



# ARISA Learning Material

**Educational Profile and EQF level: DATA SCIENTIS – EQF 6**

**PLO: 1, 2, 3, 4, 5**

**Learning Unit (LU): MACHINE LEARNING: UNSUPERVISED**

**Topic: 1-INTRODUCTION**



[www.aiskills.eu](http://www.aiskills.eu)

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

## Copyright © 2024 by the Artificial Intelligence Skills Alliance

All learning materials (including Intellectual Property Rights) generated in the framework of the ARISA project are made freely available to the public under an open license [Creative Commons Attribution–NonCommercial](#) (CC BY-NC 4.0).

## ARISA Learning Material 2024

This material is a draft version and is subject to change after review coordinated by the European Education and Culture Executive Agency (EACEA).

**Authors:** Universidad Internacional de La Rioja (UNIR)

**Disclaimer:** This learning material has been developed under the Erasmus+ project ARISA (Artificial Intelligence Skills Alliance) which aims to skill, upskill, and reskill individuals into high-demand software roles across the EU.



This project has been funded with support from the European Commission. The material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

- About ARISA

- The Artificial Intelligence Skills Alliance (ARISA) is a four-year transnational project funded under the EU's Erasmus+ programme. It delivers a strategic approach to sectoral cooperation on the development of Artificial Intelligence (AI) skills in Europe.
- ARISA fast-tracks the upskilling and reskilling of employees, job seekers, business leaders, and policymakers into AI-related professions to open Europe to new business opportunities.
- ARISA regroups leading ICT representative bodies, education and training providers, qualification regulatory bodies, and a broad selection of stakeholders and social partners across the industry.

[ARISA Partners & Associated Partners](#) | [LinkedIn](#) | [Twitter](#)

# Aprendizaje automático no supervisado

# Introducción

- **El aprendizaje no supervisado implica tareas que operan en conjuntos de datos sin respuestas etiquetadas ni valores objetivo.**
- **En cambio, el objetivo es capturar una estructura o información interesante.**
- **!!!!Válido para conjuntos de datos supervisados!!!!**
- **Aplicaciones del aprendizaje no supervisado:**
  - **Visualice la estructura de un conjunto de datos complejo.**
  - **Estimación de densidad para predecir probabilidades de eventos.**
  - **Comprima y resuma los datos.**
  - **Extraiga características para el aprendizaje supervisado.**
  - **Descubra clústeres importantes o valores atípicos.**

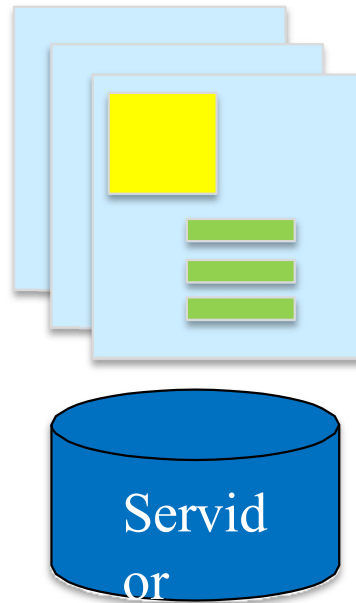
## Ejemplo



Población de usuarios

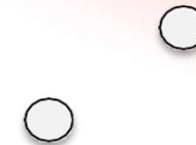
Señales de interacción

- Características del sitio utilizadas
- Páginas navegadas



Número de funciones avanzadas utilizadas

Expertos enfocados



Navegación casual

Número de páginas de productos navegadas

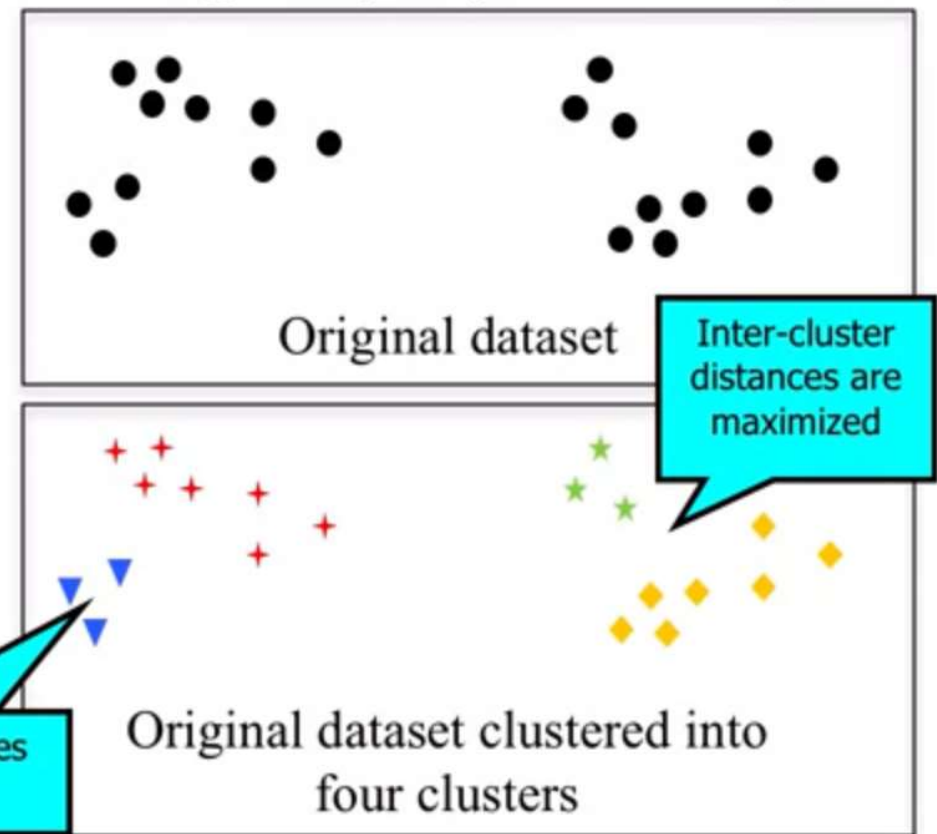
# Tipos

- **Transformaciones**
  - *Procesos que extraen o computan información (PCA, MultiDimensional Scaling MDS)*
- **Clustering (Agrupamiento)**
  - *Buscar grupos en los datos*
  - *Asigne cada punto del conjunto de datos a uno de los grupos*

# Clustering

## Finding a way to divide a dataset into groups ('clusters')

- Data points within the same cluster should be 'close' or 'similar' in some way.
- Data points in different clusters should be 'far apart' or 'different'
- Clustering algorithms output a cluster membership index for each data point:
  - *Hard clustering*: each data point belongs to exactly one cluster
  - *Soft (or fuzzy) clustering*: each data point is assigned a weight, score, or probability of membership for each cluster





# K-Means Clustering (Agrupacion K-medias)

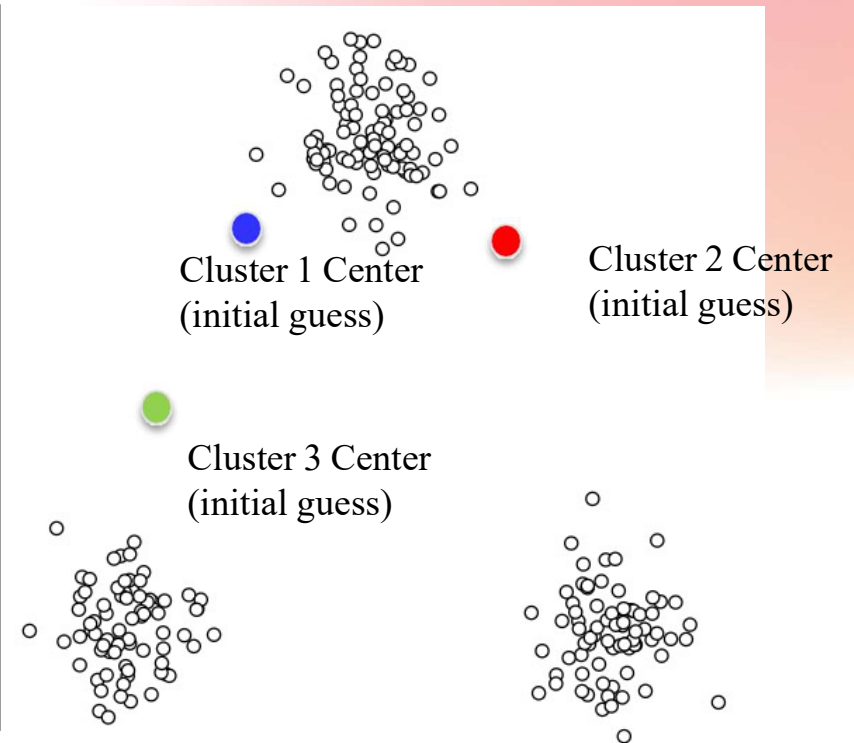
## The k-means algorithm

**Inicialización** Elija el número de clústeres  $k$  que desea encontrar. A continuación, elija  $k$  puntos aleatorios para que sirvan como una suposición inicial para los centros del clúster.

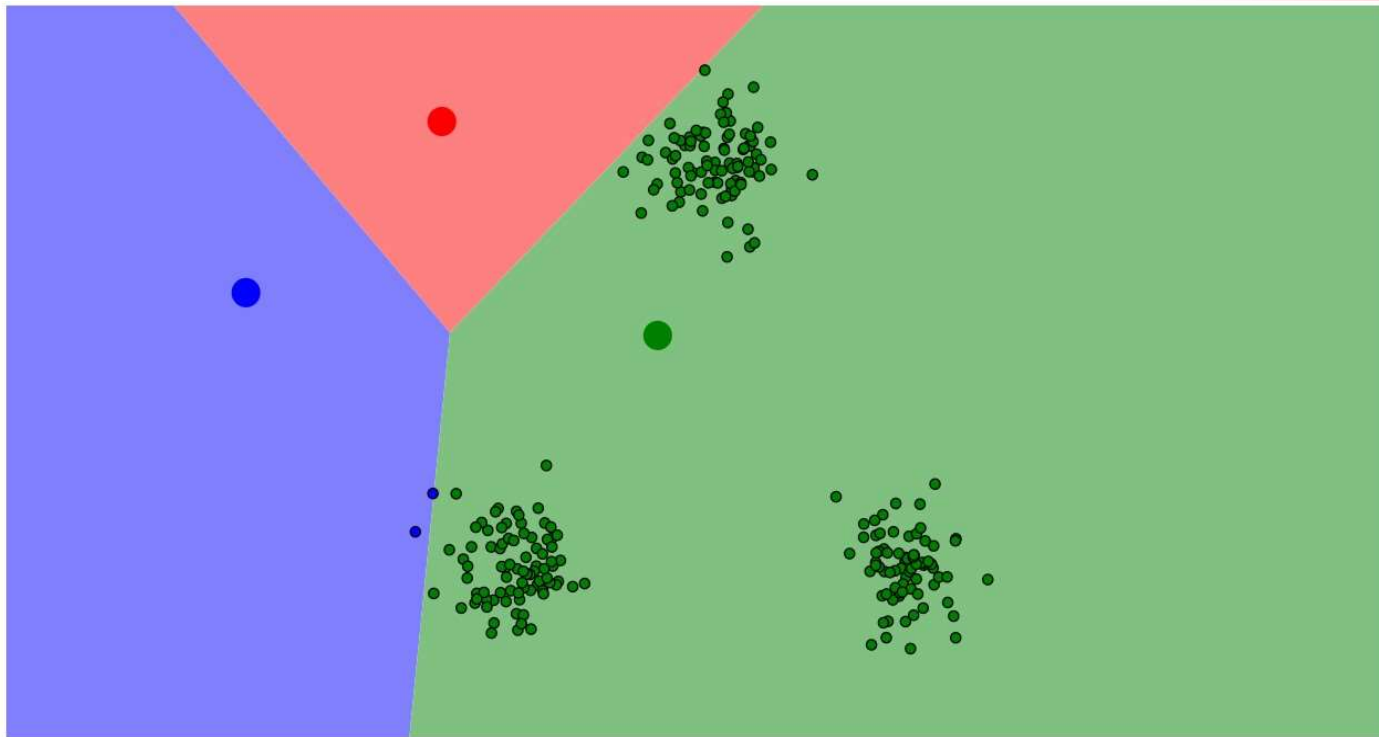
**Paso A** Asigne cada punto de datos al centro de clúster más cercano.

**Paso B** Actualice cada centro de clúster reemplazándolo por la media de todos los puntos asignados a ese grupo (en el paso A).

**Repita los pasos A y B hasta que los centros converjan en una solución estable.**



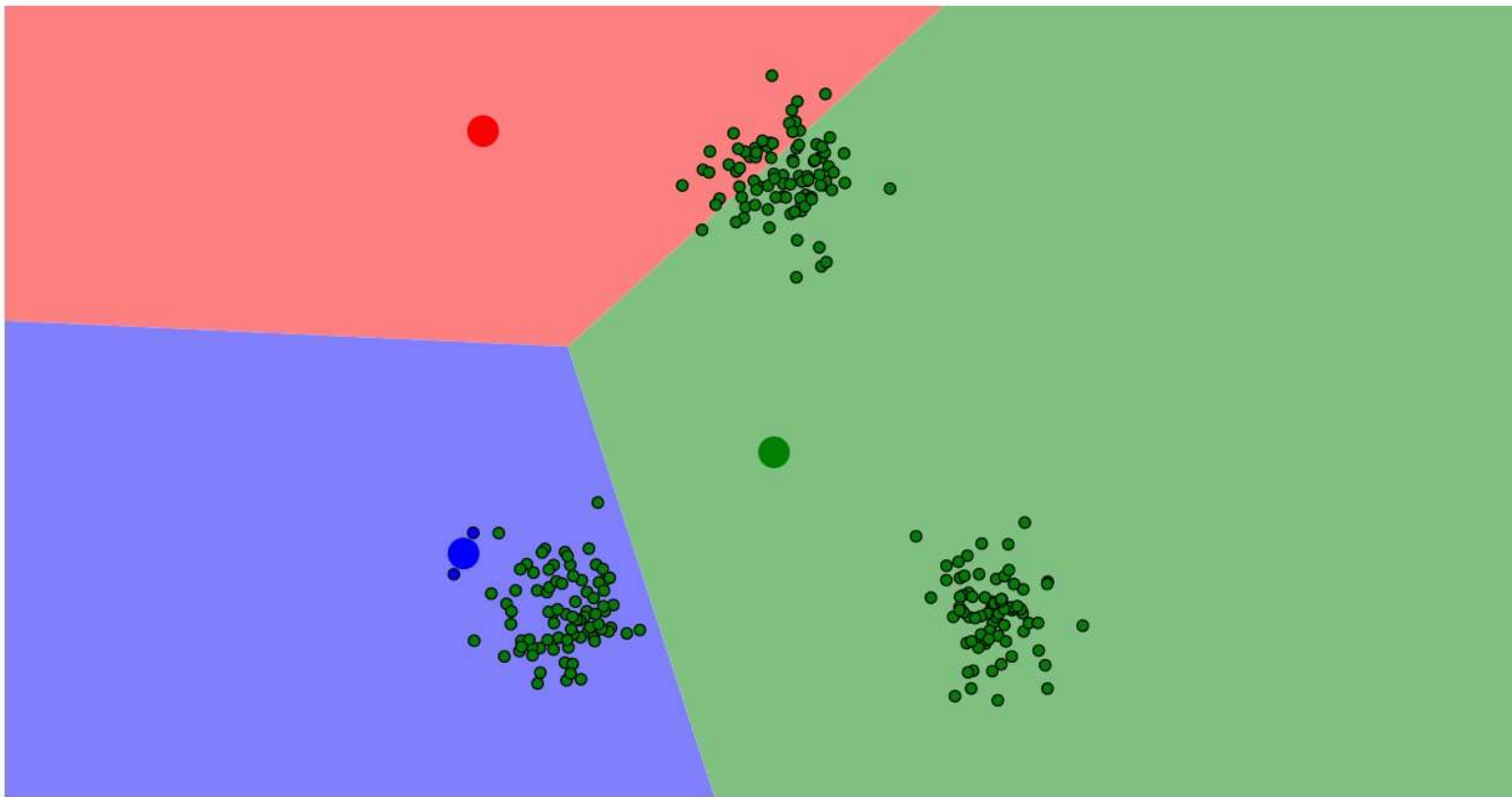
## Paso 1



**Queremos tres clústeres, por lo que se eligen tres centros al azar.**

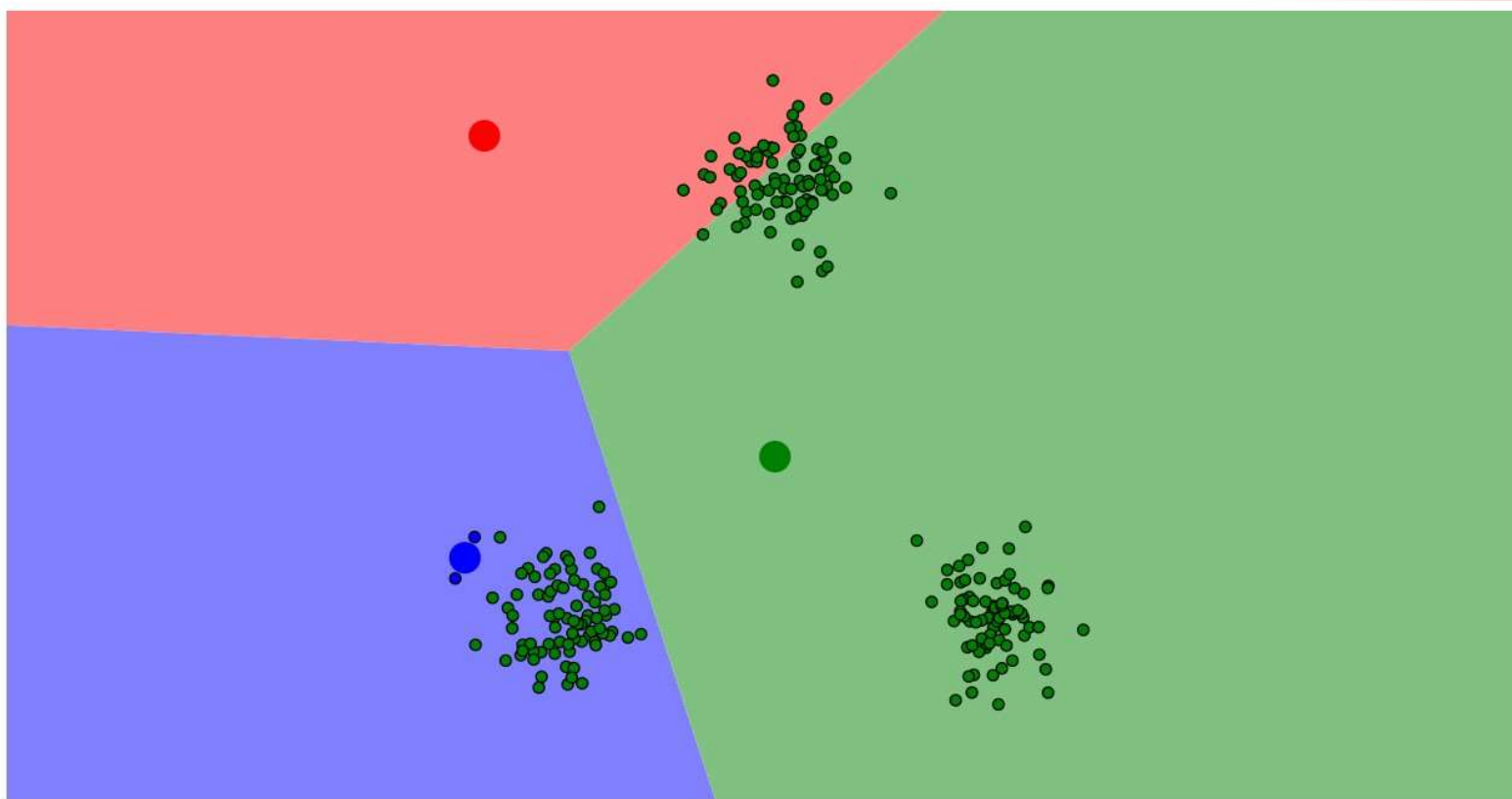
**Los puntos de datos se colorean según el centro más cercano.**

## Ejemplo de K-means: Paso 1B



**A**  
continuación,  
cada centro se  
actualiza...  
... utilizando la  
media de todos  
los puntos  
asignados a  
ese clúster.

## Paso 3

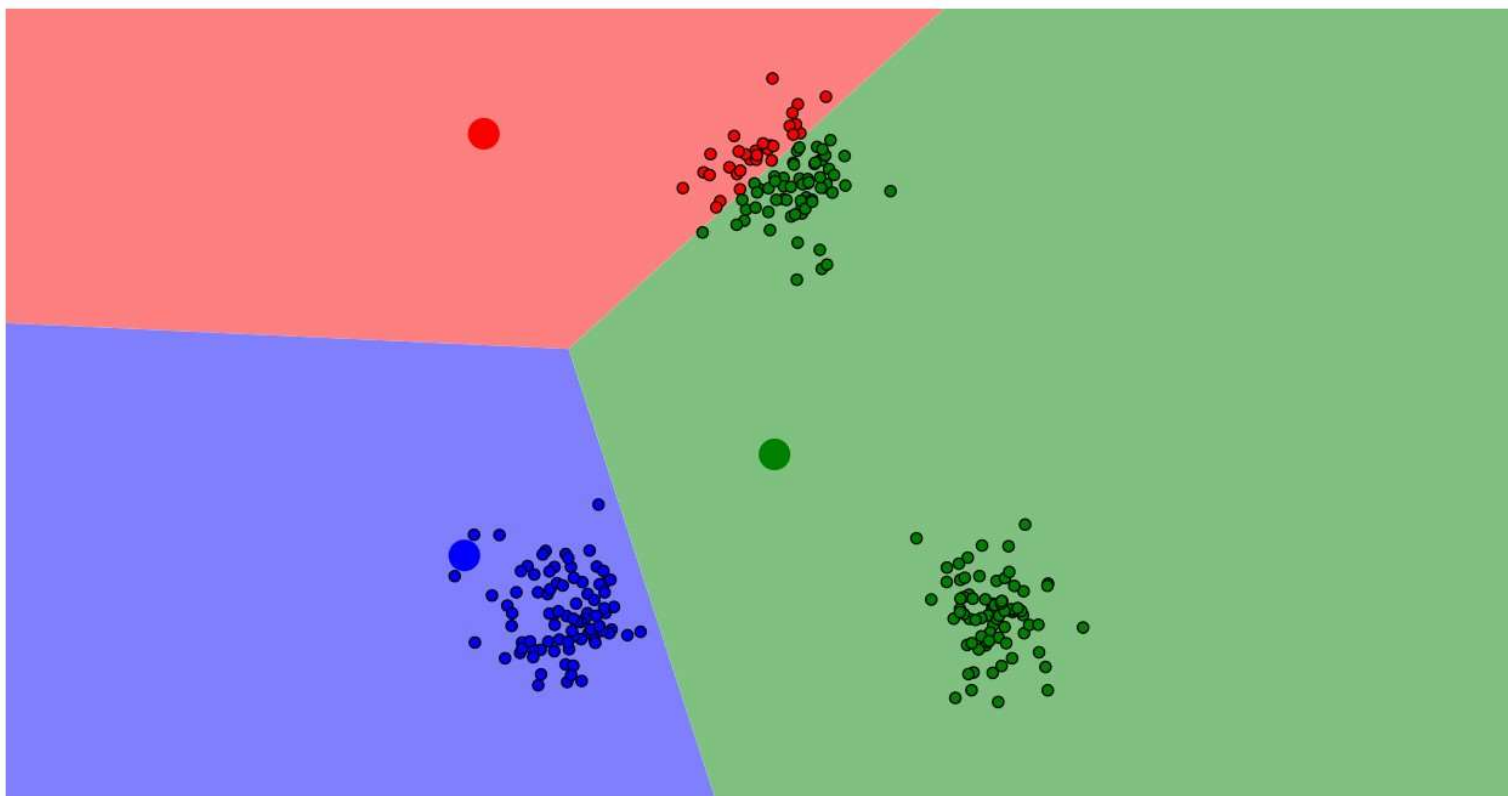


**A continuación,  
cada centro se  
actualiza...**

...

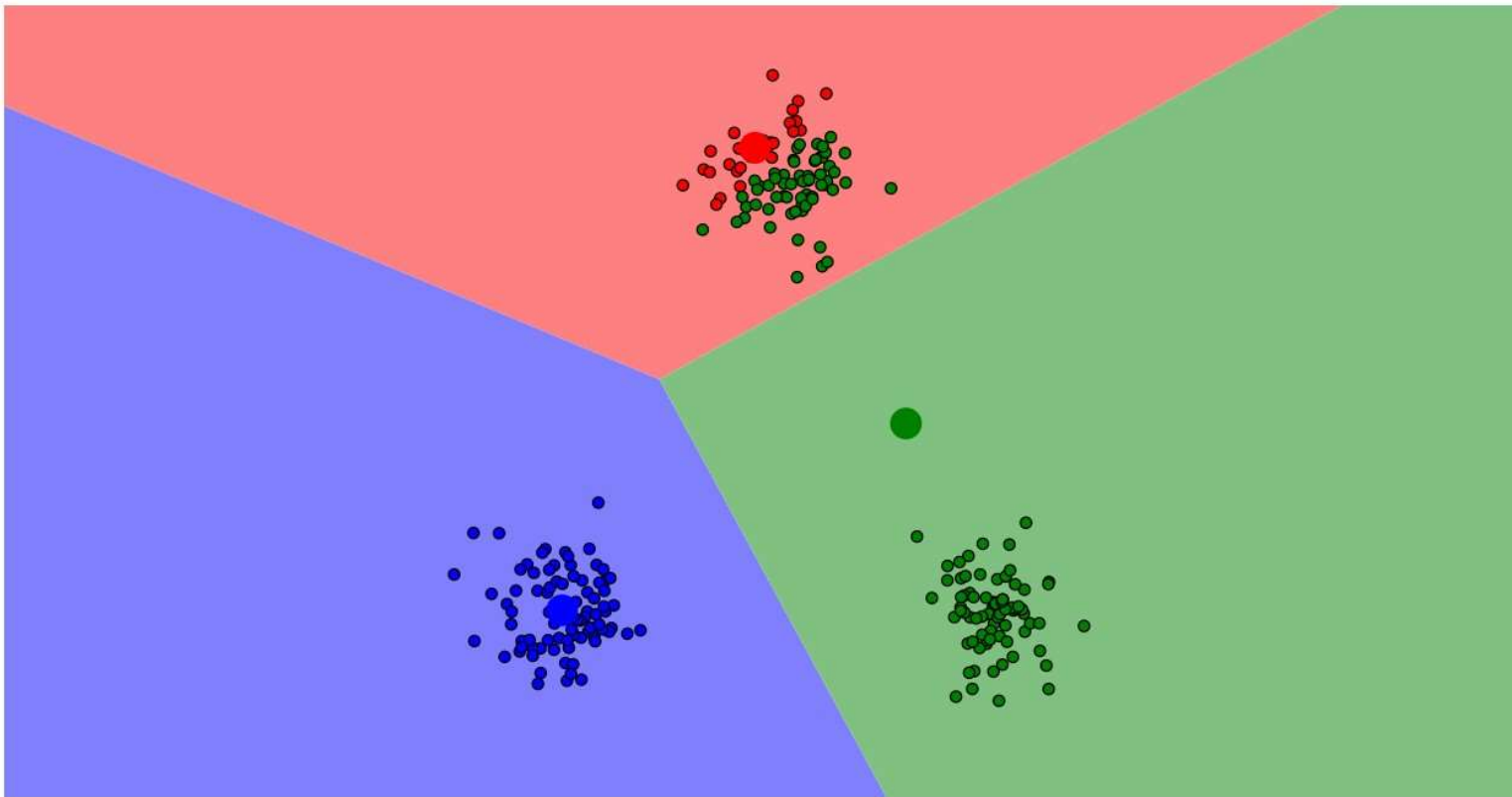
**utilizando la media  
de todos los  
puntos asignados a  
ese clúster.**

## Paso 4



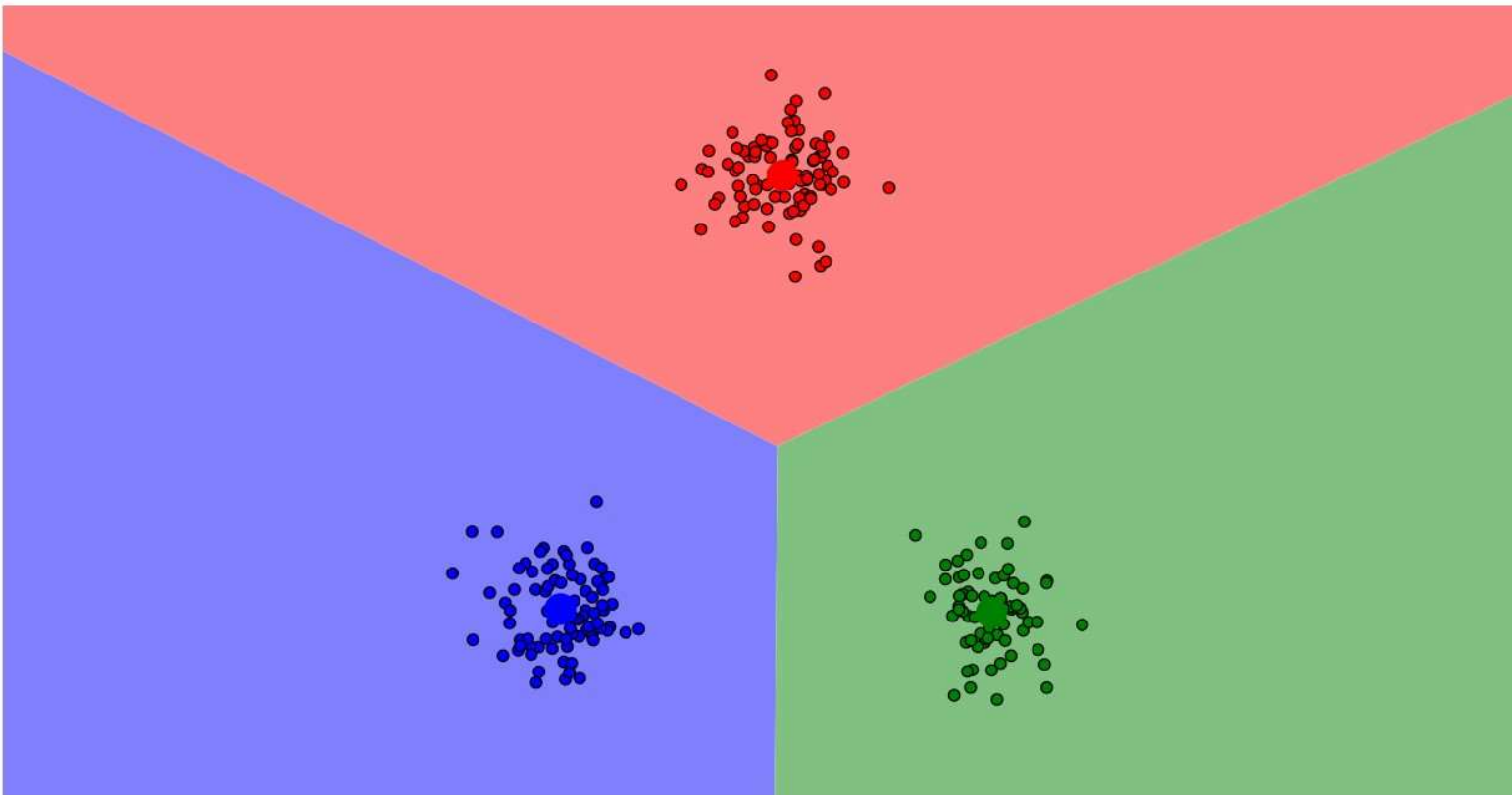
Los puntos de datos se colorean (de nuevo) según el centro más cercano.

# Ejemplo de K-means: Paso 2B



**Vuelva a  
calcular todos  
los centros de  
clúster.**

# Ejemplo de K-medias: Convergencia



**Después de repetir  
estos pasos  
durante varias  
iteraciones más...**

**¡Los centros  
convergen en una  
solución estable!  
Estos centros  
definen los  
clústeres finales.**

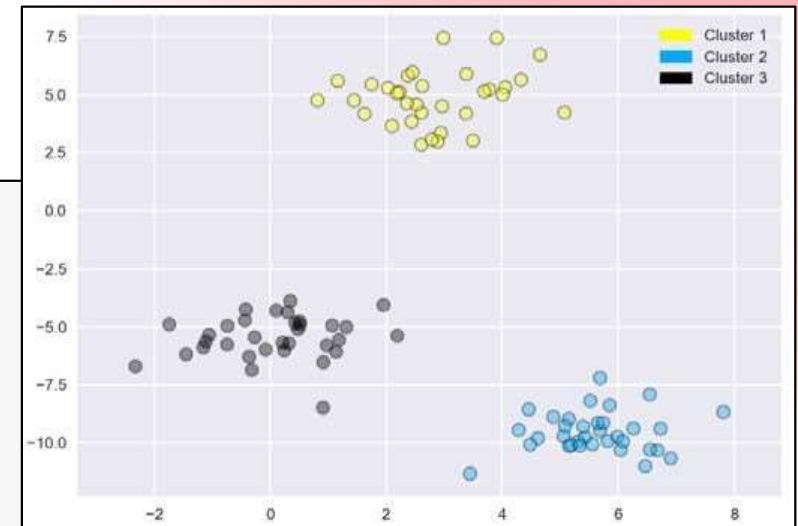
# K-means en Python

```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from adspy_shared_utilities import plot_labelled_scatter
```

```
X, y = make_blobs(random_state = 10)
```

```
kmeans = KMeans(n_clusters = 3)
kmeans.fit(X)
```

```
plot_labelled_scatter(X, kmeans.labels_, ['Cluster 1', 'Cluster 2', 'Cluster 3'])
```





# K-medias en el conjunto de datos de frutas

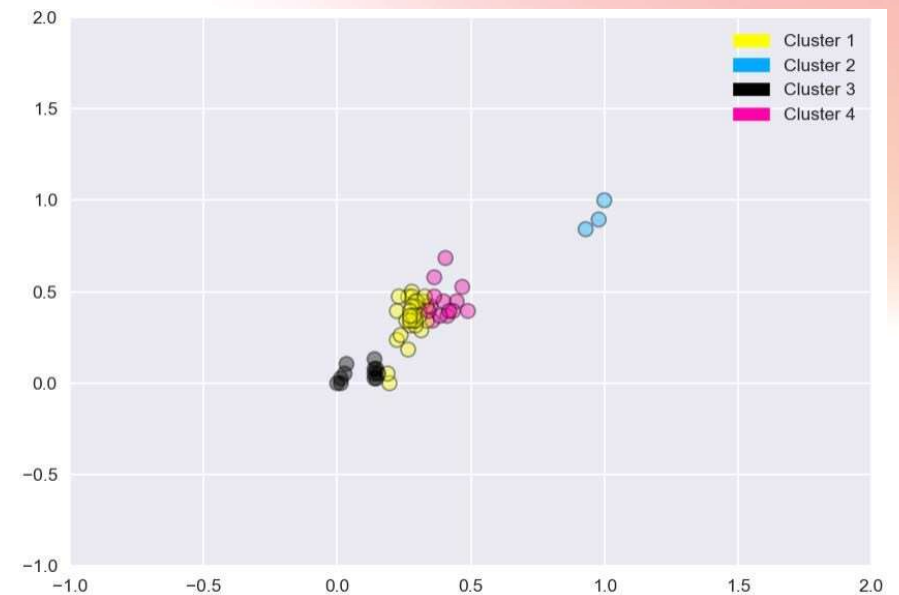
```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from adspy_shared_utilities import plot_labelled_scatter
from sklearn.preprocessing import MinMaxScaler

fruits = pd.read_table('fruit_data_with_colors.txt')
X_fruits = fruits[['mass', 'width', 'height', 'color_score']].as_matrix()
y_fruits = fruits[['fruit_label']] - 1

X_fruits_normalized = MinMaxScaler().fit(X_fruits).transform(X_fruits)

kmeans = KMeans(n_clusters = 4, random_state = 0)
kmeans.fit(X_fruits)

plot_labelled_scatter(X_fruits_normalized, kmeans.labels_,
                      ['Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4'])
```



¿Puede interpretar cómo se corresponden estos racimos con las etiquetas de las frutas verdaderas?

# Entradas de Kmeans

***n\_clusters****int, default=8*

The number of clusters to form as well as the number of centroids to generate.

***init****{'k-means++', 'random'}, callable or array-like of shape (n\_clusters, n\_features), default='k-means++'*

Method for initialization:

'k-means++' : Selecciona los centros de clústeres iniciales para la agrupación en clústeres de k-medias de una manera inteligente para acelerar la convergencia.

'random': Elija n\_clusters observaciones (filas) al azar de los datos de los centroides iniciales.

Si se pasa una matriz, debe tener la forma (n\_clusters, n\_features) y dar los centros iniciales.

Si se pasa un invocable, debe tomar los argumentos X, n\_clusters y un estado aleatorio y devolver una inicialización.

***N\_init***, **default=10**

Número de veces que se ejecutará el algoritmo k-means con diferentes semillas de centroide.

Los resultados finales serán la mejor salida de n\_init carreras consecutivas en términos de inercia.

# Salidas de Kmeans

**cluster\_centers\_***ndarray of shape (n\_clusters, n\_features)*

Coordenadas de los centros de los clústeres.

Si el algoritmo se detiene antes de converger completamente (ver tol y max\_iter), estos no serán consistentes con labels\_.

**labels\_***ndarray of shape (n\_samples,)*

Etiquetas de cada punto

**inertia\_***float*

Suma de las distancias al cuadrado de las muestras a su centro de clúster más cercano.

MIDE QUÉ TAN COMPACTO ES UN CLÚSTER, CUANTO MÁS PEQUEÑO MEJOR...

**n\_iter\_***int*

Número de iteraciones ejecutadas.

# Problemas con KMeans

- ¿Cuántos Clusters Tengo que Considerar?
  - Se debe seleccionar un numero de clusters que minimicen algún parámetro de la calidad del cluster
  - Kmeans proporciona la “inertia” como medida del error cuadrático medio: cuanto mas pequeño este valor mas compactos son los clusters ...
- Solucionado lo anterior, ¿Dónde los inicializo?
  - La posición inicial de las semillas afecta al resultado final
  - Cuando mas cerca este la semilla original del cluster final mas rápida será la convergencia
  - Kmeans++ como parámetro de inicialización soluciona el problema

# Inertia (inercia)

- La inercia mide la calidad de los clusters formados mediante K-Means. Se calcula midiendo la distancia entre cada punto de datos y su centroide:

$$\sum_{i=1}^N (x_i - C_k)^2$$

- El algoritmo K-means tiene como objetivo elegir centroides que minimicen la inercia, o el criterio de suma de cuadrados dentro del clúster. La inercia puede reconocerse como una medida de la coherencia interna de los clústeres.
- El mismo concepto se puede aplicar para identificar el número de clústeres: probar KMEANS con diferente número de clústeres y para cada uno obtener el `intertia_` y seleccionar un compromiso entre `intertia_` y número de clústeres.
- Este "codo" no siempre puede identificarse sin ambigüedades, lo que hace que este método sea muy subjetivo y poco fiable.

## Otras métricas

- SILHOUTTE
- DAVIES BOULDIN
-

# Calidad de los CLusters

- "Bien agrupado":

- Los puntos dentro del clúster están cerca unos de otros y los clústeres están bien separados



- "Poorly Clustered":

- Los puntos dentro del clúster pueden estar muy separados y los clústeres muy cerca



## Métricas para medir la calidad de la agrupación en clústeres

- En esta sección se tratan las métricas de agrupación en clústeres "internas".
- Las métricas internas implican calcular la relación entre la distancia entre puntos dentro del clúster y la distancia entre clústeres



# Davies-Bouldin Index

- Define  $S_i$  to denote cluster  $i$  and  $C_i$  to denote the center of cluster  $i$
- Compactness of cluster  $i$  is the average distance between points in cluster  $i$  and its centre

$$\text{compact}(S_i) = \frac{1}{|S_i|} \sum_{X \in S_i} \text{dist}(C_i, X)$$

- Distance between clusters is defined as distance between cluster centres:  $M_{ij} = \text{dist}(C_i, C_j)$
- Define  $D$  matrix as

$$D_{ij} = \frac{\text{compact}(S_i) + \text{compact}(S_j)}{M_{ij}} \quad i \neq j \quad D_{ii} = 0$$

This entry is ratio of compactness for clusters  $i$  and  $j$  to the distance between them

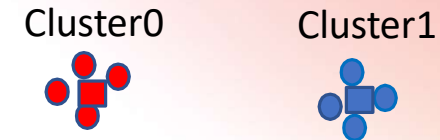
- Davies-Bouldin Index defined ( $N$  is number of clusters)

$$DB = \frac{1}{N} \sum_i \max_j D_{ij}$$

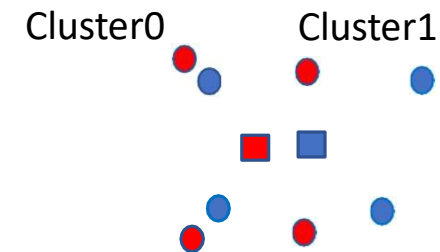
## Examples

- Davies-Bouldin Index close to 0 indicates well separated, compact clusters
- Davies-Bouldin Index  $\gg 1$  indicates poorly separated clusters
- Well separated “compact” clusters

- $compact(clus_0), compact(clus_1) < dist(C_0, C_1)$
- DB Index  $< 1$



- Not well separated clusters not compact clusters
- $compact(clus_0), compact(clus_1) > dist(C_0, C_1)$
- DB Index  $> 1$



# Silhouette Index

- Silhouette index is defined for each point in the dataset and index value for entire dataset is mean of these individual values.
- Silhouette index is between -1 and 1
- Silhouette index is 0 for cluster with 1 point
- For  $X_i$  in cluster  $S_i$  with more than 1 point, define (avg distance to other points in cluster):

# Silhouette Index

- For  $X_i$  in cluster  $S_i$  with more than 1 point, define (avg distance to other points in cluster):

$$a(X_i) = \frac{1}{|S_i| - 1} \sum_{X \in S_i} \text{dist}(X_i, X)$$

- Define minimum avg distance to points within other clusters as:

$$b(X_i) = \min_{k \neq i} \frac{1}{|S_k|} \sum_{X \in S_k} \text{dist}(X_i, X)$$

- Silhouette index for  $X_i$  defined as:

$$\text{Silhouette}(X_i) = \frac{b(X_i) - a(X_i)}{\max(a(X_i), b(X_i))}$$

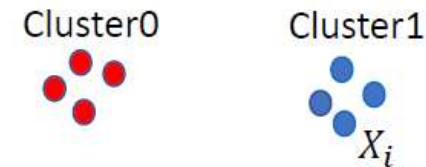
## Examples

- Silhouette index near 1 indicates well separated clusters
- Silhouette index near -1 indicates poorly separated clusters

- Well separated “compact” clusters

- $a(X_i) \ll b(X_i)$

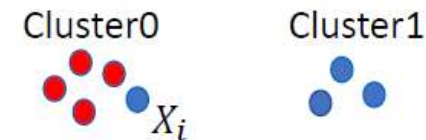
- $Silhouette(X_i) = \frac{b(X_i) - a(X_i)}{\max(a(X_i), b(X_i))} \approx 1$



- Not well separated clusters

- $a(X_i) \gg b(X_i)$

- $Silhouette(X_i) = \frac{b(X_i) - a(X_i)}{\max(a(X_i), b(X_i))} \approx -1$



## Agglomerative Clustering

- También se denomina análisis de conglomerados jerárquicos o HCA
- Método que crea una jerarquía de clústeres.
- Tipos:
  - Aglomeración: De abajo hacia arriba: Cada observación comienza en su propio clúster, y los pares de grupos se fusionan a medida que uno asciende en la jerarquía.
  - Divisoria: De arriba hacia abajo: Todas las observaciones comienzan en un grupo y las divisiones se realizan de forma recursiva a medida que uno se desplaza hacia abajo en la jerarquía.

# Agglomerative Clustering

- **Método Jerárquico:**

Agrupar objetos en clusters de forma progresiva, partiendo de puntos individuales hasta formar grupos más grandes, creando una estructura jerárquica.

- **Enfoque Bottom-Up:** Comienza con cada punto como un cluster independiente, fusionando iterativamente los clusters más cercanos hasta obtener la cantidad deseada de grupos.

- **Ventajas:**

- No necesita definir inicialmente el número exacto de clusters.
- Produce un dendrograma (visualización de jerarquía), que permite decidir visualmente el número óptimo de clusters.

## Agglomerative Clustering

**1.Inicialización:** Cada punto es un cluster individual.

**2.Medición de distancia:** Calcula la distancia entre clusters usando métricas como:

1. Enlace único (**single linkage**): distancia entre puntos más cercanos.
2. Enlace completo (**complete linkage**): distancia entre puntos más alejados.
3. Enlace promedio (**average linkage**): distancia promedio entre todos los puntos de dos clusters.

**3.Fusión Iterativa:** Fusiona los dos clusters más cercanos en cada iteración.

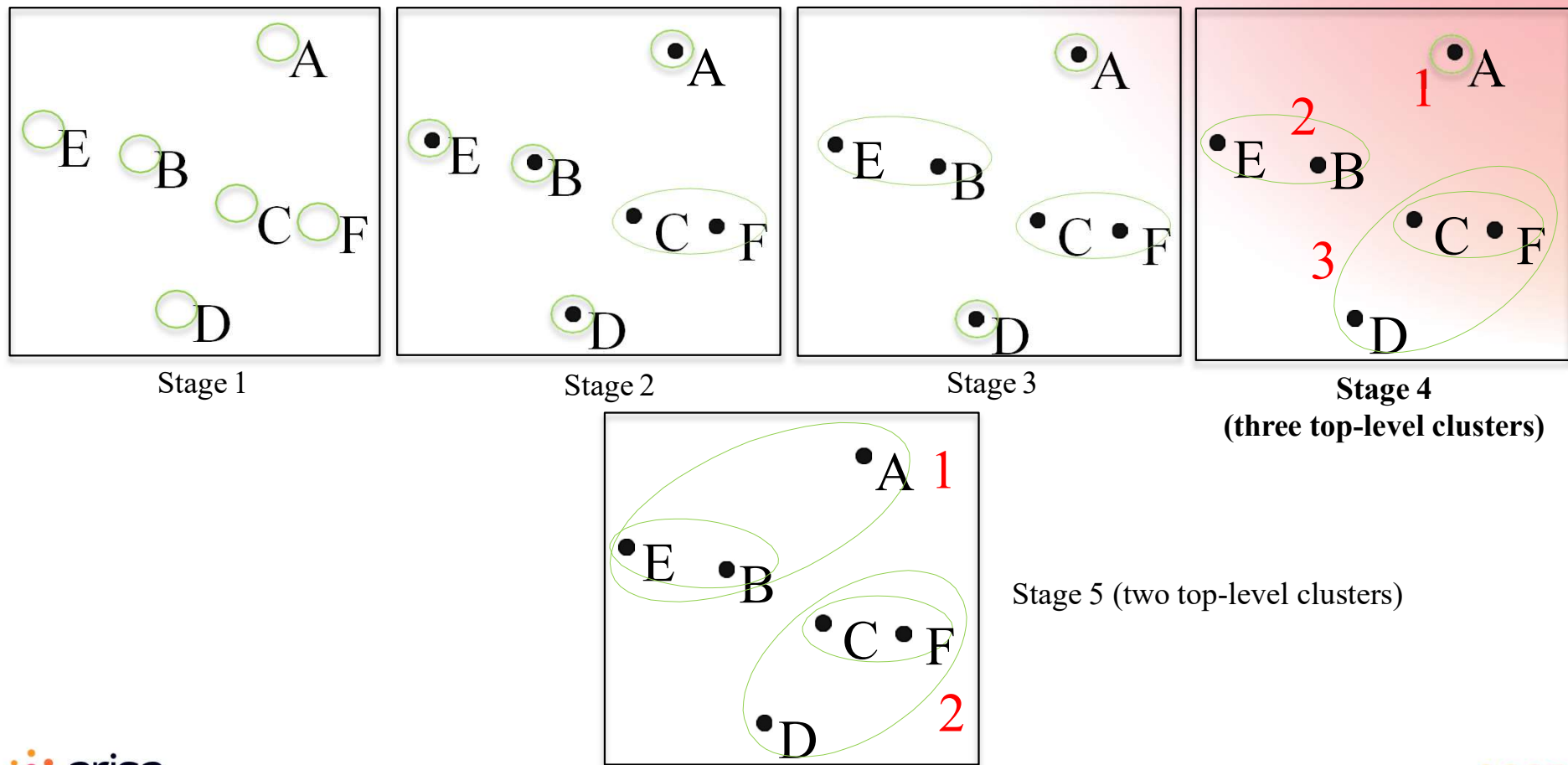
**4.Repetición:** Continúa fusionando clusters hasta alcanzar la cantidad deseada o hasta obtener un dendrograma completo.

### Resultado:

- Un dendrograma que muestra claramente cómo los clusters están relacionados.
- Clusters bien definidos según la proximidad y la métrica escogida.

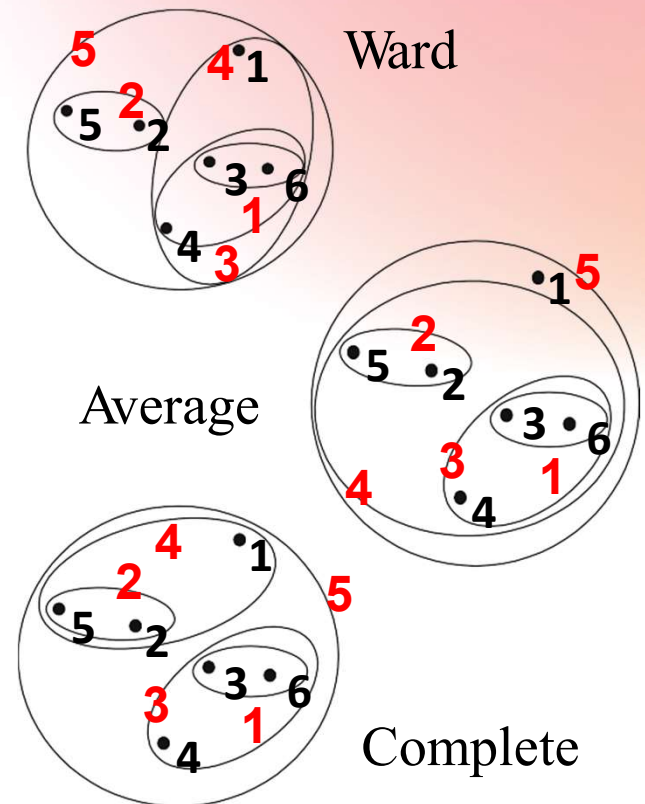
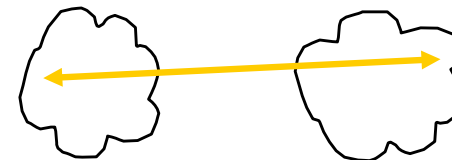
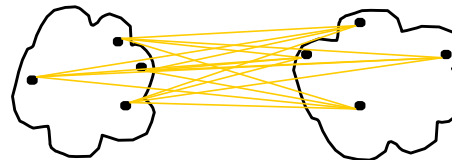
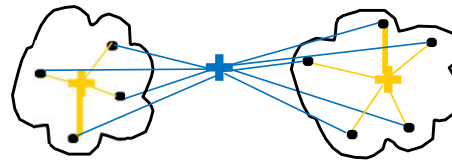


## Agglomerative Clustering Example



## Agglomerative Clustering: Linkage

- **Ward's method**
  - *Least increase in total variance (around cluster centroids)*
- **Average linkage**
  - *Average distance between clusters*
- **Complete linkage**
  - *Max distance between clusters*



## Agglomerative Clustering

```
from sklearn.datasets import make_blobs
from sklearn.cluster import AgglomerativeClustering
from adspy_shared_utilities import plot_labelled_scatter

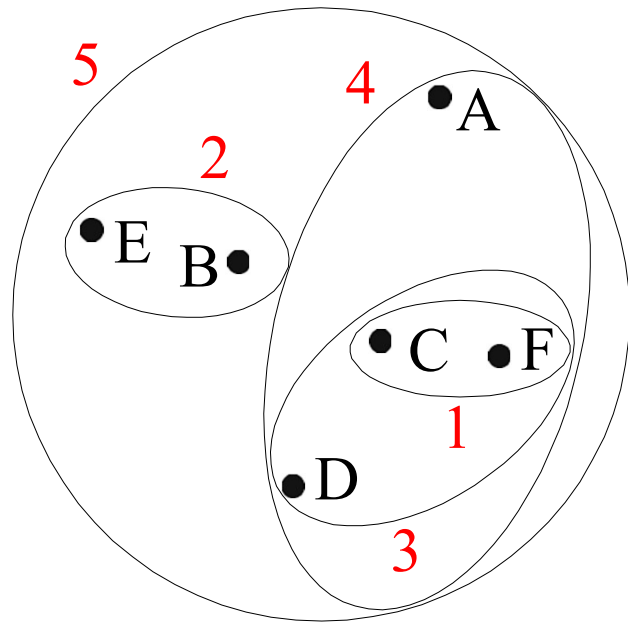
X, y = make_blobs(random_state = 10)

cls = AgglomerativeClustering(n_clusters = 3)
cls_assignment = cls.fit_predict(X)

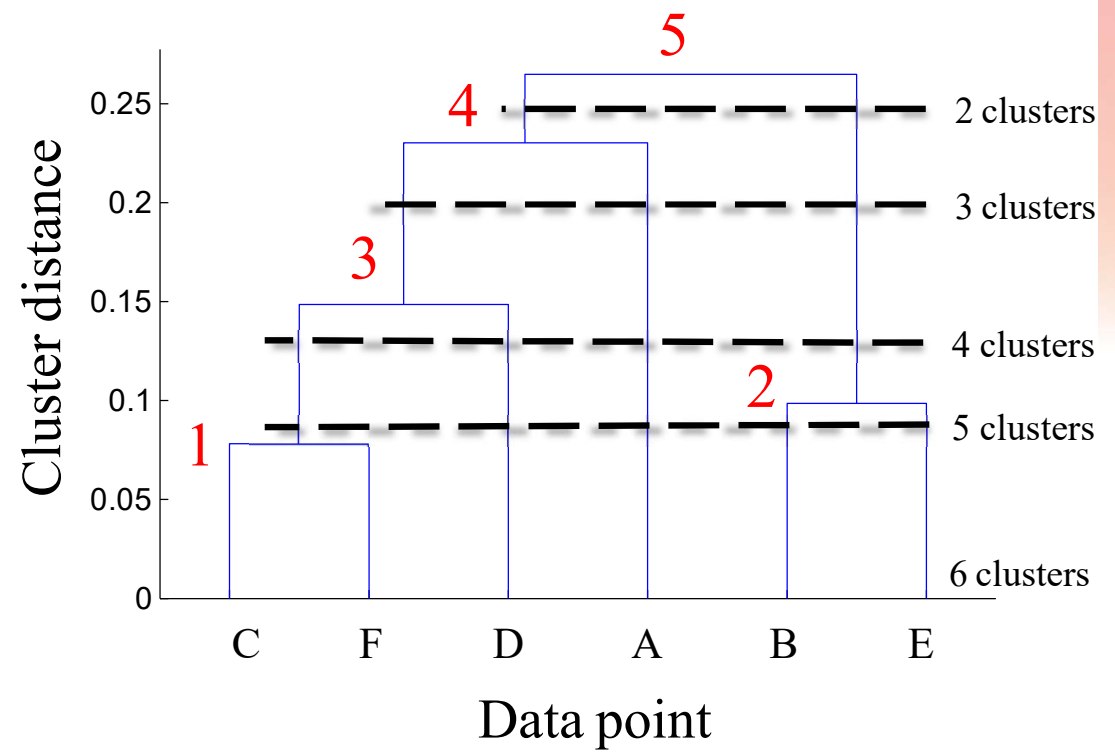
X, y = make_blobs(random_state = 10)
plot_labelled_scatter(X, cls_assignment,
                      ['Cluster 1', 'Cluster 2', 'Cluster 3'])
```



## Hierarchical Clustering



## Dendrogram

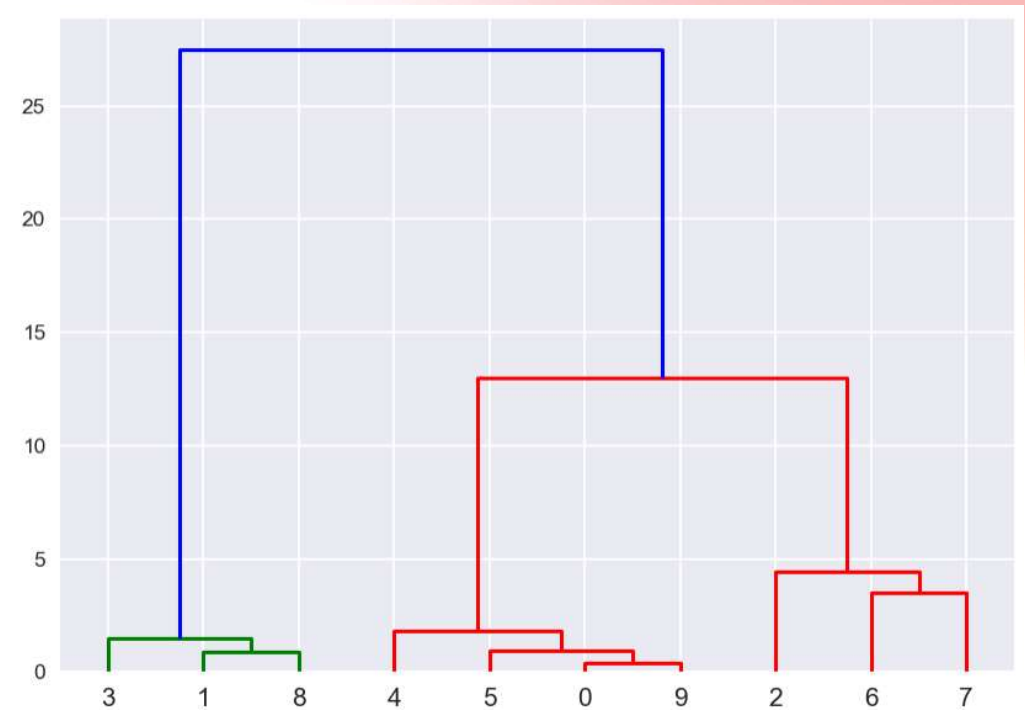


# Hyerarchical Clustering

```
from scipy.cluster.hierarchy import ward, dendrogram
from sklearn.datasets import make_blobs
from sklearn.cluster import AgglomerativeClustering

X, y = make_blobs(random_state = 10, n_samples = 10)

plt.figure()
dendrogram(ward(X))
plt.show()
```



# DBSCAN

DBSCAN is a density-based algorithm.

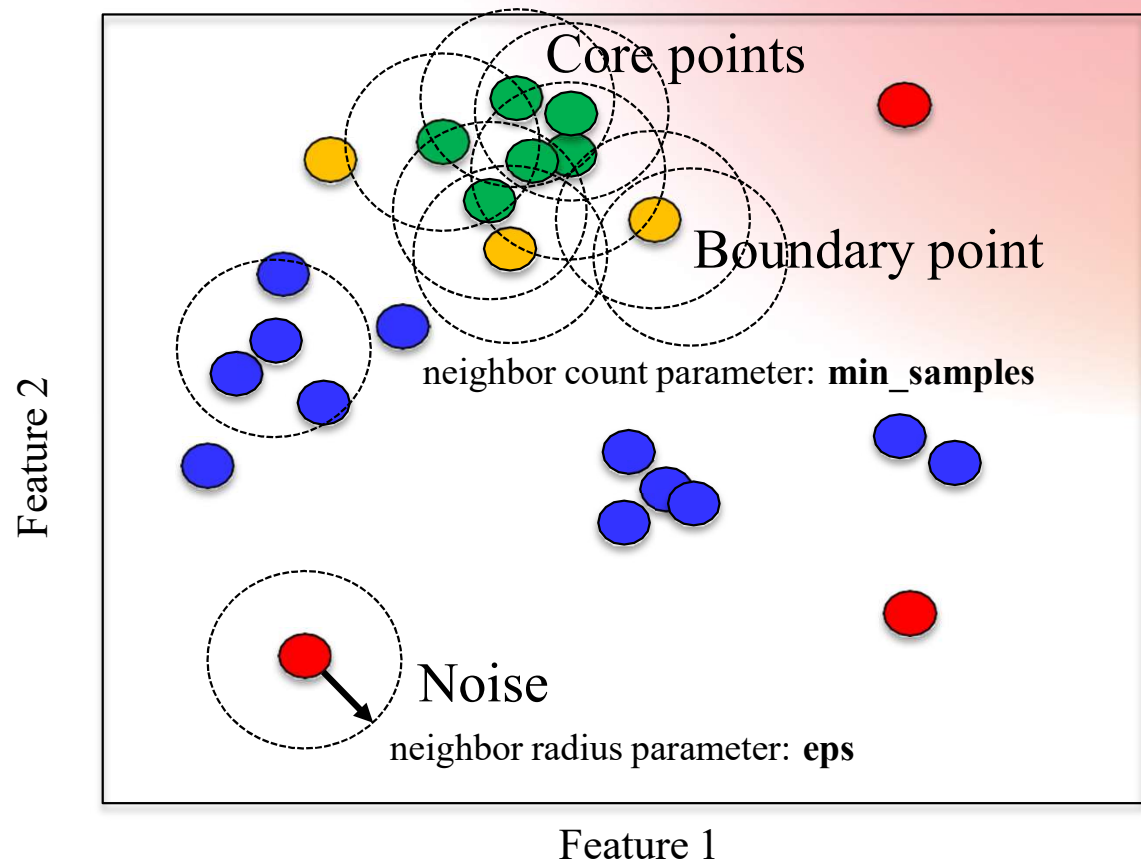
- Density = number of points within a specified radius  $r$  (Eps)
- A point is a **core point** if it has more than a specified number of points (MinPts) within Eps

These are points that are at the interior of a cluster

- A **border point** has fewer than MinPts within Eps, but is in the neighborhood of a core point
- A **noise point** is any point that is not a core point or a border point.

## DBSCAN Clustering

- A diferencia de k-means, no es necesario especificar # de clústeres
- Relativamente eficiente: se puede utilizar con grandes conjuntos de datos
- Identifica los puntos de ruido probables

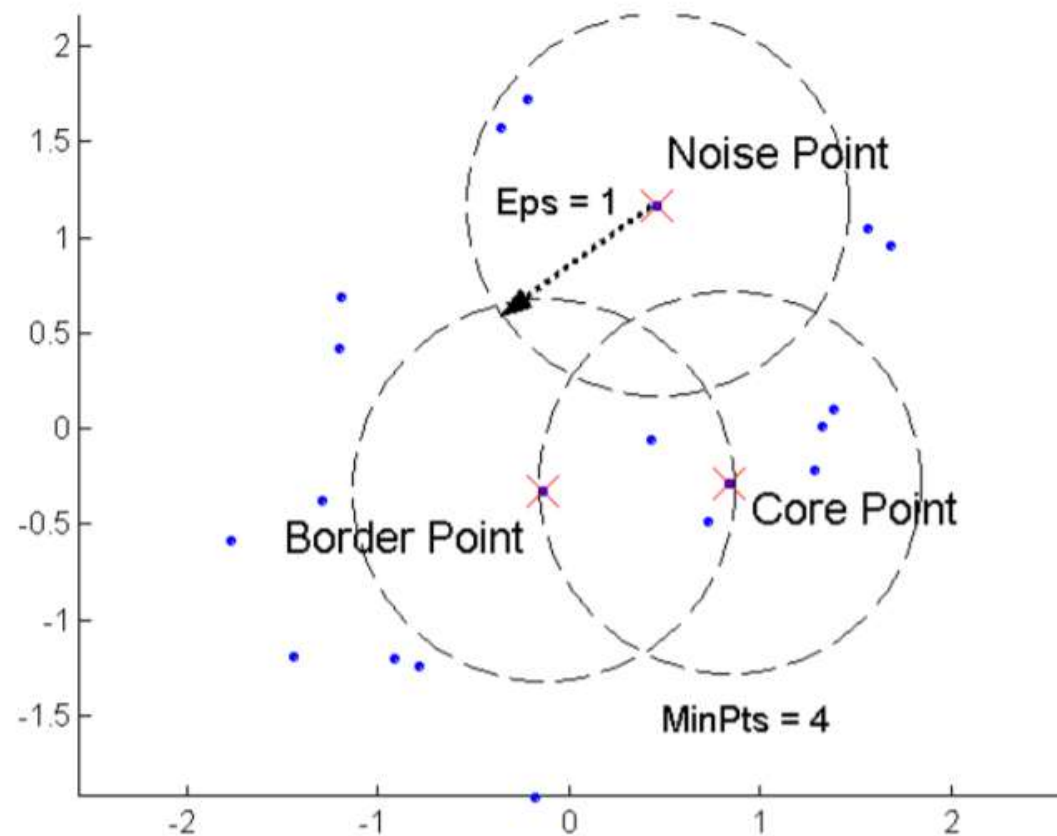


## DBSCAN Clustering

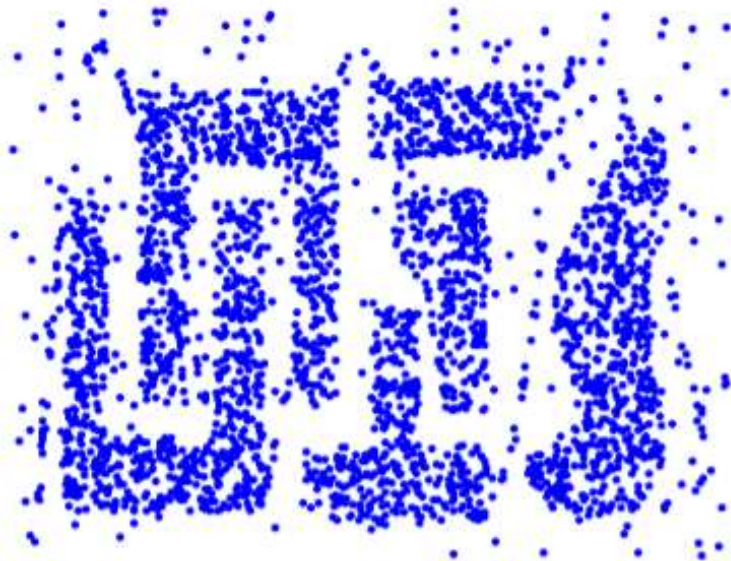
- **$\epsilon$  (epsilon):** Radio de búsqueda alrededor de cada punto.
- **minPts (mínimo de puntos):** Número mínimo de puntos que deben estar dentro de un radio  $\epsilon$  para considerar que un punto es central en un cluster.
- Funcionamiento del algoritmo:
  - 1- Se elige un punto no visitado.
  - 2- Se cuentan los puntos dentro de su radio  $\epsilon$ : Si tiene al menos minPts, es un punto central y se expande un nuevo cluster. Si tiene menos de minPts, se marca como ruido, pero puede convertirse en parte de un cluster más adelante.
  - 3- Se repite hasta visitar todos los puntos.



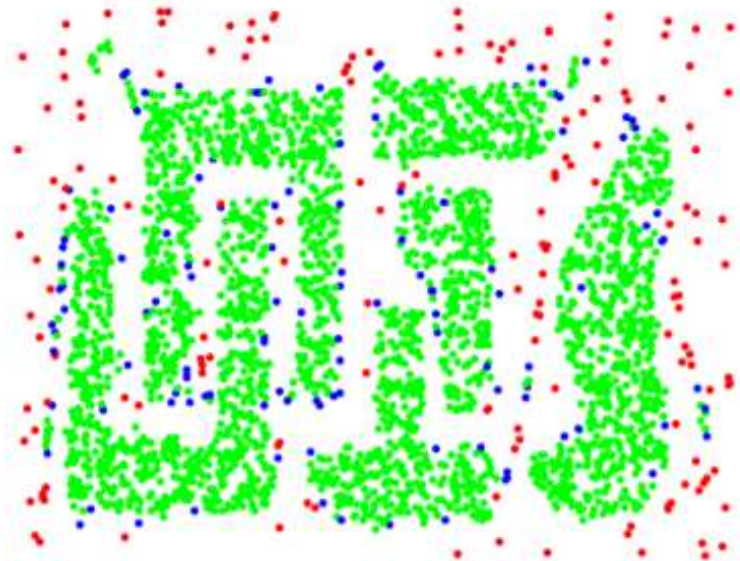
# DBSCAN: Core, Border, and Noise points



# DBSCAN: Large Eps



Original Points

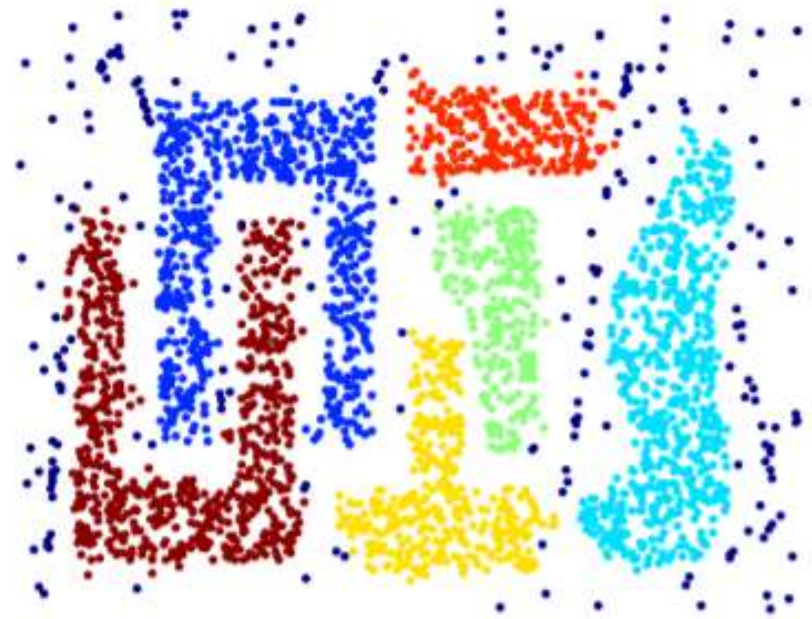


Point types: **core**,  
**border** and **noise**

# DBSCAN: Optimal Eps



Original Points



Clusters

# Entradas

- DBSCAN detecta el numero de clusters automáticamente
- Eps: existen mecanismos heurísticos para determinarlo
- Min\_samples: no suele ser grande (4,5 o  $2 \times \text{dimension}$ )

# Min\_points

- Cuanto mayor sea el conjunto de datos, mayor debe ser el valor de MinPts
- Si el conjunto de datos es más ruidoso, elija un valor mayor de MinPts
- Por lo general, los MinPts deben ser mayores o iguales que la dimensionalidad del conjunto de datos
- Para datos bidimensionales, utilice el valor predeterminado de DBSCAN de MinPts = 4 (Ester et al., 1996).
- Si los datos tienen más de 2 dimensiones, elija  $\text{MinPts} = 2 * \text{dim}$ , donde dim = las dimensiones del conjunto de datos (Sander et al., 1998).

# Valor óptimo de $\varepsilon$

- Este tema ya es de investigación.
- Esta técnica calcula la distancia media entre cada punto y sus  $k$  vecinos más cercanos, donde  $k$  = los MinPts.
- A continuación, las  $k$ -distancias medias se trazan en orden ascendente en un gráfico de  $k$ -distancias.
- Encontrarás el valor óptimo para  $\varepsilon$  en el punto de máxima curvatura (es decir, donde el gráfico tiene la mayor pendiente).

# Nearest Neighbors

- Aprendizaje no supervisado para implementar búsquedas de vecinos.
- `sklearn.neighbors.NearestNeighbors`
  - `n_neighbors` (int, default=5): Number of neighbors to use by default for kneighbors queries.
  - `NearestNeighbors` implementa el aprendizaje no supervisado de los vecinos más cercanos. Actúa como una interfaz uniforme para tres algoritmos diferentes de vecinos más cercanos: `BallTree`, `KDTree` y un algoritmo de fuerza bruta basado en rutinas en `sklearn.metrics.pairwise`. La elección del algoritmo de búsqueda de vecinos se controla a través de la palabra clave 'algoritmo', que debe ser una de ['auto', 'ball\_tree', 'kd\_tree', 'bruto']. Cuando se pasa el valor predeterminado 'auto', el algoritmo intenta determinar el mejor enfoque a partir de los datos de entrenamiento.



# DBSCAN Clustering

```
from sklearn.cluster import DBSCAN
from sklearn.datasets import make_blobs

X, y = make_blobs(random_state = 9, n_samples = 25)

dbscan = DBSCAN(eps = 2, min_samples = 2)

cls = dbscan.fit_predict(X)
print("Cluster membership values:\n{}", format(cls))

plot_labelled_scatter(X, cls + 1,
                      ['Noise', 'Cluster 0', 'Cluster 1', 'Cluster 2'])
```

Cluster membership values:  
{ } [ 0 1 0 2 0 0 0 2 2 -1 1 2 0 0 -1 0 0 1 -1 1 1 2 2 2 1]





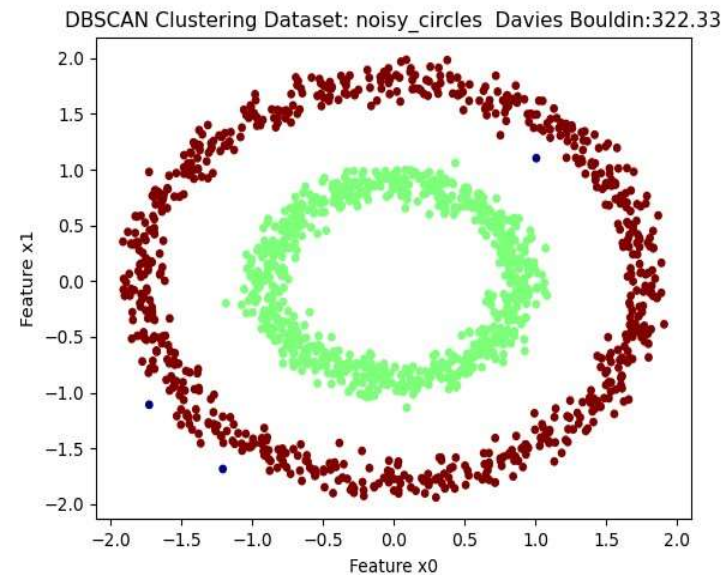
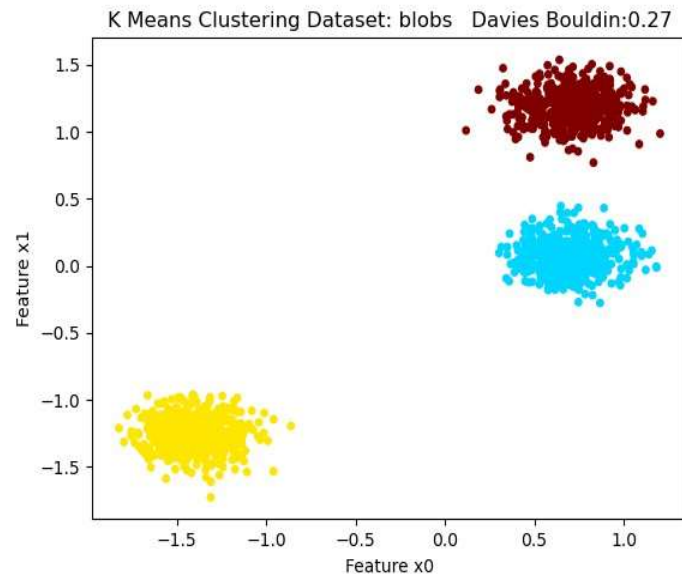
# Examples

## Example 1:

- “blobs” dataset with 1500 points using K Means (3 clusters)

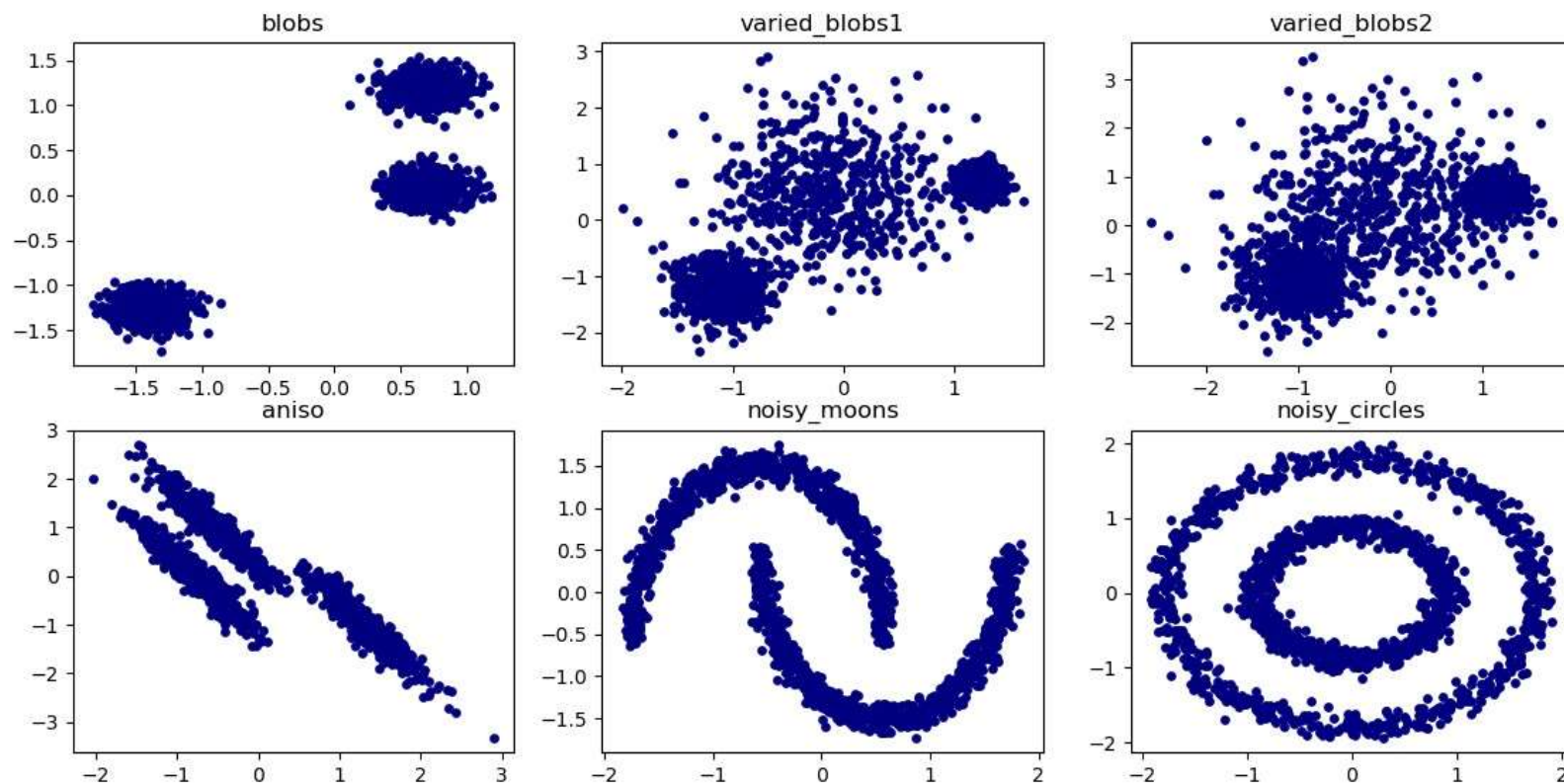
## Example 2:

- “noisy\_circles” dataset with 1500 points using DBSCAN (minpts = 5,  $\varepsilon = 0.18$ )

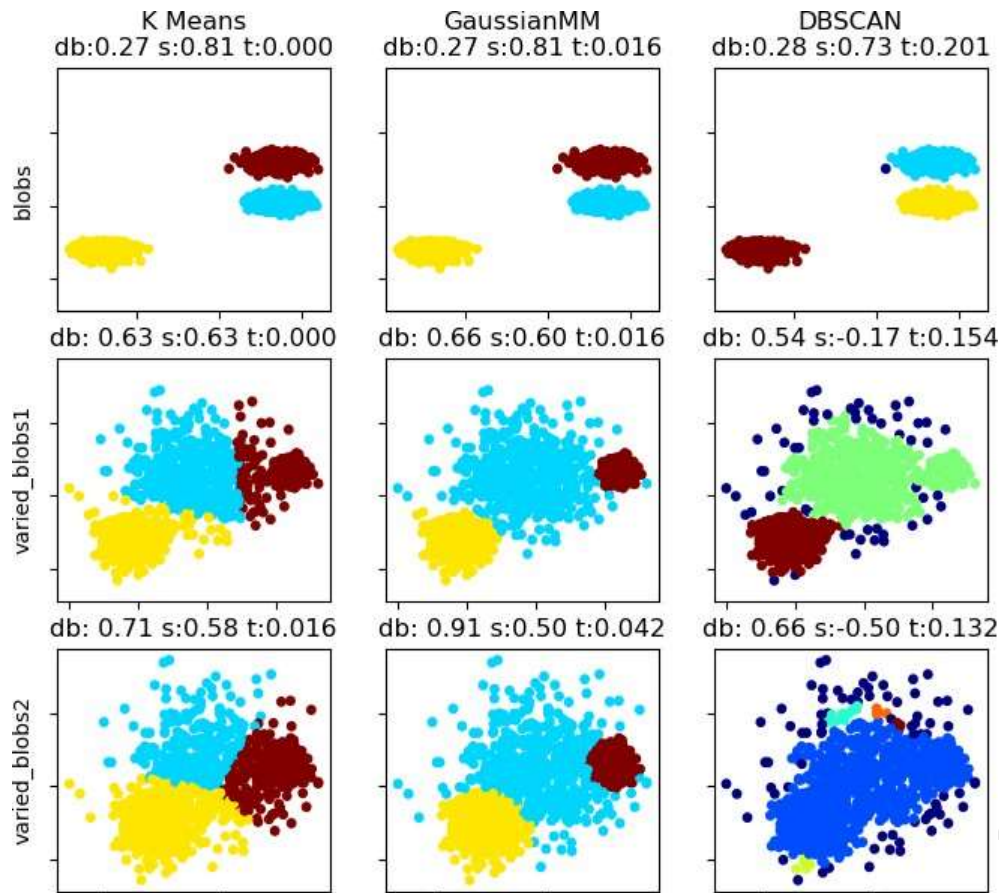


# Comparación de algoritmos: conjuntos de datos

- Conjuntos de datos de Sklearn utilizando 1500 puntos de datos

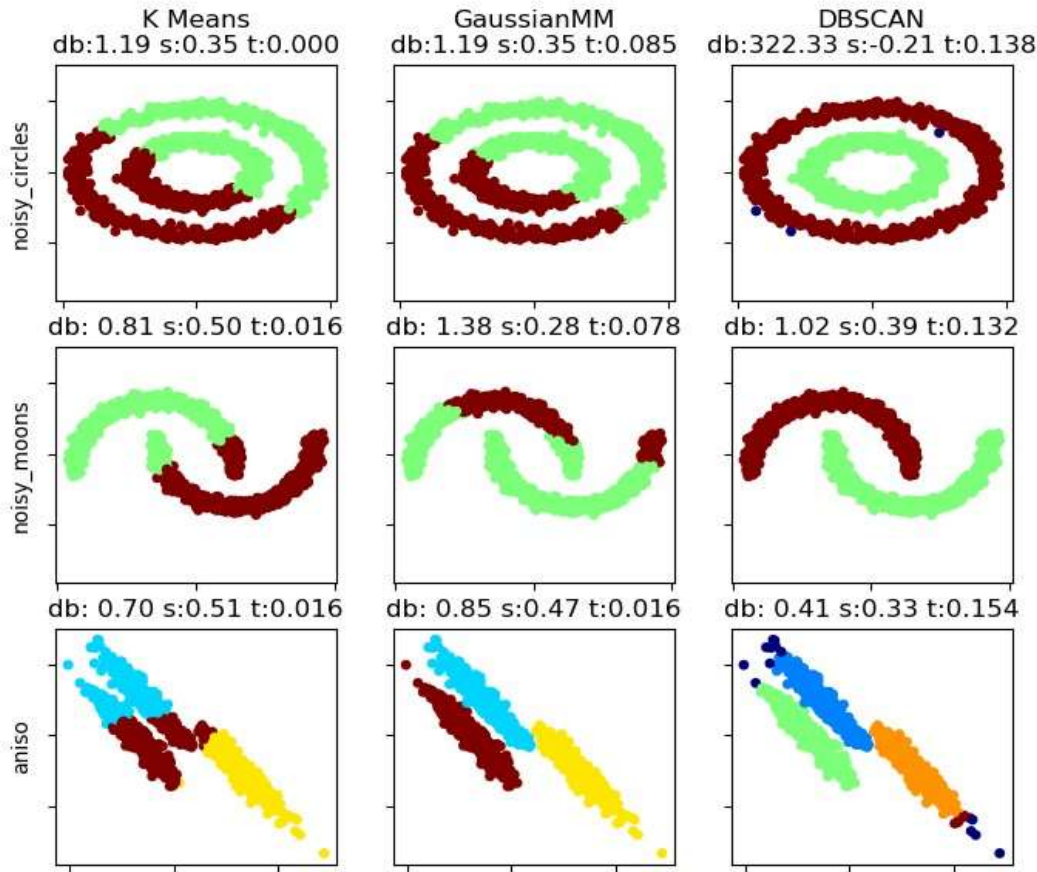


# Comparación de algoritmos: Conjunto 1



- K Means y GaussianMM:
  - Actuar de manera similar
  - K Means mas rápido que GMM
- DBSCAN: Impactado por Minpts y Epsilon:
  - Si la densidad es demasiado baja: entonces muchos puntos pertenecen a un solo clúster
  - Si la densidad es alta: entonces muchos puntos de ruido
  - No le va bien con clústeres de densidad variable

# Comparaciones de algoritmos: Conjunto 2



K-means:



No funciona bien para regiones no convexas (círculos o lunas)

No funciona bien para alargados  
Regiones (ANISO)

DBSCAN:

Puede manejar regiones no convexas



#AIskills4all |  @AIskillsEU |  [linkedin.com/AIskillsEU](https://www.linkedin.com/AIskillsEU)



[www.aiskills.eu](https://www.aiskills.eu)

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.