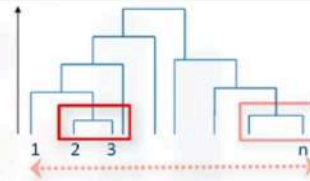
Si bien no requiere una cantidad específica de clusters, en algunas aplicaciones queremos una partición de clusters distintos como en flat clustering. En esos casos, la jerarquía tiene que ser cortada en algún punto.



More on Hierarchical Clustering

Agglomerative algorithm

1. Create n clusters, one for each data point
2. Compute the Proximity Matrix
3. Repeat
 - i. Merge the two closest clusters
 - ii. Update the proximity matrix
4. Until only a single cluster remains



$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ d(n,1) & d(n,2) & \dots & \dots & 0 \end{bmatrix}$$

Como calculamos las distancias entre clusters?

Similarity/Distance



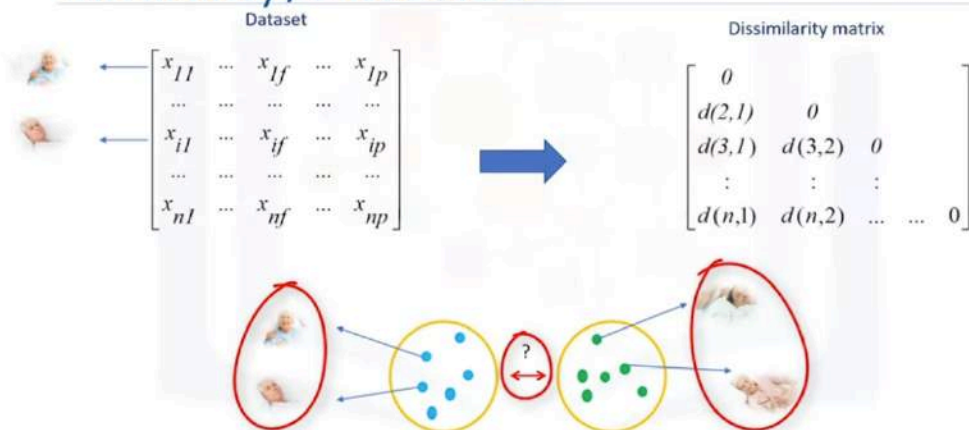
Patient 1		
Age	BMI	BP
54	190	120



Patient 2		
Age	BMI	BP
50	200	125

$$\begin{aligned} \text{Dis}(p1, p2) &= \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \\ &= \sqrt{(54 - 50)^2 + (190 - 200)^2 + (120 - 125)^2} \\ &= 11.87 \end{aligned}$$

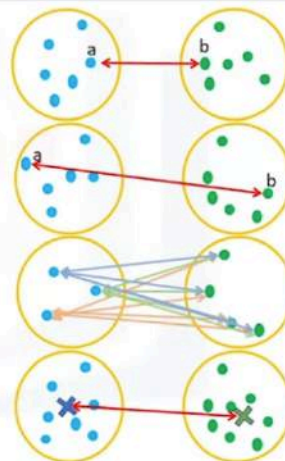
Similarity/Distance



El problema es que metimos a los wachs adentro de clusters. Como medimos las distancias ENTRE clusters? Hay diferentes criterios...

Distance between clusters

- **Single-Linkage Clustering**
 - Minimum distance between clusters
- **Complete-Linkage Clustering**
 - Maximum distance between clusters
- **Average Linkage Clustering**
 - Average distance between clusters
- ★ • **Centroid Linkage Clustering**
 - Distance between cluster centroids



Por lo general depende 100% en el tipo de datos, la dimensionalidad y el tipo de dataset con el que estemos trabajando.

Advantages vs. disadvantages

Advantages	Disadvantages
Doesn't required number of clusters to be specified.	Can never undo any previous steps throughout the algorithm.
Easy to implement.	Generally has long runtimes.
Produces a dendrogram, which helps with understanding the data.	Sometimes difficult to identify the number of clusters by the dendrogram.

Hierarchical clustering Vs. K-means

K-means	Hierarchical Clustering
1. Much more efficient	1. Can be slow for large datasets
2. Requires the number of clusters to be specified	2. Does not require the number of clusters to run
3. Gives only one partitioning of the data based on the predefined number of clusters	3. Gives more than one partitioning depending on the resolution
4. Potentially returns different clusters each time it is run due to random initialization of centroids	4. Always generates the same clusters

DENSITY BASED CLUSTERING

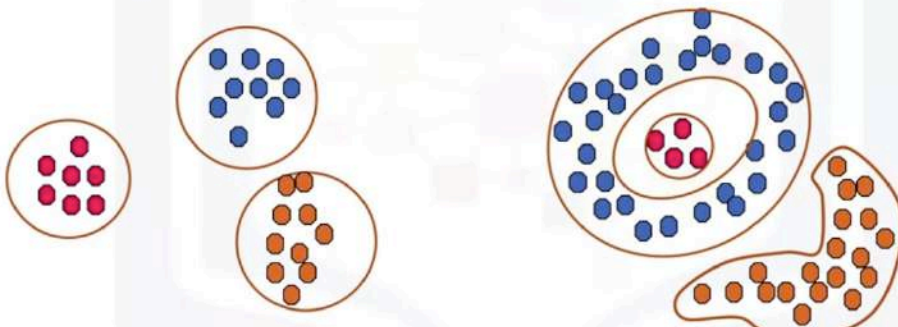
DBSCAN Clustering

Clave usarlo cuando estamos analizando spatial data.

Density-based clustering

- Spherical-shape clusters

- Arbitrary-shape clusters

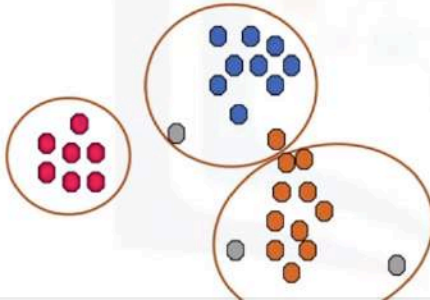


Cuando queremos **clusters de formas arbitrarias o clusters dentro de clusters**, los métodos tradicionales pueden no lograr buenos resultados. Los elementos dentro de un mismo cluster pueden no compartir suficientes similitudes o bien la performance del algoritmo puede ser mala.

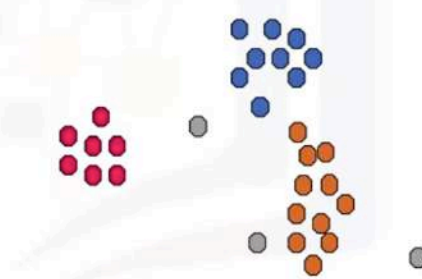
Además, mientras los algoritmos de partición como kmeans, pueden ser fáciles de entender y de implementar, estos no tienen noción de ningún tipo de outliers. TODOS los puntos son asignados a un cluster, incluso si no debieran pertenecer a ninguno. En el Domain de la detección de anomalías, esto por supuesto que trae problemas.

k-Means Vs. density-based clustering

- k-Means assigns all points to a cluster even if they do not belong in any

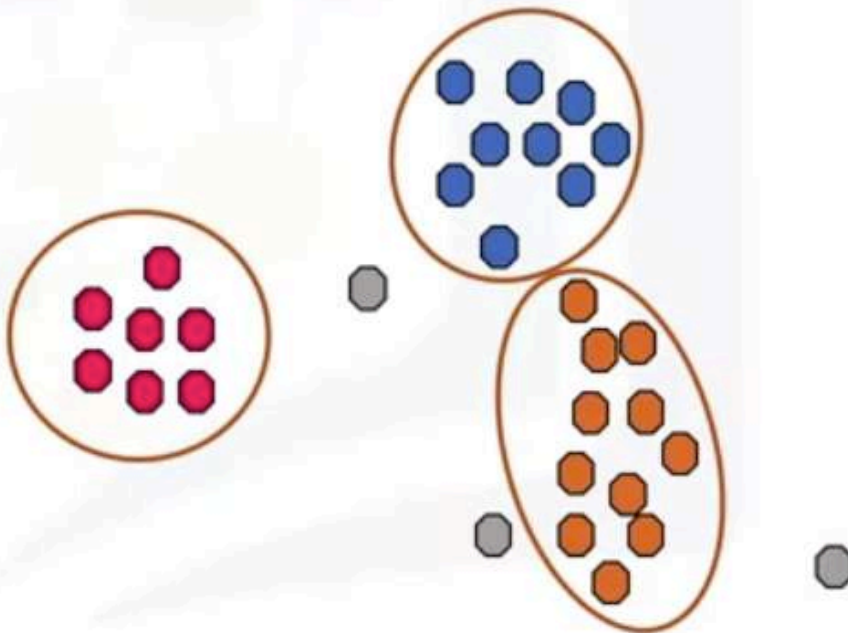


- Density-based Clustering locates regions of **high density**, and separates outliers



Por el contrario, los amigos algoritmos de densidad como DBSCAN si que separan a los outliers!

- Density-based Clustering locates regions of **high density**, and separates outliers



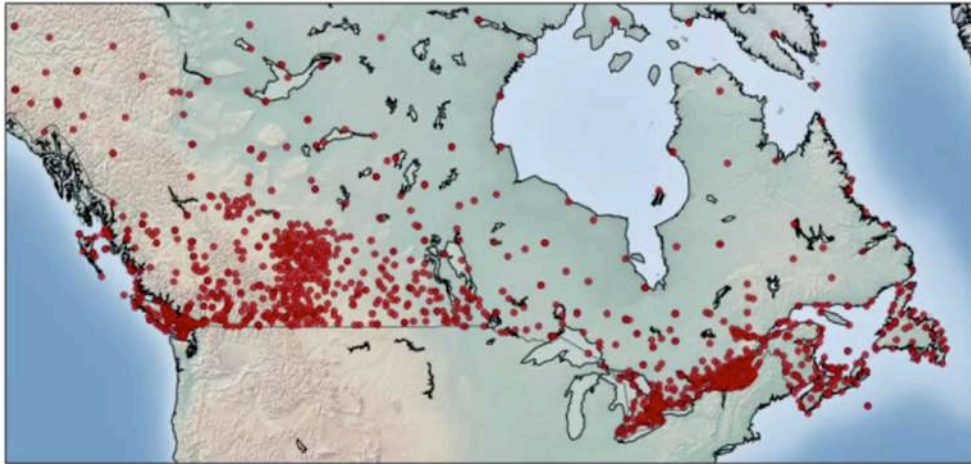
Se define densidad como la cantidad de puntos en un radio específico.

Dentro de los algoritmos de densidad, el mas popular es DBSCAN! *Puede encontrar clusters de cualquier forma arbitraria sin ser afectado por el*

ruido de los datos.

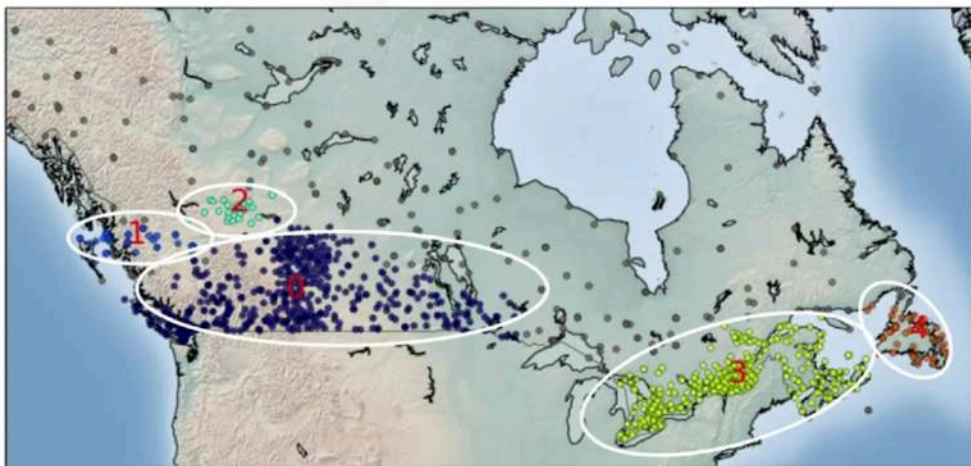
Antes de DBSCAN:

DBSCAN for class identification



Después:

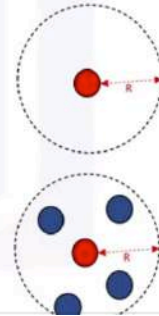
DBSCAN for class identification



Vamo a ver este algoritmo pa ver como funciona:

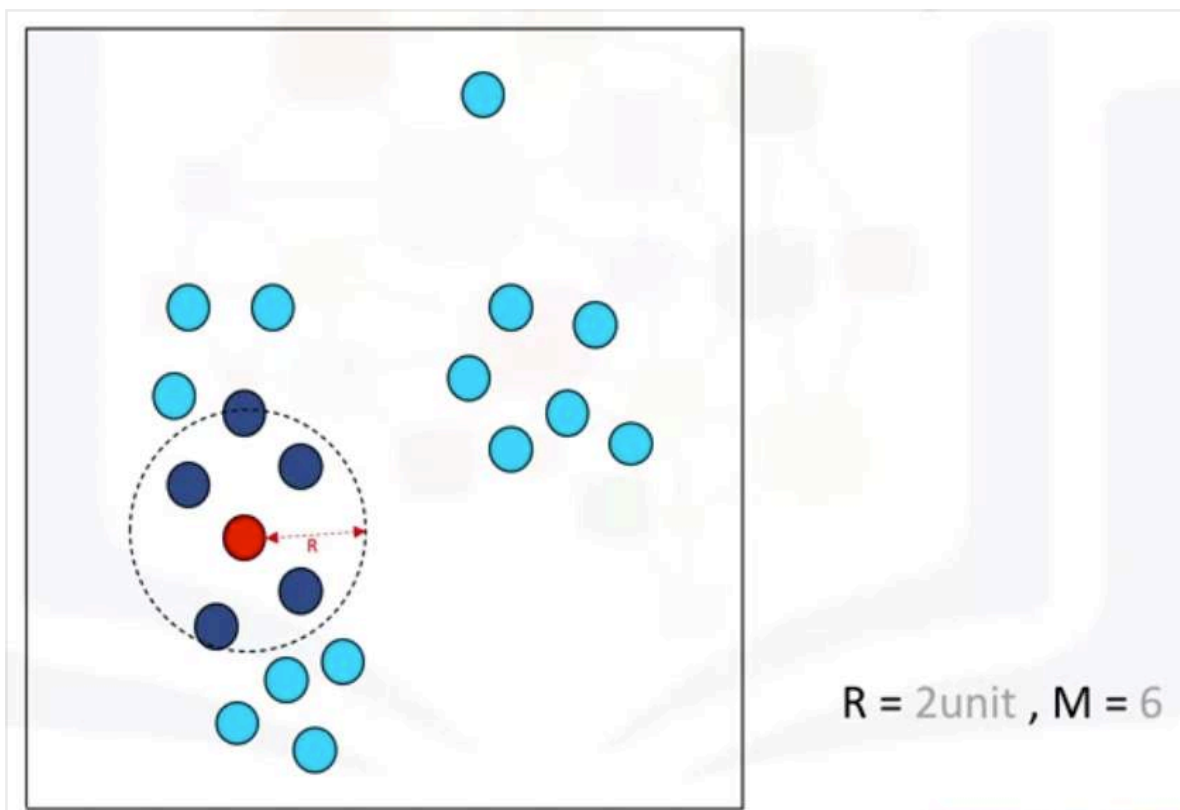
What is DBSCAN?

- DBSCAN (**D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise)
 - Is one of the most common clustering algorithms
 - Works based on density of objects
- R (**R**adius of neighborhood)
 - Radius (R) that if includes enough number of points within, we call it a dense area
- M (**M**in number of neighbors)
 - The minimum number of data points we want in a neighborhood to define a cluster

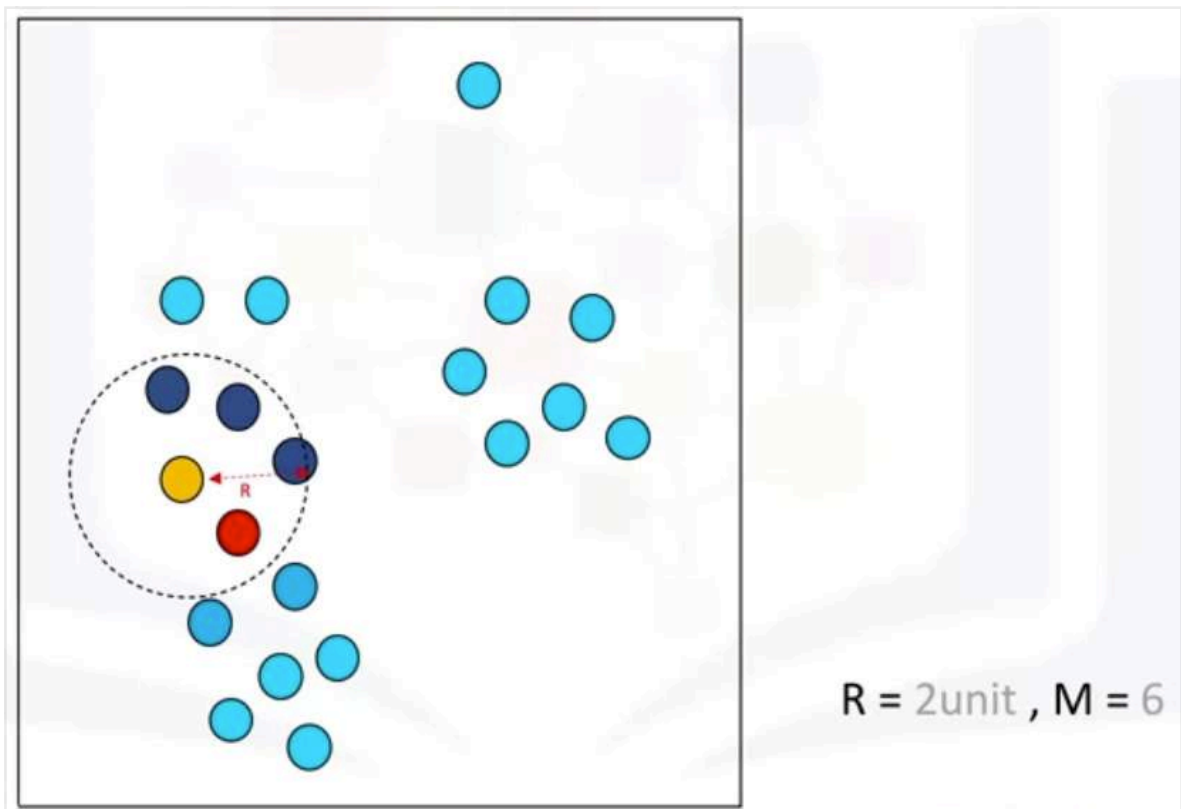


Algunas definiciones:

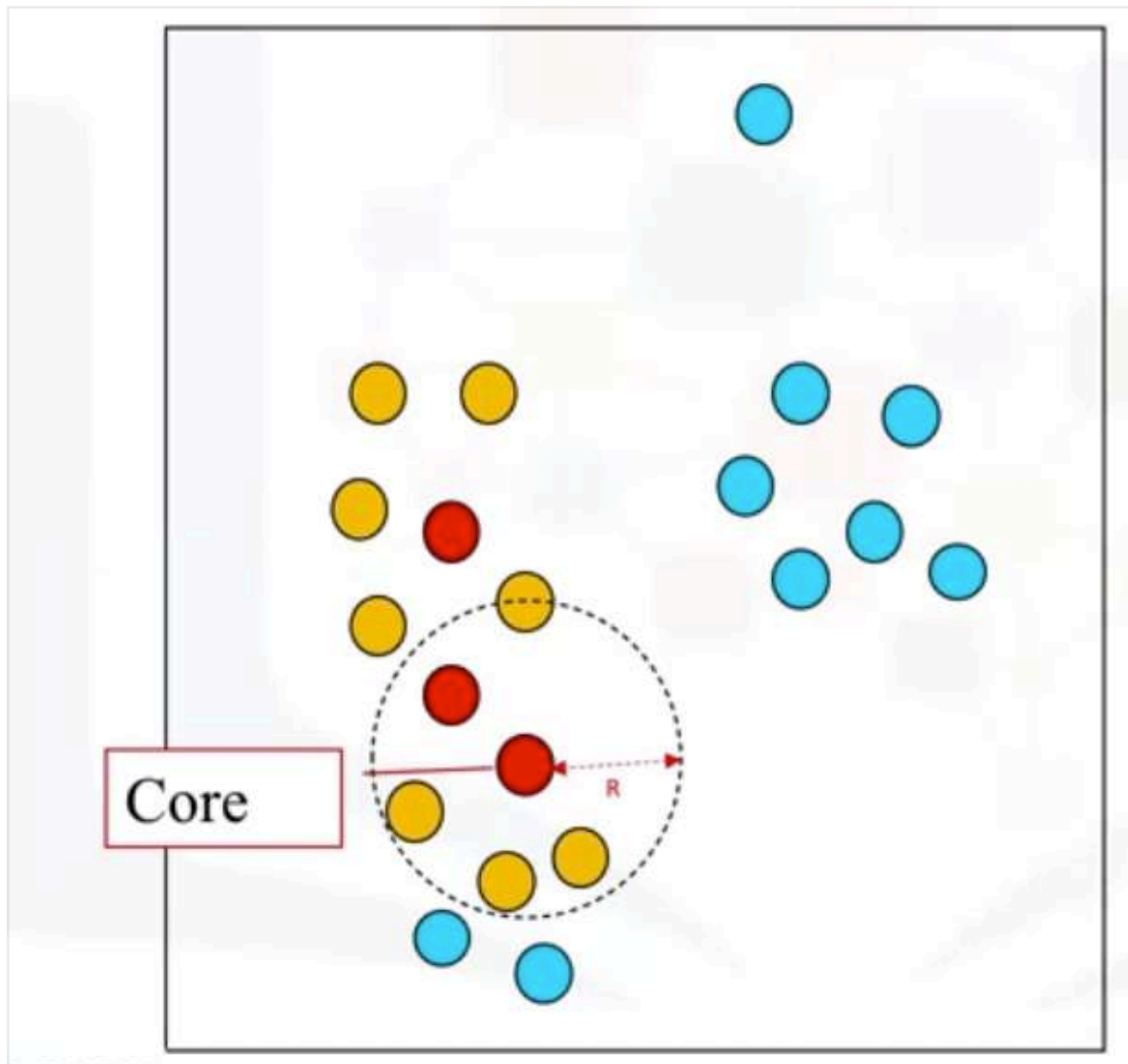
-Core point = Si y solo si en el vecindario (dentro del radio definido) de este hay al menos M puntos.



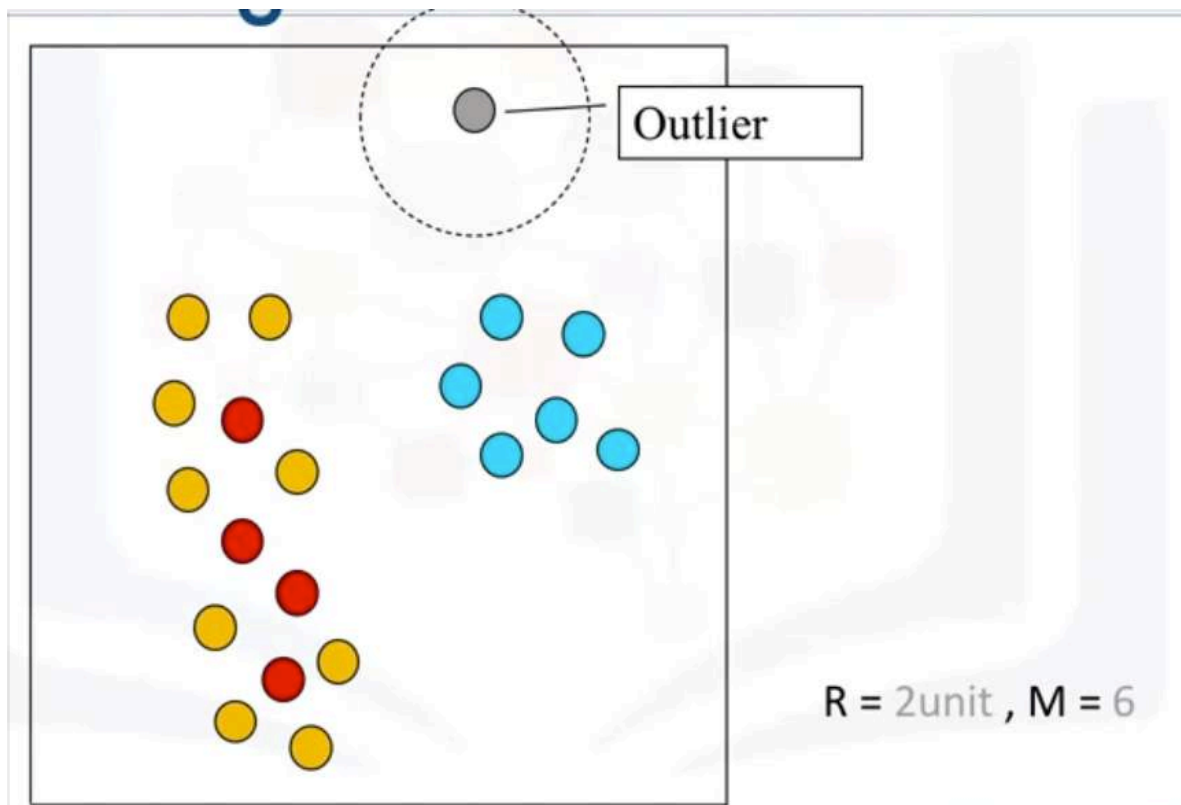
- Border Point: básicamente, si no es un core point. Tiene que tener menos de M puntos O es accesible desde algún core point. Donde ser accesible significa que este adentro de la distancia del core point.



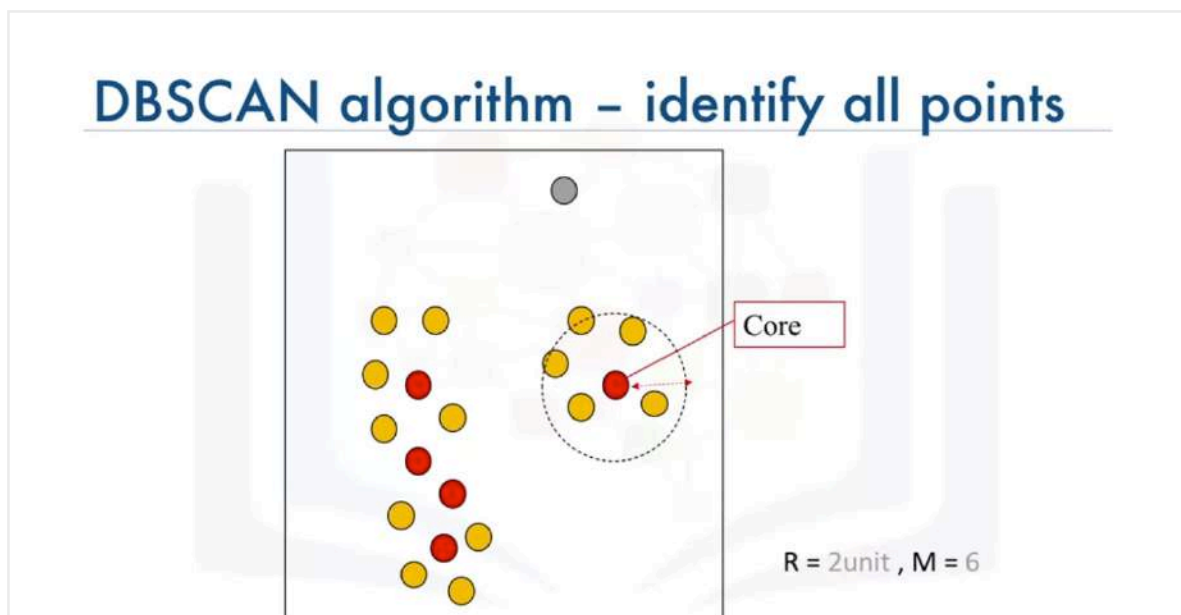
Siguiendo asi....



- **Outlier Point:** punto que no es un core point y ademas no esta lo suficiente mente cerca para ser accesible por un core point.

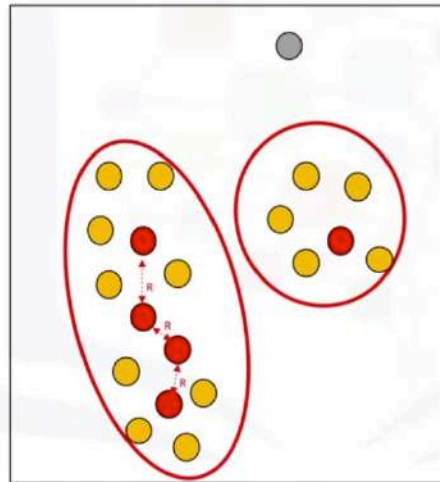


Una vez que analizamos todos:



El siguiente paso es **conectar core points que sean vecinos! Y ponerlos en el mismo cluster.** Un cluster esta formado por lo menos por un core point y todos sus core points accesibles + todos sus Borders points.

DBSCAN algorithm – clusters?



$R = 2\text{unit}$, $M = 6$

Porque esta piola DbScan?

1. Arbitrarily shaped clusters
2. Robust to outliers
3. Does not require specification of the number of clusters