

---

# Nociones de Aprendizaje Automático para Aplicaciones Nucleares

Aprendizaje No Supervisado

---

# Aprendizaje no supervisado

---

- Aunque la mayoría de las aplicaciones actuales del aprendizaje automático se basan en el aprendizaje supervisado, la mayoría de los datos disponibles (y más en esta área) son datos **no etiquetados**.
- La tarea de etiquetar datos es costosa, compleja, y a menudo demanda mucho más tiempo y esfuerzo que el dedicado a probar algoritmos de aprendizaje automático y desarrollar una solución.
- A esto se suma que, en términos generales, mientras más complejo sea el problema que deseamos abordar mayor será la cantidad de datos necesarios. La demanda de datos incrementa tanto con la complejidad del problema como con la sofisticación de los métodos utilizados.
- Acá es donde entra el **aprendizaje no supervisado**.

# Aprendizaje no supervisado

---

- **Reducción de la dimensionalidad:** simplificación de datos eliminando características redundantes o poco informativas manteniendo la mayor cantidad de información posible.

# Aprendizaje no supervisado

---

- **Reducción de la dimensionalidad:** simplificación de datos eliminando características redundantes o poco informativas manteniendo la mayor cantidad de información posible.
- ***Clustering*** (agrupamiento): busca agrupar instancias similares en *clusters*. Piensen en cosas como segmentación de clientes, sistemas de recomendación, agrupar las fotos de nuestras mascotas en el celular, etc.

# Aprendizaje no supervisado

---

- **Reducción de la dimensionalidad:** simplificación de datos eliminando características redundantes o poco informativas manteniendo la mayor cantidad de información posible.
- ***Clustering*** (agrupamiento): busca agrupar instancias similares en *clusters*. Piensen en cosas como segmentación de clientes, sistemas de recomendación, agrupar las fotos de nuestras mascotas en el celular, etc.
- **Detección de anomalías:** si se aprende la generalidad de cómo son los datos “normales” se pueden identificar los que no lo sean.



# Aprendizaje no supervisado

---

- **Reducción de la dimensionalidad:** simplificación de datos eliminando características redundantes o poco informativas manteniendo la mayor cantidad de información posible.
- **Clustering** (agrupamiento): busca agrupar instancias similares en *clusters*. Piensen en cosas como segmentación de clientes, sistemas de recomendación, agrupar las fotos de nuestras mascotas en el celular, etc.
- **Detección de anomalías:** si se aprende la generalidad de cómo son los datos “normales” se pueden identificar los que no lo sean.
- **Estimación de densidad:** se busca estimar la función de densidad de probabilidad del proceso aleatorio que generó el conjunto de datos.

# Aprendizaje no supervisado

---

- **Reducción de la dimensionalidad:** simplificación de datos eliminando características redundantes o poco informativas manteniendo la mayor cantidad de información posible.
- **Clustering** (agrupamiento): busca agrupar instancias similares en *clusters*. Piensen en cosas como segmentación de clientes, sistemas de recomendación, agrupar las fotos de nuestras mascotas en el celular, etc.
- **Detección de anomalías:** si se aprende la generalidad de cómo son los datos “normales” se pueden identificar los que no lo sean.
- **Estimación de densidad:** se busca estimar la función de densidad de probabilidad del proceso aleatorio que generó el conjunto de datos.

Estos métodos se suelen usar como herramientas en el análisis de datos o como un paso de preprocesamiento previo a aplicar otros algoritmos de AA

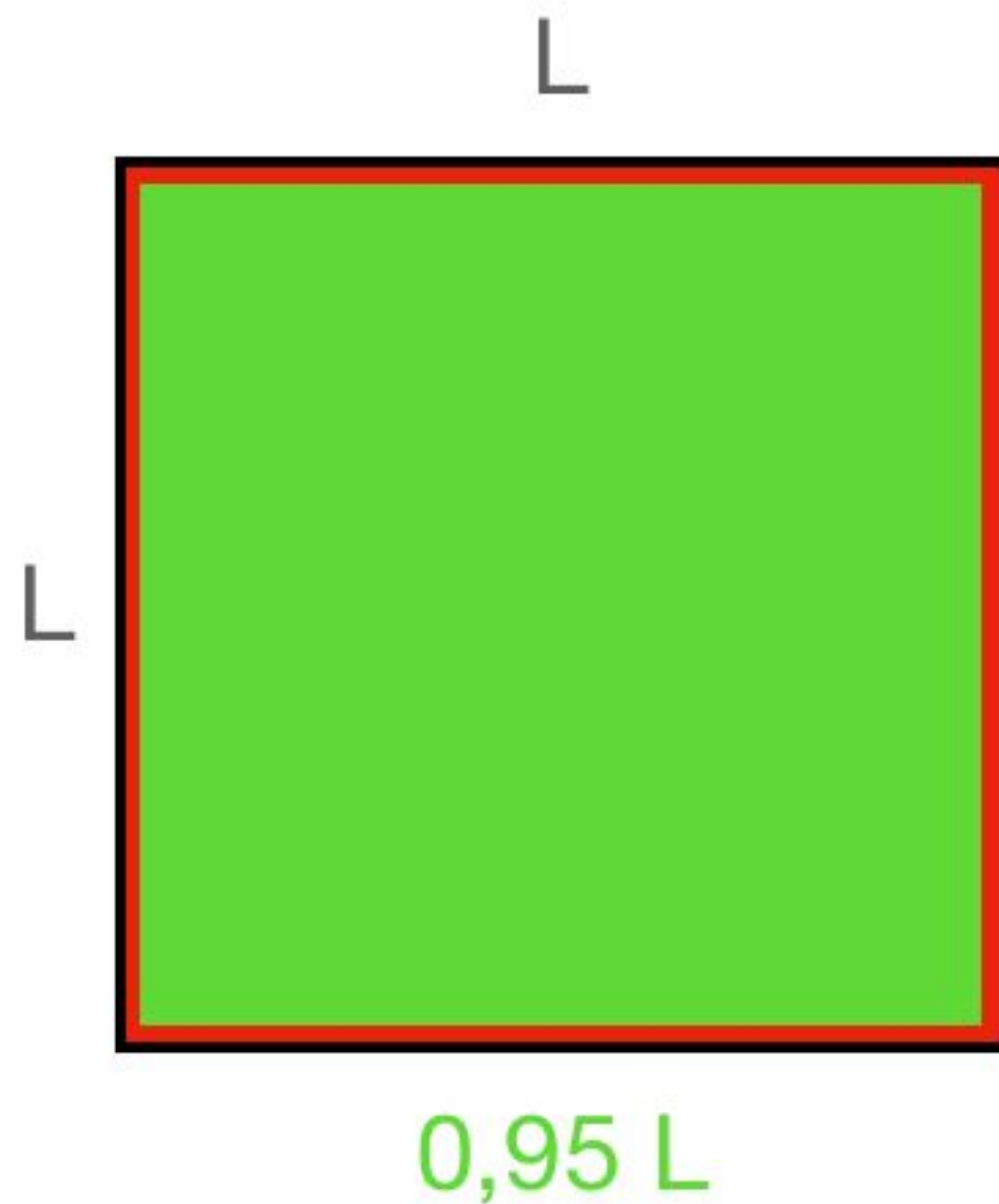
# Reducción de la dimensionalidad

---

- Muchos problemas de aprendizaje automático implican cientos o incluso miles de características para cada instancia de entrenamiento.
- Utilizar todas estas características no sólo pueden hacer que el entrenamiento sea extremadamente lento, sino que también pueden hacer que sea mucho más difícil encontrar una buena solución.
- Este problema se conoce como **la maldición de la dimensionalidad**.



# Reducción de la dimensionalidad

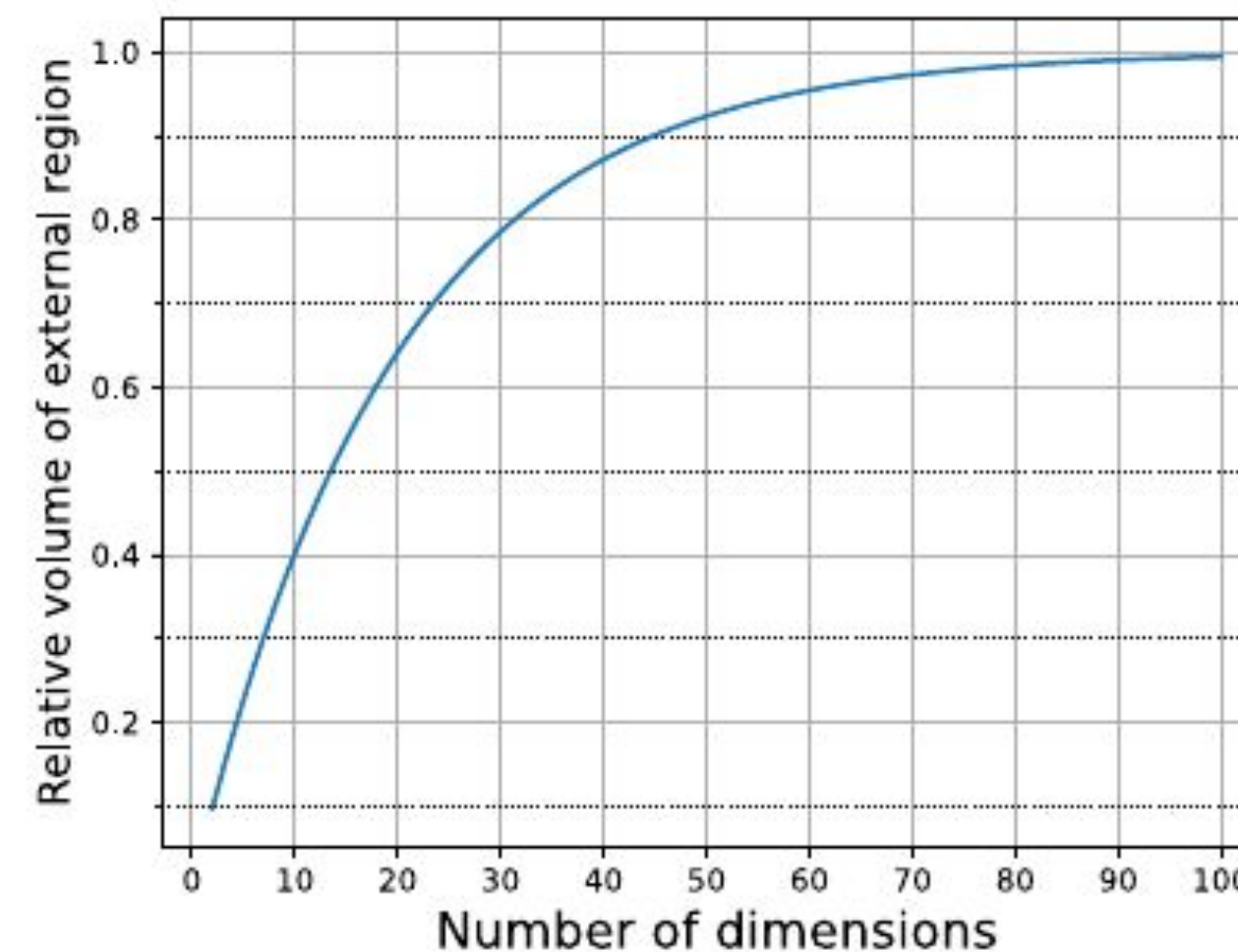


2D

$$\frac{V}{V} = \frac{V - V}{V} = 1 - \frac{V}{V} = 1 - \frac{0,95^2 L^2}{L^2} = 1 - 0,95^2 = 0,0975$$

3D

$$\frac{V}{V} = \frac{V - V}{V} = 1 - \frac{V}{V} = 1 - \frac{0,95^3 L^3}{L^3} = 1 - 0,95^3 = 0,143$$



Rodrigo F. Díaz, Aprendizaje Automático (2021)

# Reducción de la dimensionalidad

---

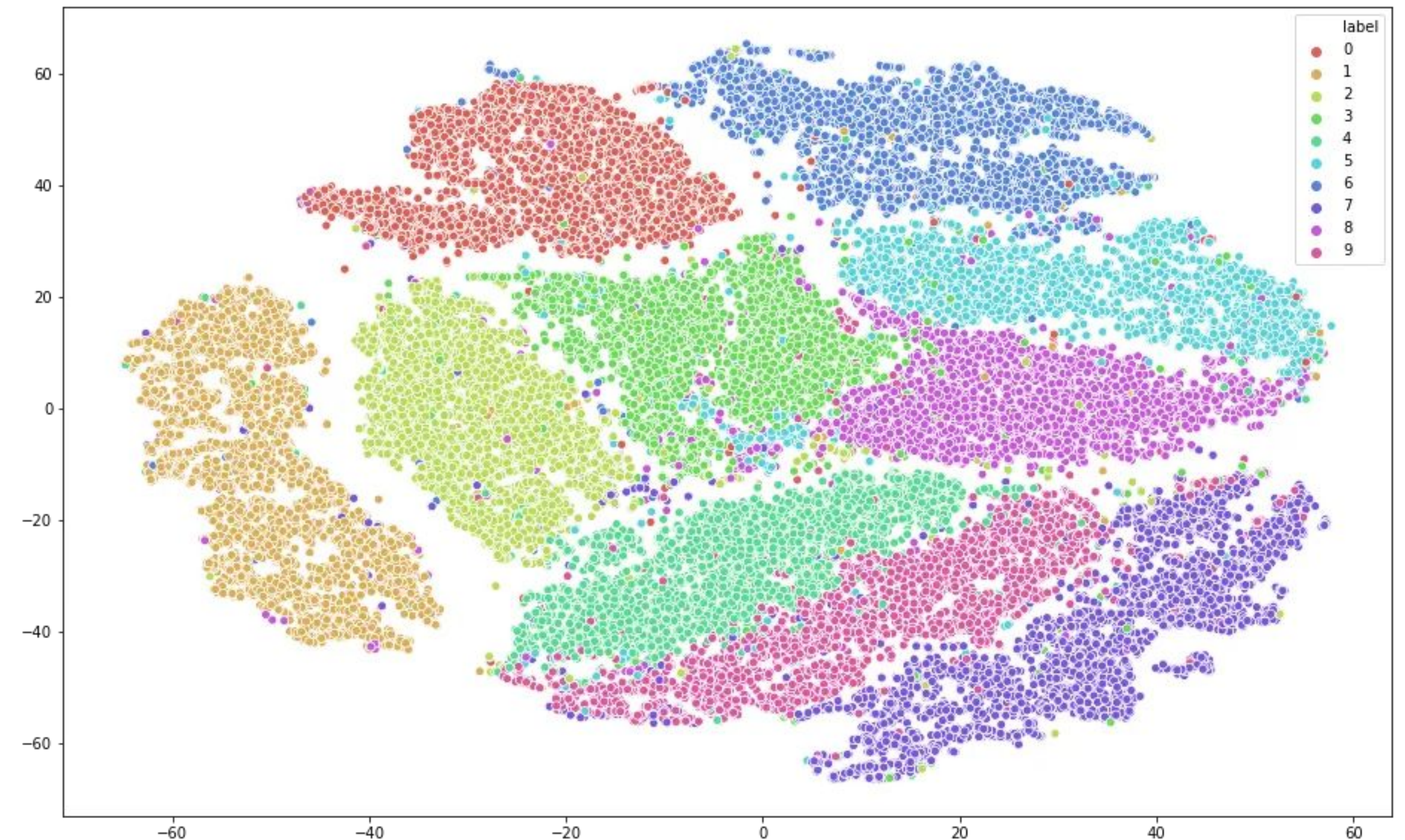
- La distancia media entre puntos aumenta, los datos tienden a ser muy dispersos.
- Una nueva instancia va a estar muy lejos de cualquier instancia de entrenamiento, las predicciones van a ser menos confiables porque van a estar basadas en extrapolaciones muy grandes.
- Mayor riesgo de *overfitting*.
- Entrenamientos más costosos, necesidad de una cantidad **enorme** (\*) de datos con sus problemas de obtención y manipulación (o directamente, imposibilidad práctica).

(\*) Con 100 características, todas entre 0 y 1, se necesitarían más instancias de entrenamiento que átomos en el universo observable para que cada punto de entrenamiento esté dentro de 0,1 de media, suponiendo que estuvieran repartidas uniformemente en todas las dimensiones. (Gerón, Capítulo 8)



# Reducción de la dimensionalidad

- Las ventajas de reducir las dimensiones de los datos son claras:
  - Alejarnos de la maldición de la dimensionalidad.
  - Mejora el rendimiento (tiempo de cómputo, memoria).
  - Mejora la calidad de los datos eliminando el ruido de dimensiones irrelevantes o redundantes.
  - Herramienta contra el *overfitting*.
  - Facilita la visualización, podemos representar datos complejos en dos o tres dimensiones para entender patrones.



Scatter plot 2D de los datos MNIST tras aplicar PCA (50 componentes) y luego t-SNE ([Link](#))



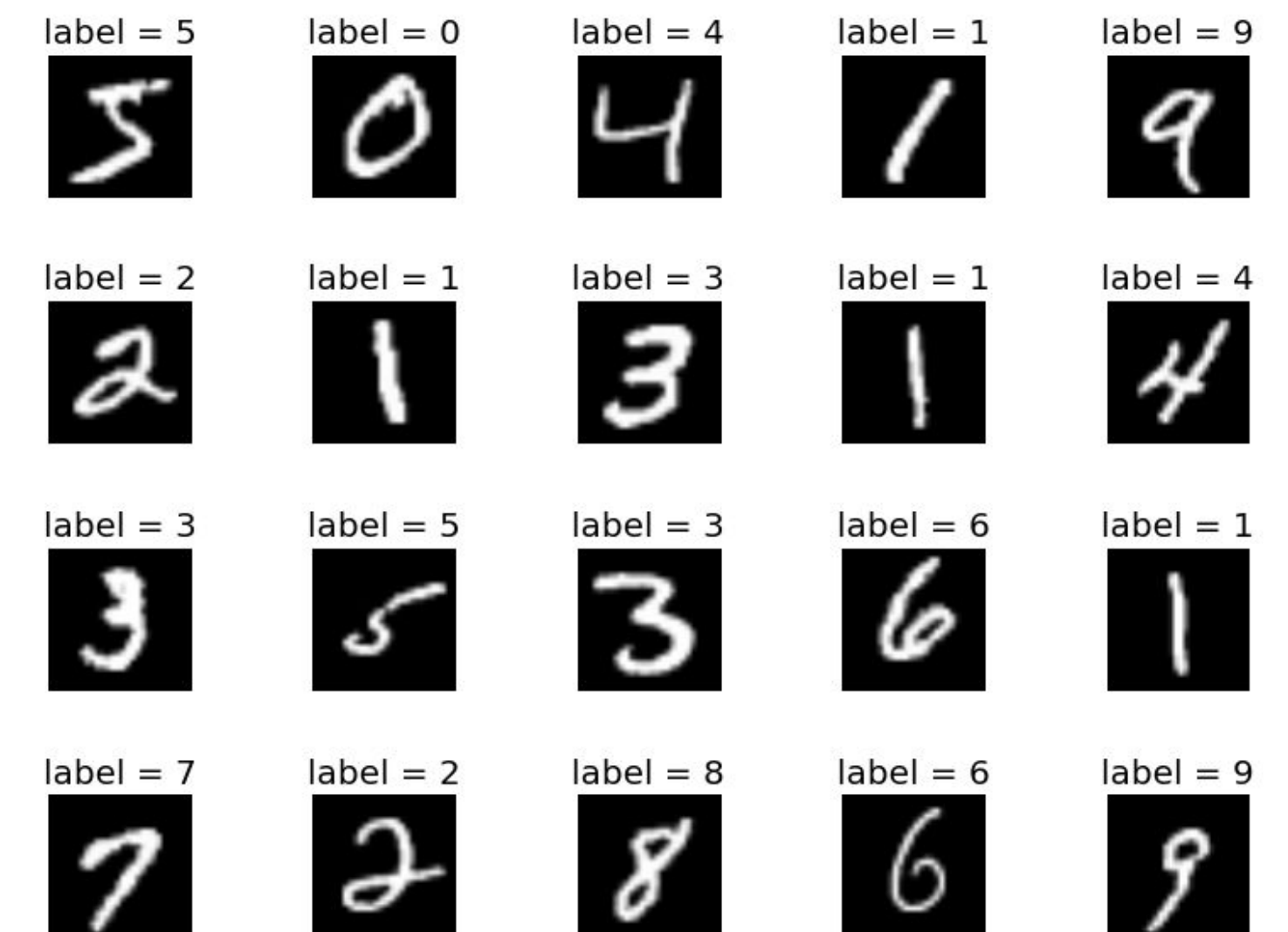
# Reducción de la dimensionalidad

---

- En la mayoría de los problemas del mundo real, las instancias de entrenamiento no están repartidas uniformemente por todas las dimensiones.
- Muchas características son casi constantes, mientras que otras están muy correlacionadas.
- Se puede pensar, entonces, que todas las instancias de entrenamiento se encuentran dentro (o cerca) de un subespacio de dimensiones mucho más bajas que las del espacio original.
- Pensemos el caso de los datos de MNIST.

# Reducción de la dimensionalidad

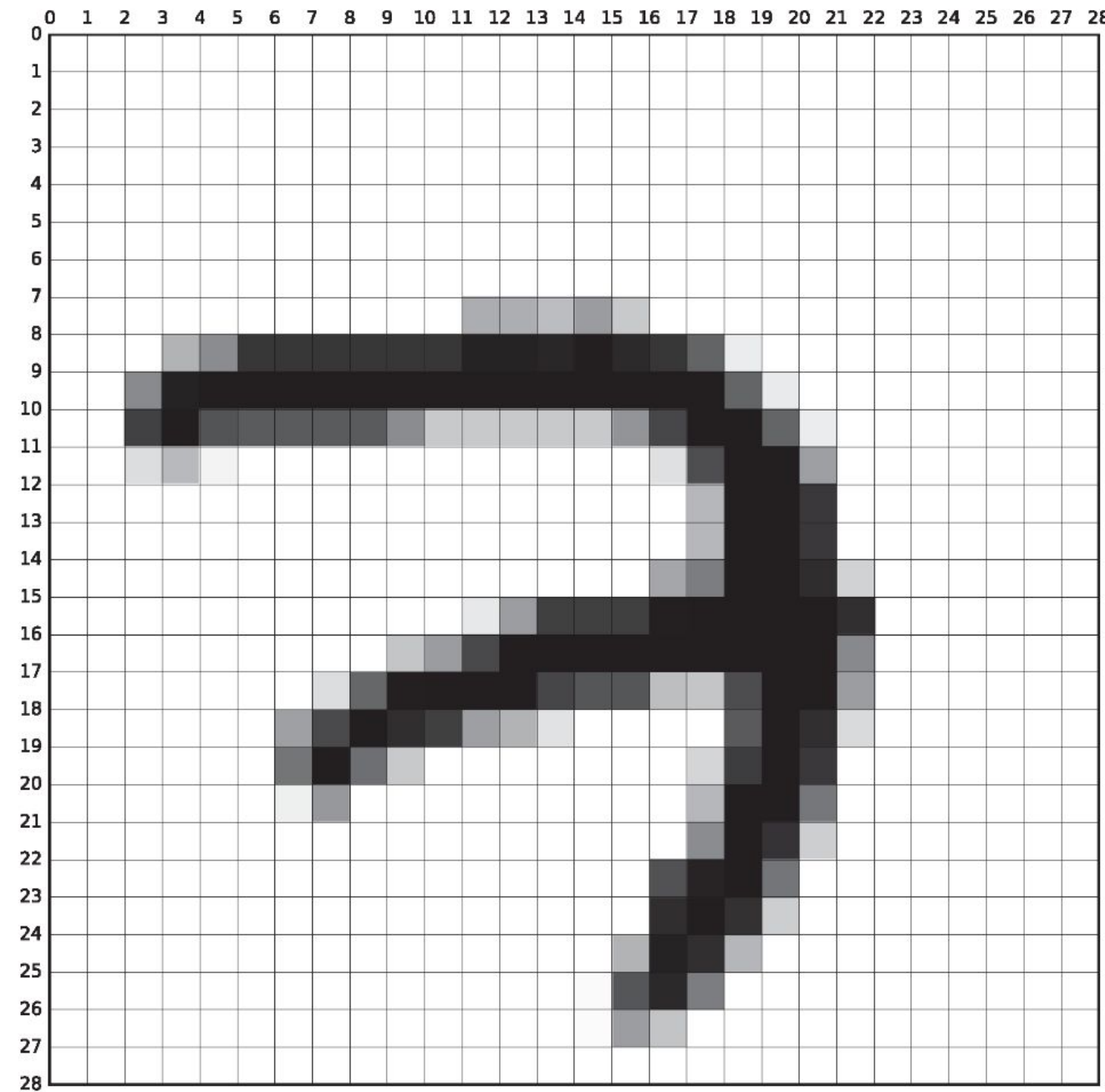
- MNIST (Modified National Institute of Standards and Technology) es un conjunto clásico de datos usado para entrenar y evaluar modelos de aprendizaje automático, especialmente en reconocimiento de imágenes.
- Contiene 70000 imágenes de dígitos escritos a mano (0–9) con su correspondiente etiqueta.
- Cada imagen tiene  $28 \times 28$  píxeles en escala de grises, o sea cada píxel puede tomar uno de 256 valores posibles





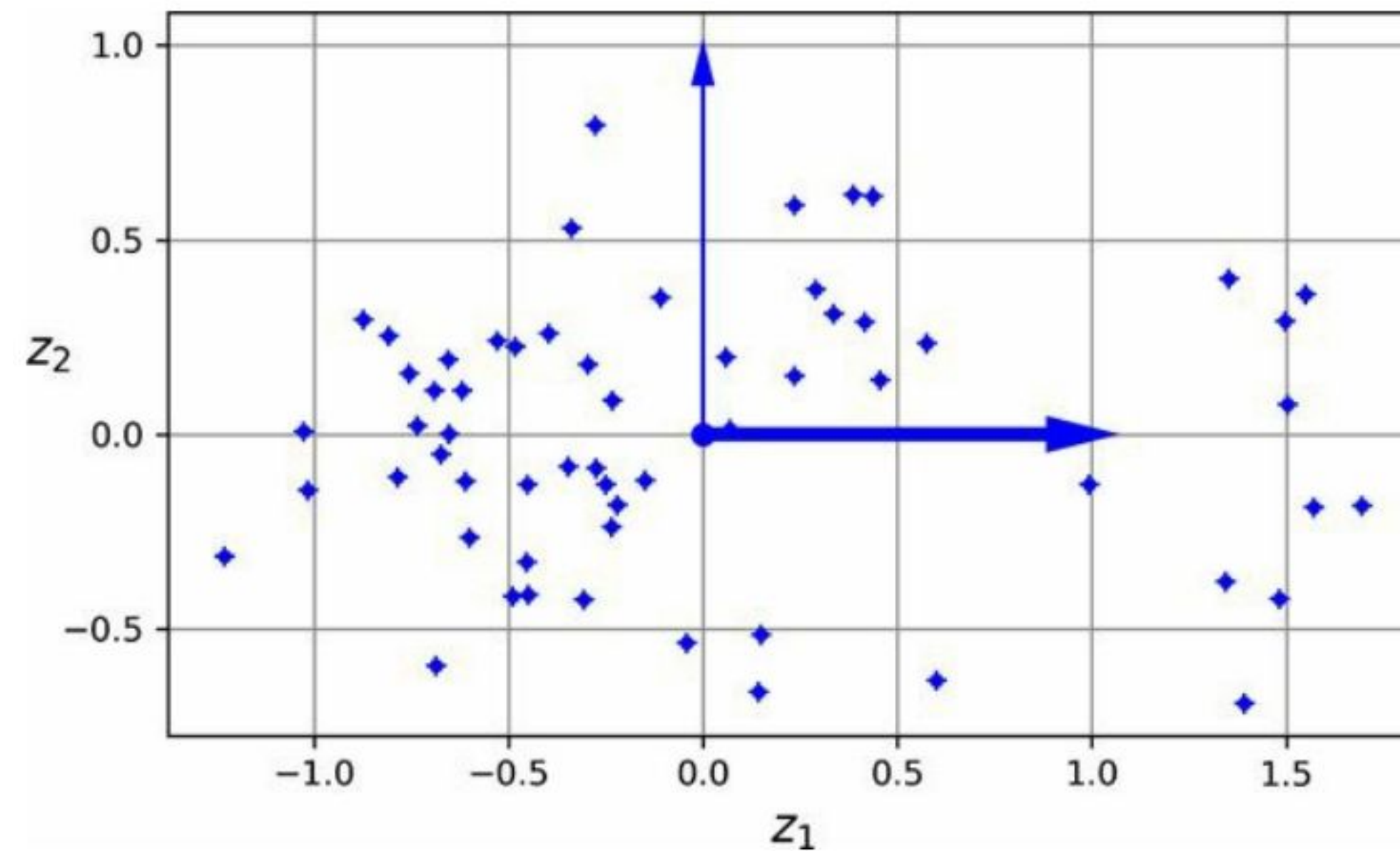
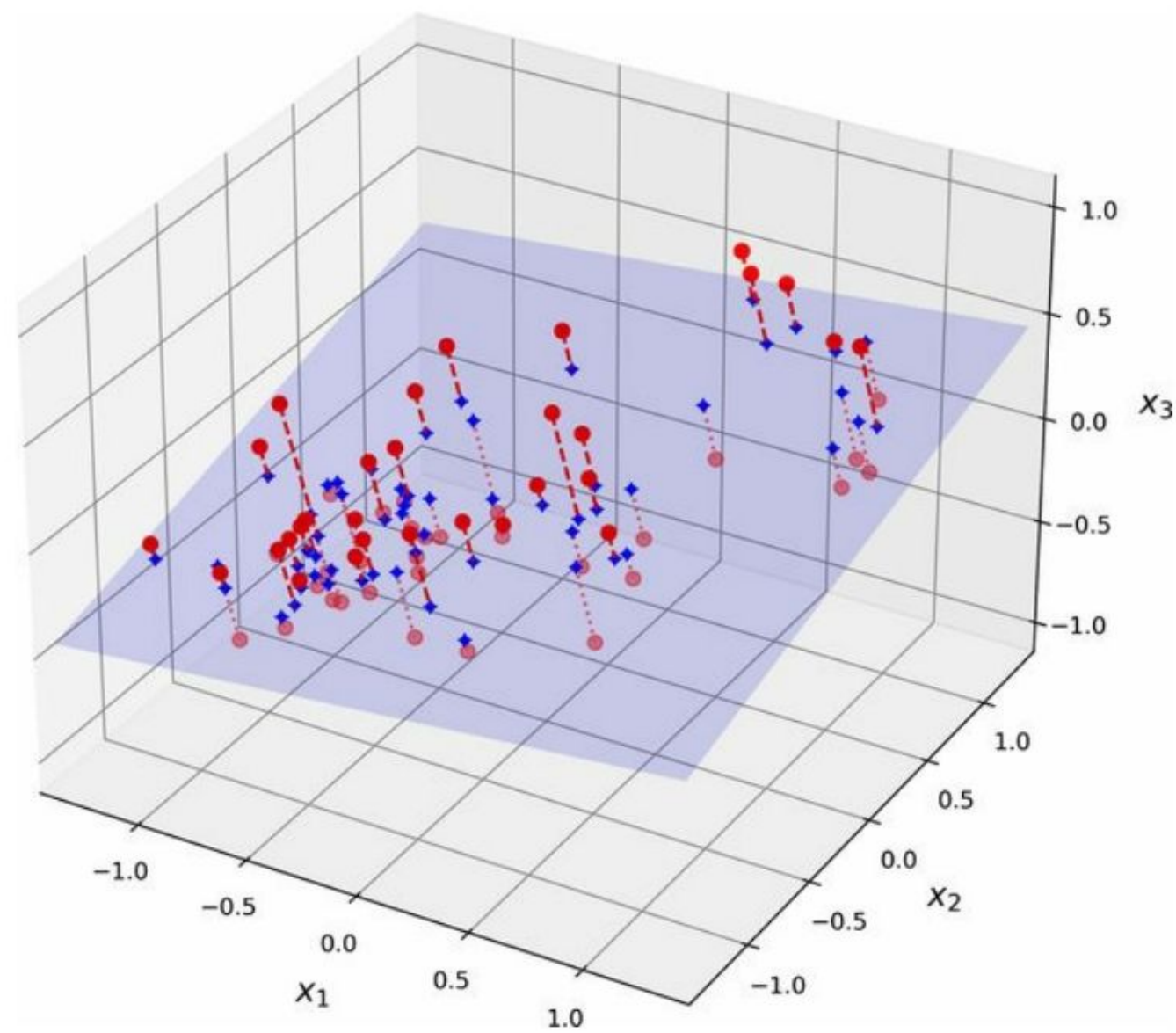
# Reducción de la dimensionalidad

- Cada imagen de MNIST “vive” en un espacio de 784 dimensiones.
- ¿Le parece que todas estas dimensiones son igual de importantes al transmitir información?



# Reducción de la dimensionalidad

- Pensemos en proyecciones, podemos simplemente proyectar los datos en hiperplanos de dimensiones inferiores.

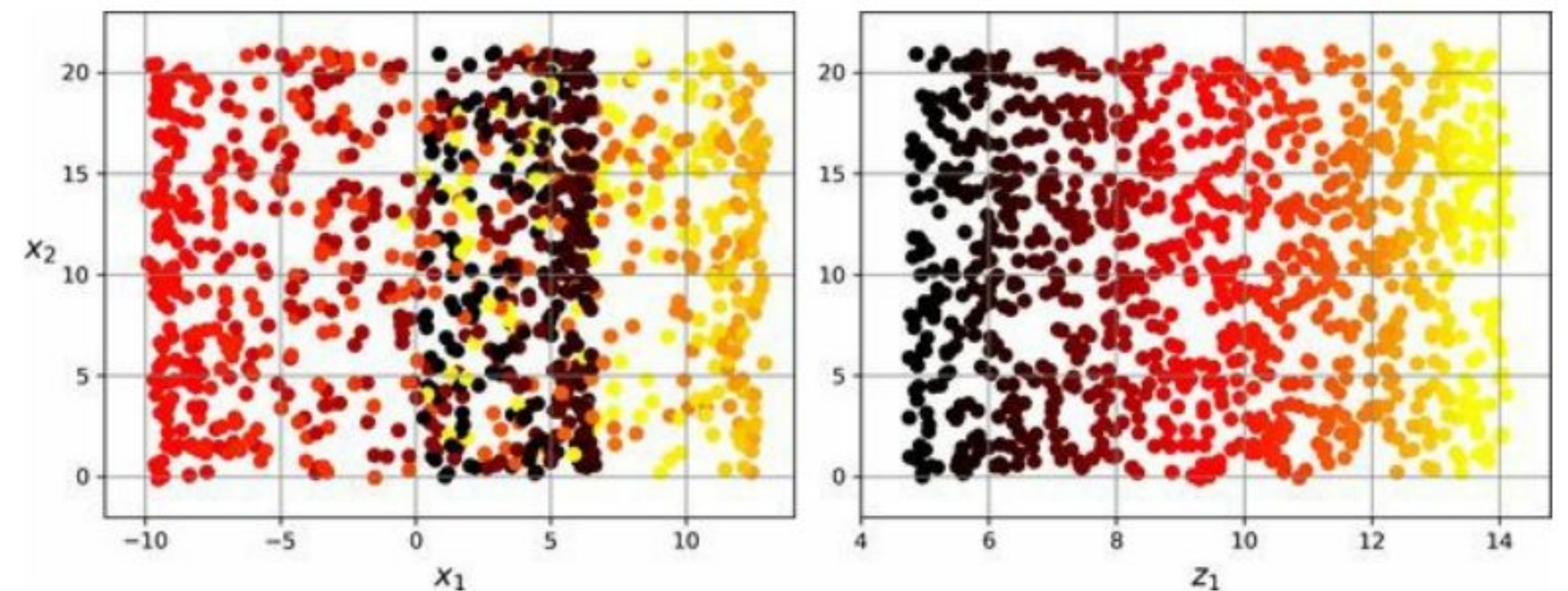
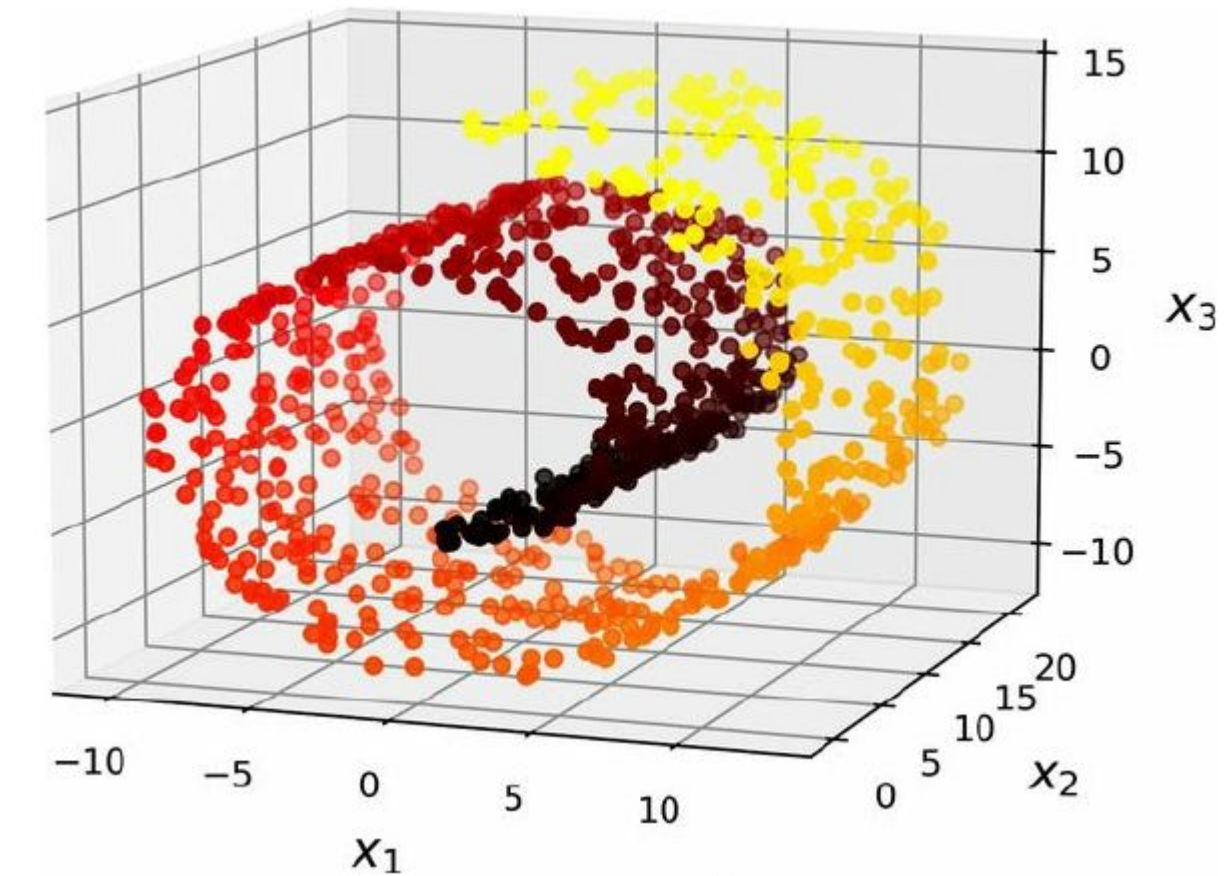


Imágen del Gerón



# Reducción de la dimensionalidad

- Otra opción más avanzada puede ser lo que se conoce como *Manifold Learning*.
- Esta técnica busca reducir la dimensionalidad de los datos preservando sus propiedades geométricas y estructurales más relevantes.
- Parte de la idea de que, aunque los datos estén en un espacio de alta dimensión, en realidad se encuentran sobre una “variedad” (*manifold*) de menor dimensión que describe su estructura esencial.
- En el ejemplo del *Swiss roll*, se puede apreciar la diferencia entre proyectar los datos y “desenrollar” la forma subyacente del *manifold*.



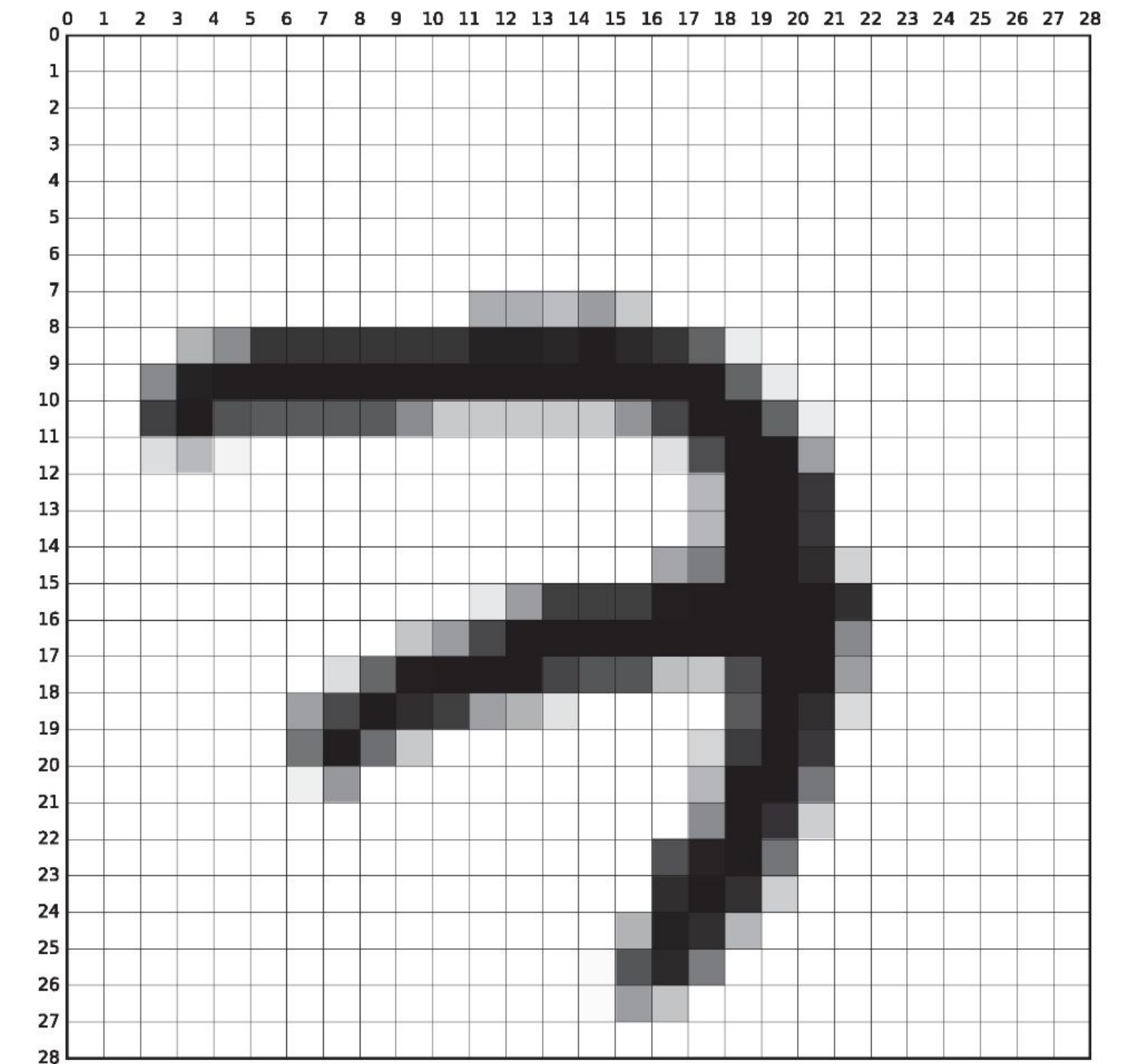
Aplastando con proyecciones (izq.) Vs. Unrolling

Imagen del Gerón



# Reducción de la dimensionalidad

- Pensemos en MNIST: las imágenes de dígitos manuscritos comparten muchas similitudes como líneas conectadas, bordes blancos y números centrados.
- Si generáramos imágenes al azar, sólo una fracción ínfima se parecería a un dígito real.
- En otras palabras, las imágenes posibles de dígitos ocupan una región muy pequeña del espacio total, lo que equivale a decir que están concentradas en una variedad de menor dimensión.



# Preservando la varianza al proyectar

- Antes de poder proyectar el conjunto de entrenamiento en un hiperplano de dimensiones inferiores, primero hay que elegir el adecuado.
- Las diferentes proyecciones conservan diferentes porcentajes de la varianza (en el ejemplo: máxima, media y baja)
- Parece razonable seleccionar el eje que conserva la máxima cantidad de varianza, ya que lo más probable es que pierda menos información que las demás proyecciones.
- Otra forma de justificar esta elección es que **se trata del eje que minimiza la distancia cuadrática media entre el conjunto de datos original y su proyección sobre ese eje**. Esta es la sencilla idea que subyace a PCA.

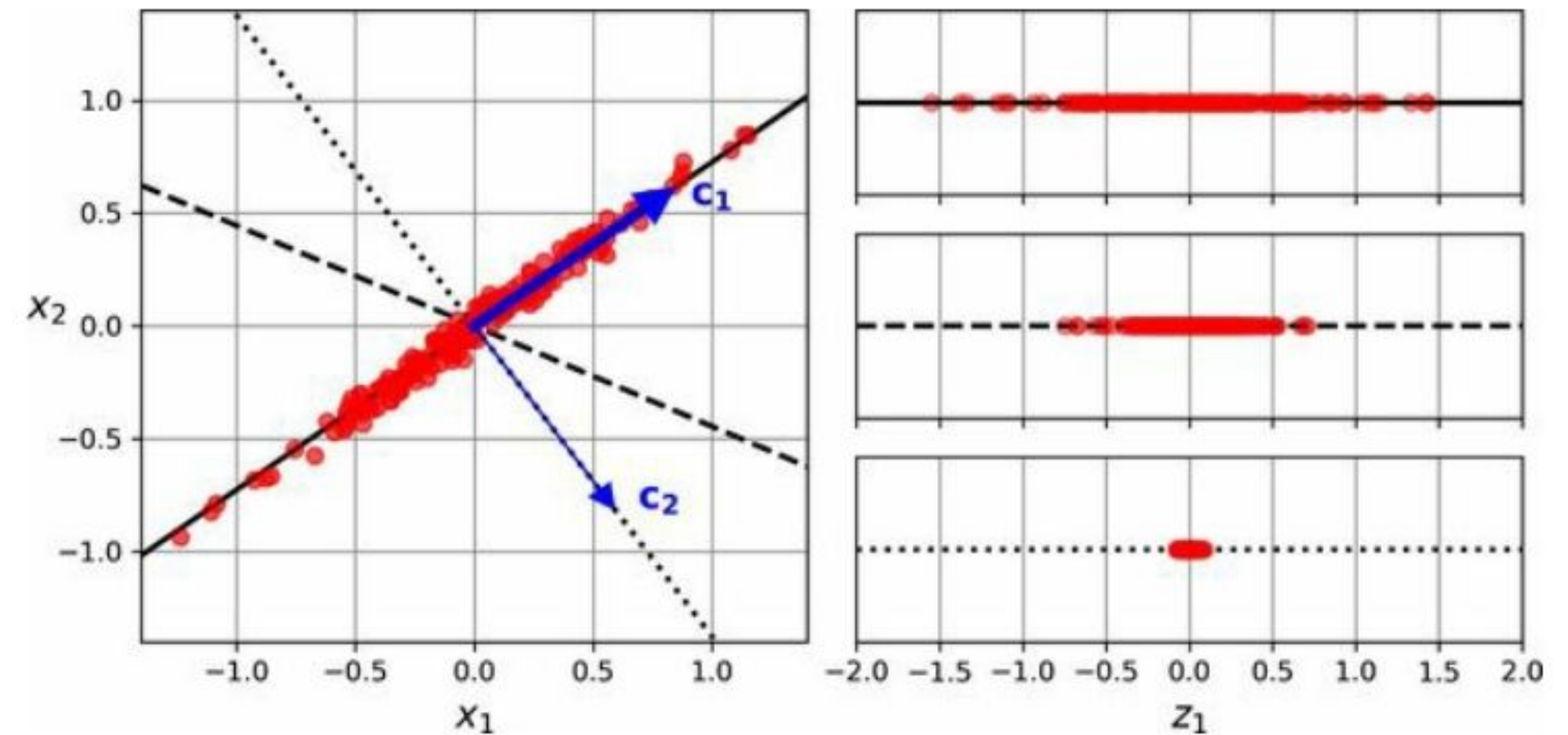
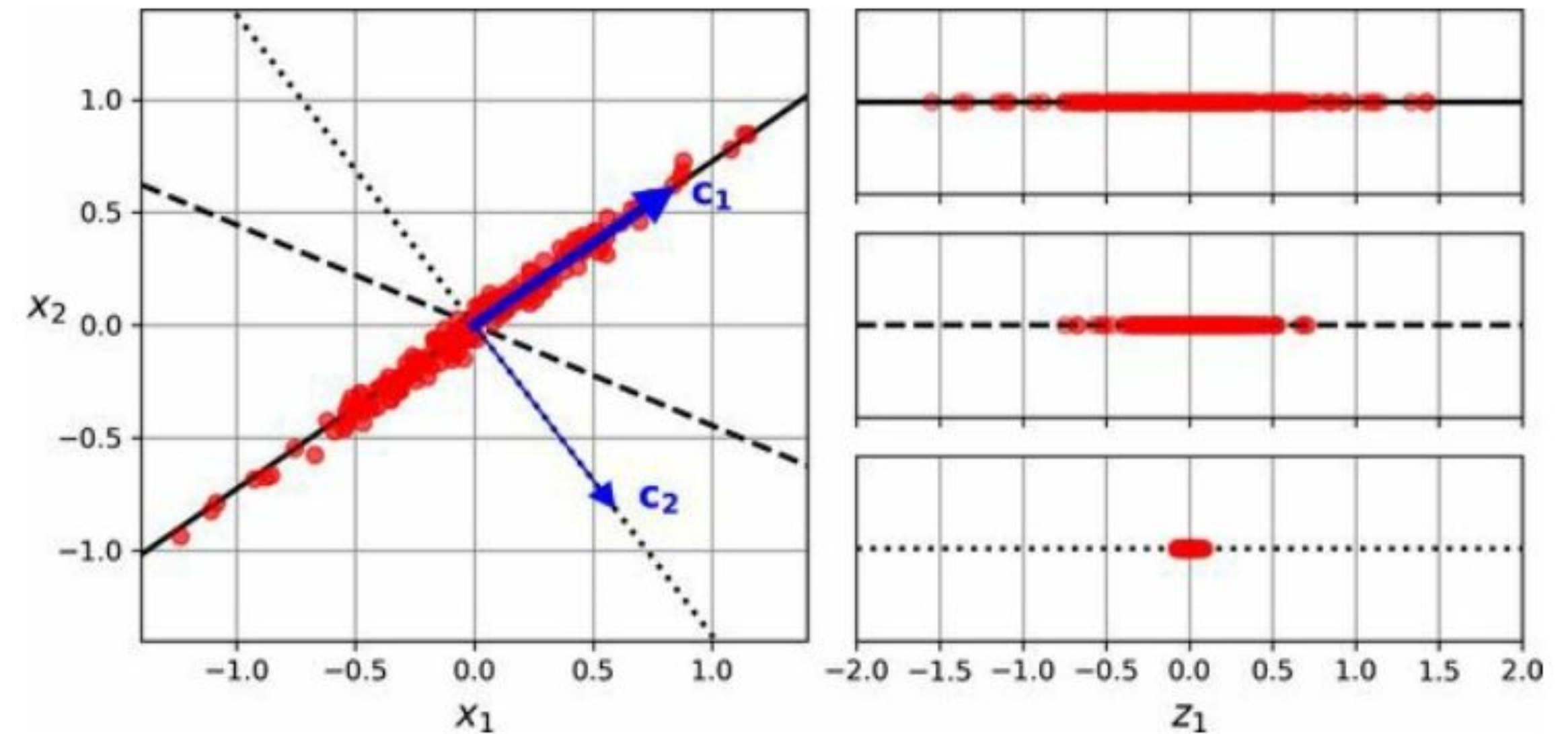


Imagen del Gerón



# Análisis de Componentes Principales (PCA)

- PCA identifica el eje que representa la mayor cantidad de varianza en el conjunto de entrenamiento (la línea continua).
- También encuentra un segundo eje, **ortogonal al primero**, que representa la mayor parte de la varianza restante (en este ejemplo 2D no hay más opciones que la línea punteada).
- Si hay más dimensiones, PCA también encontrará un tercer eje, ortogonal a los dos ejes anteriores, y un cuarto, un quinto, y así sucesivamente. Tantos ejes como dimensiones tengan los datos.

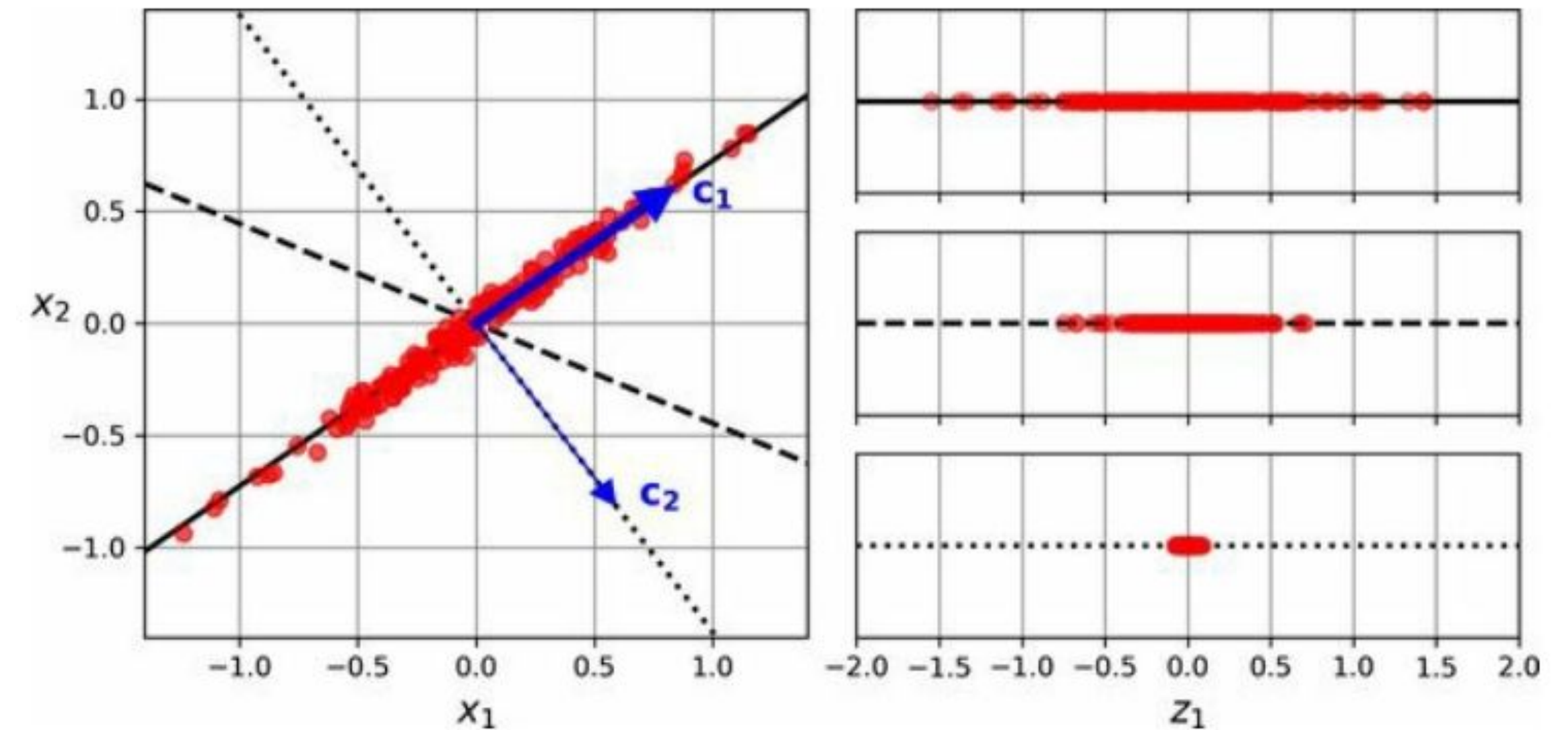


Imágen del Gerón

[LINK: StatQuest: PCA - Practical Tips](#)

# Análisis de Componentes Principales (PCA)

- El  $i$ -ésimo eje se llama el  $i$ -ésimo componente principal (CP) de los datos.
- En el ejemplo de la derecha el primer CP es el eje sobre el que se encuentra  $c_1$ , el segundo es  $c_2$ .
- En la figura de abajo, los dos primeros están en el plano y el tercero al eje ortogonal.



- Luego de la proyección el primer CP corresponde al eje  $Z_1$  y el segundo a  $Z_2$ .

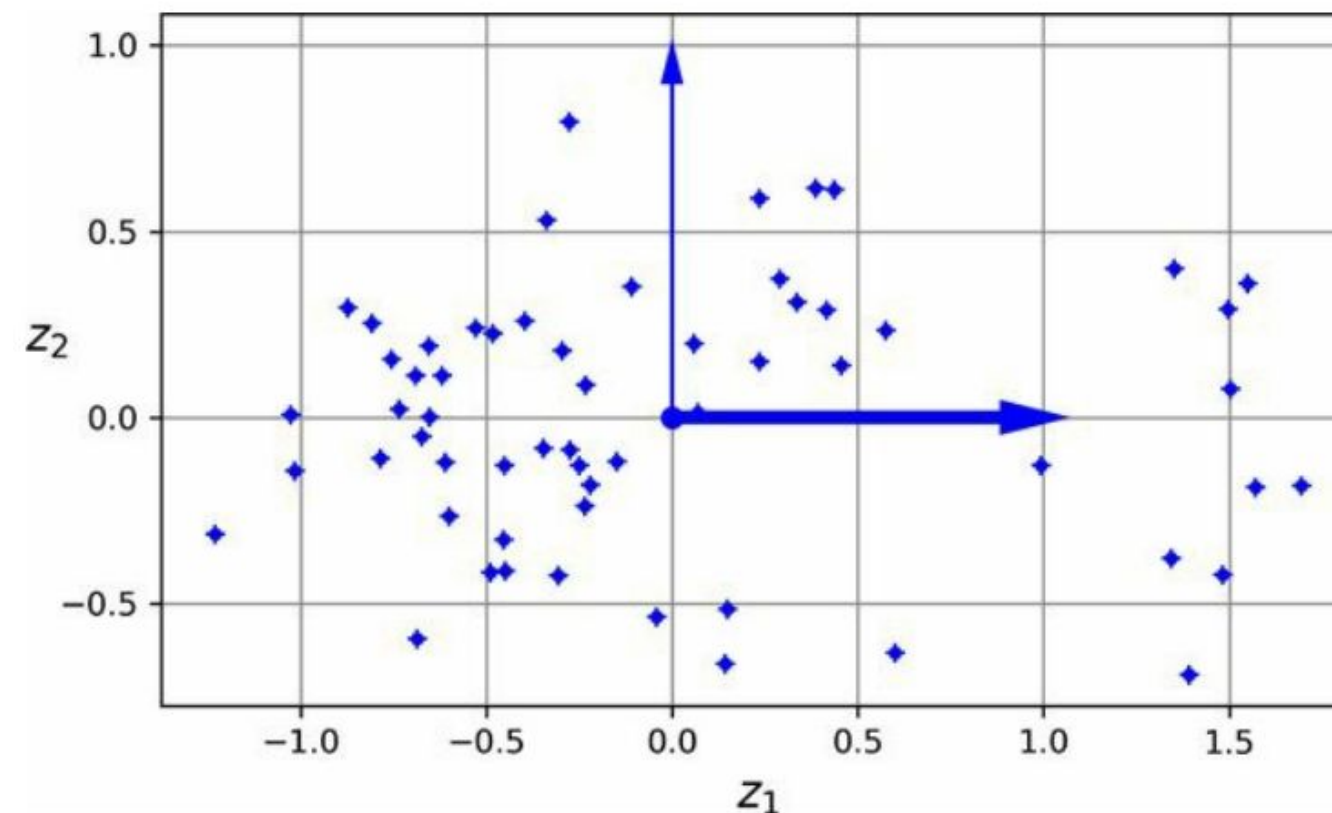
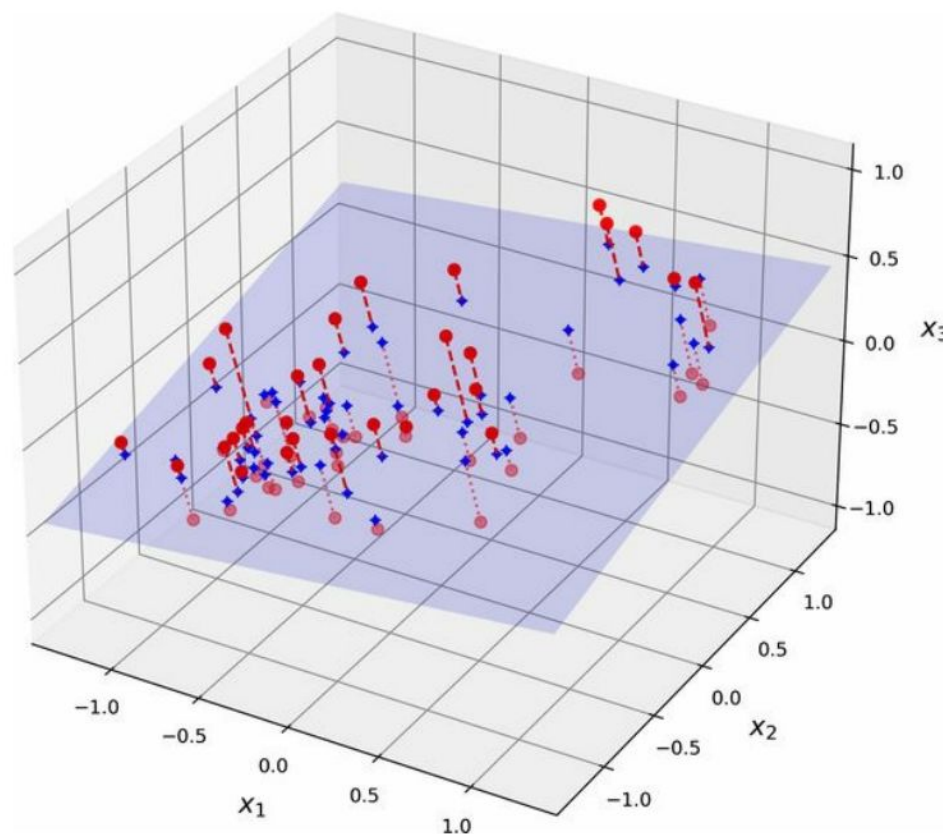
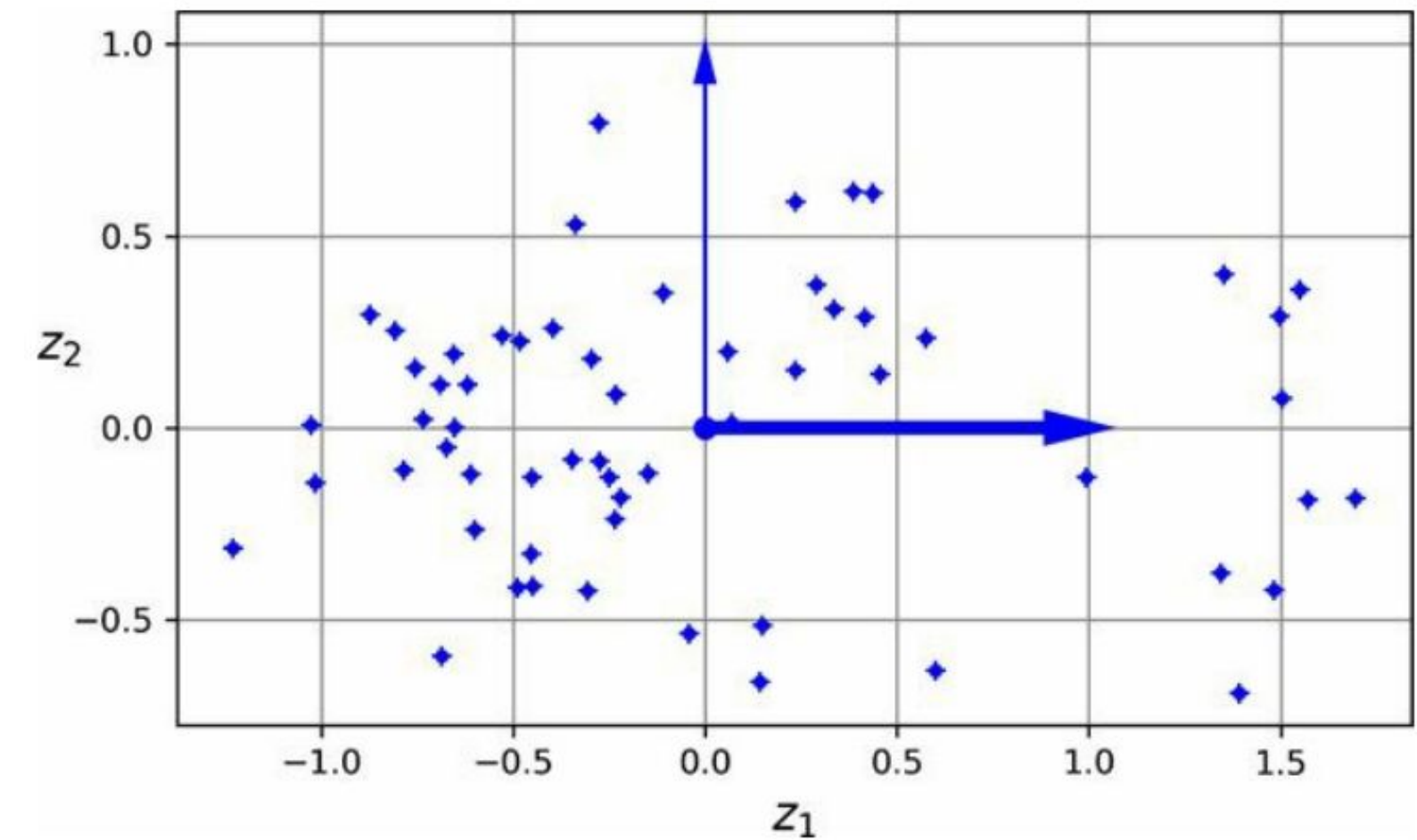


Imagen del Gerón



# Explained Variance Ratio

- Una información útil de este proceso es el *explained variance ratio*.
- Este ratio indica la proporción de la varianza del conjunto de datos que se encuentra en cada componente principal.
- En scikit-learn está accesible a través de la variable `explained_variance_ratio_`.
- En el ejemplo, alrededor del 76% de la varianza de los datos están en la primera CP y un 15% en la segunda, dejando un 9% para la tercera, por lo que es razonable pensar que contiene muy poca información.



```
>>> pca.explained_variance_ratio_  
array([0.7578477, 0.15186921])
```

Imagen del Gerón

# Elegir la cantidad de dimensiones

- Si queremos visualizar datos es sencillo, vamos a elegir dos o tres CP.
- Si, en cambio, estamos buscando reducir para poder seguir trabajando con los datos podemos buscar que número de componentes explican una porción suficientemente grande de la varianza (por ejemplo un 95%).
- Otra opción es plotear la varianza explicada en función del número de dimensiones, buscando el “codo” en la curva donde la varianza deja de “crecer rápido”.
- Finalmente, también se puede pensar como un hiperparámetro más y probar diferentes combinaciones junto al algoritmo con el que estamos haciendo la clasificación o regresión posterior.

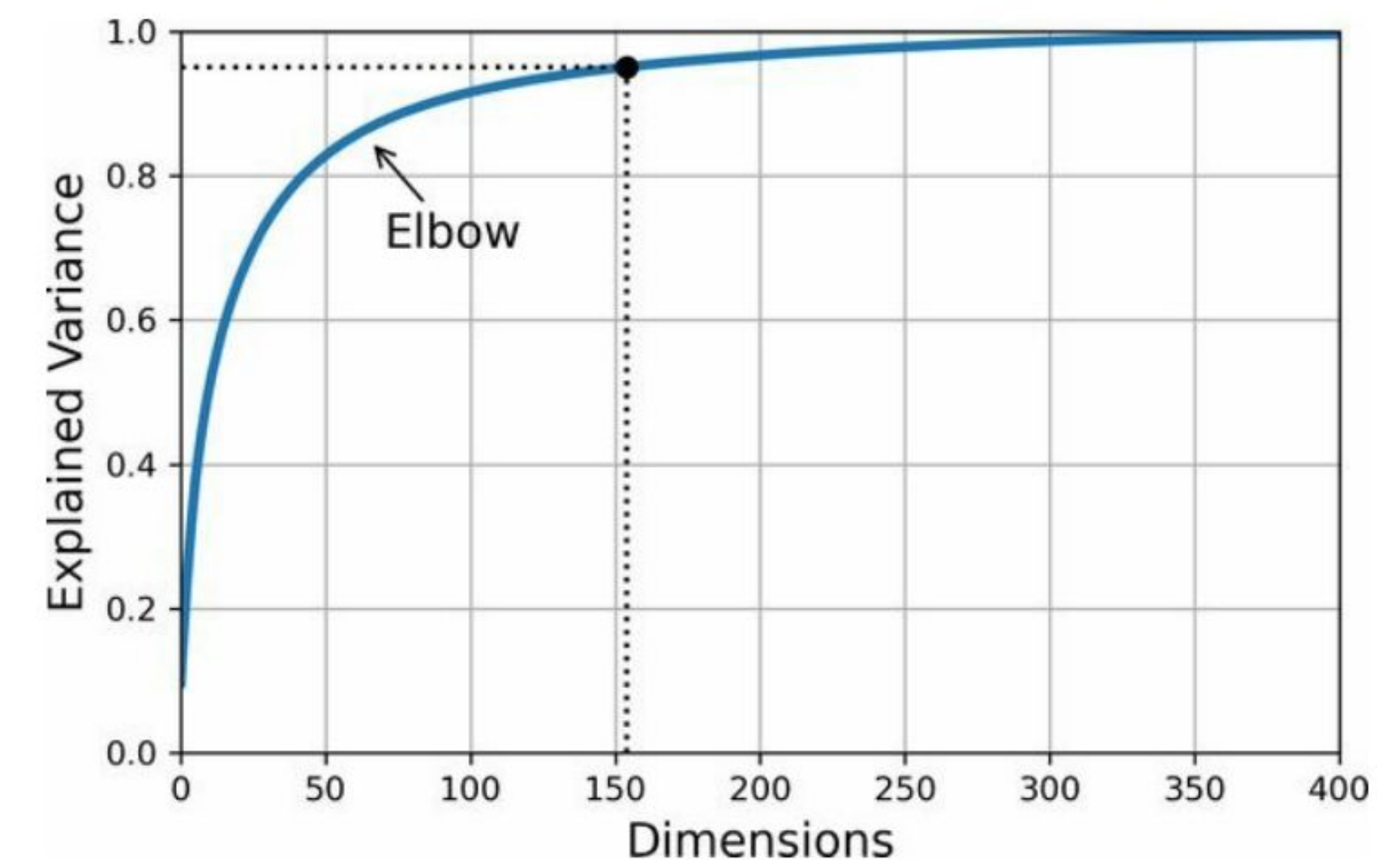


Imagen del Gerón



# PCA para compresión

- Luego de reducir sus dimensiones, el conjunto de entrenamiento ocupa mucho menos.
- Por ejemplo, tras aplicar PCA al conjunto de datos MNIST y conservar el 95% de su varianza, nos quedamos con 154 características, en lugar de las 784 originales.
- Ahora tenemos un 20% del tamaño original habiendo perdido solamente un 5% de su varianza.
- Es posible hacer la transformación inversa al espacio original, obviamente no nos va a dar los datos originales, pero se va a parecer bastante.
- La distancia media al cuadrado entre los datos originales y los datos reconstruidos se denomina **error de reconstrucción**.

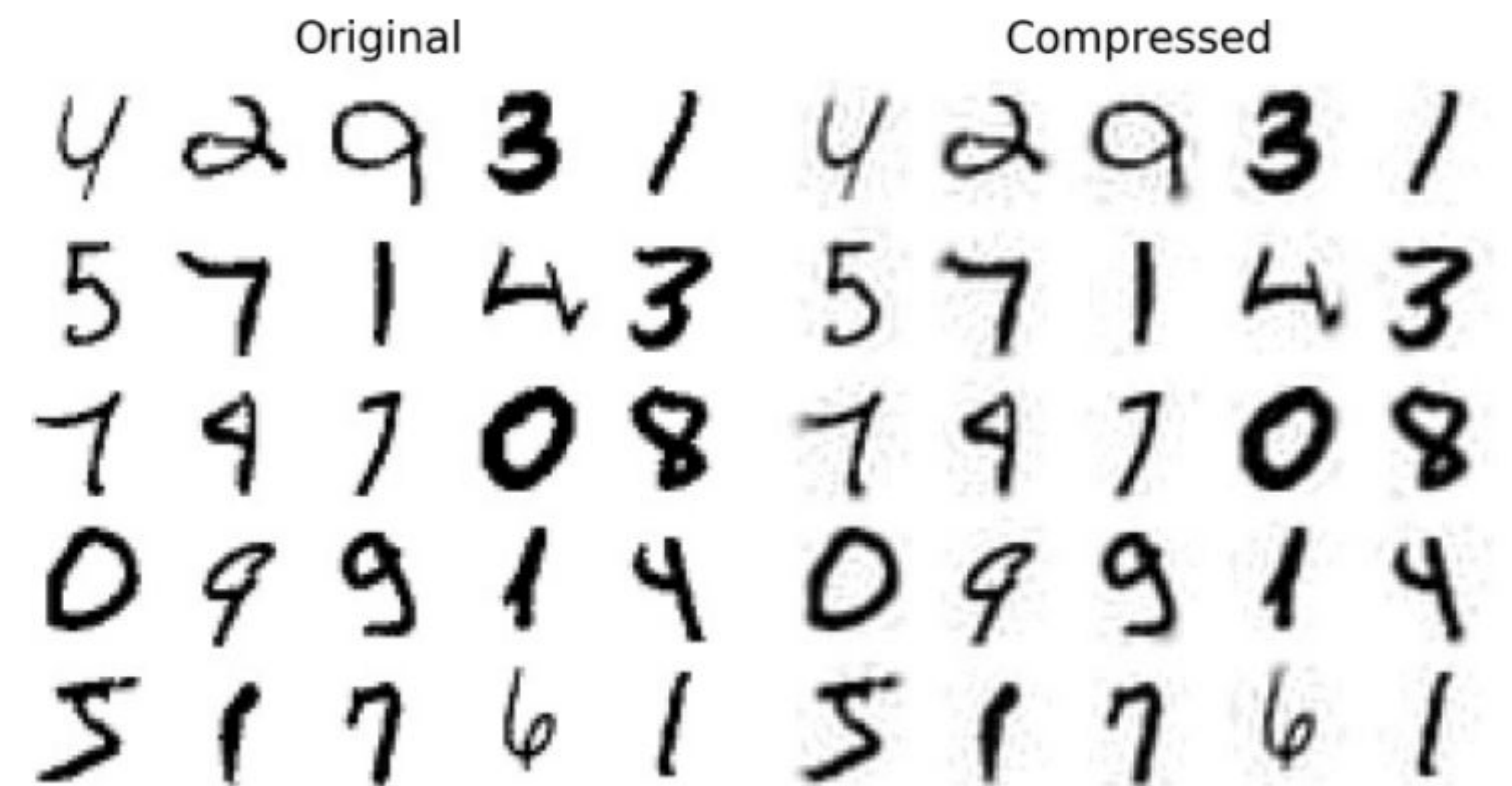


Imagen del Gerón



# Clustering

---

- *Clustering* (agrupamiento) es una técnica de aprendizaje no supervisado que busca organizar los datos en grupos (*clusters*), de modo que los elementos dentro de un mismo grupo sean más parecidos entre sí que con los de otros grupos.
- Usos principales:
  - Se puede usar para la detección de anomalías, para clasificar, para hacer un análisis exploratorio de los datos, etc.
  - Se suele usar en conjunción con las técnicas de reducción de la dimensionalidad para optimizar tiempos o resultados.
- Pensemos cómo es que se puede hacer esto de manera no supervisada.



# Clustering

---



- Puede que no sepamos **qué** tipo de árbol es cada uno, pero podemos saber **cuáles** son del mismo tipo (más o menos).



# Clustering

---



- Puede que no sepamos **qué** tipo de árbol es cada uno, pero podemos saber **cuáles** son del mismo tipo (más o menos).
- Depende de las **características** que elijamos ver.



# Clustering

---



- Puede que no sepamos **qué** tipo de árbol es cada uno, pero podemos saber **cuáles** son del mismo tipo (más o menos).
- Depende de las **características** que elijamos ver.
- ¿Color? ¿forma de la copa? ¿diámetro?, ¿altura? ¿una combinación de ambos?



# Clustering

---



- Puede que no sepamos **qué** tipo de árbol es cada uno, pero podemos saber **cuáles** son del mismo tipo (más o menos).
- Depende de las **características** que elijamos ver.
- ¿Color? ¿forma de la copa? ¿diámetro?, ¿altura? ¿una combinación de ambos?
- Si elegimos pocos, no podemos agrupar bien; si elegimos muchos, perdemos generalidad. **Es un balance.**



# Clustering



- Cada árbol puede representarse con un vector de características
- (altura, diámetro, tono de verde, ... ).

(25, 120, 3A5311, ...)

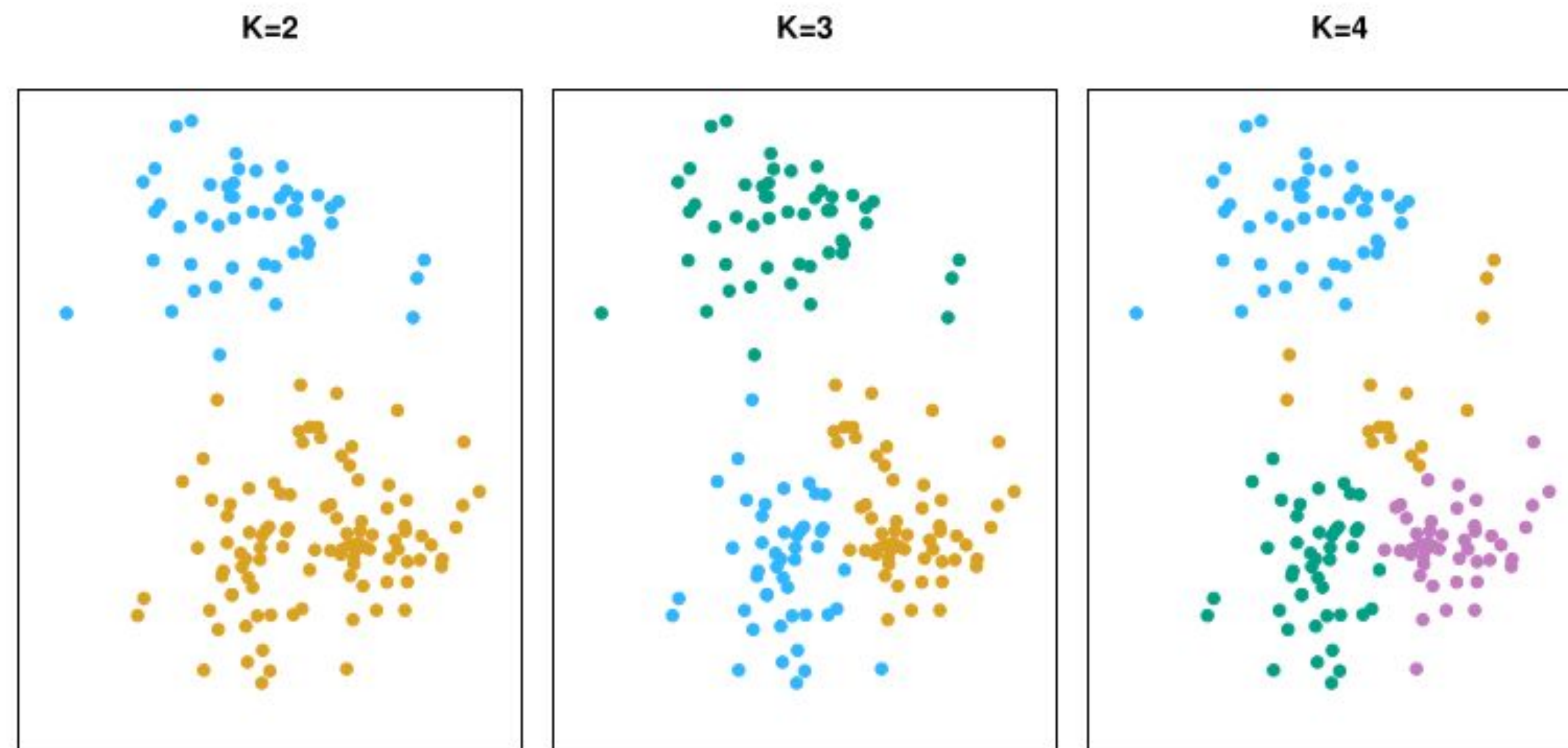
(10, 30, 597D35, ...)

- Luego podemos ver que los árboles más similares son los que están más cerca en este espacio de características.



# K-Means

- K-Means es uno de los métodos más populares, si bien no es de los más poderosos, pero su rapidez y simplicidad lo convierte en una excelente opción como punto de partida para explorar problemas de *clustering*.
- Simplemente se le dan los datos y se le indica un número  $k$  que representa la cantidad de *clusters* a buscar y el algoritmo le asigna un grupo a cada punto de nuestro dataset.

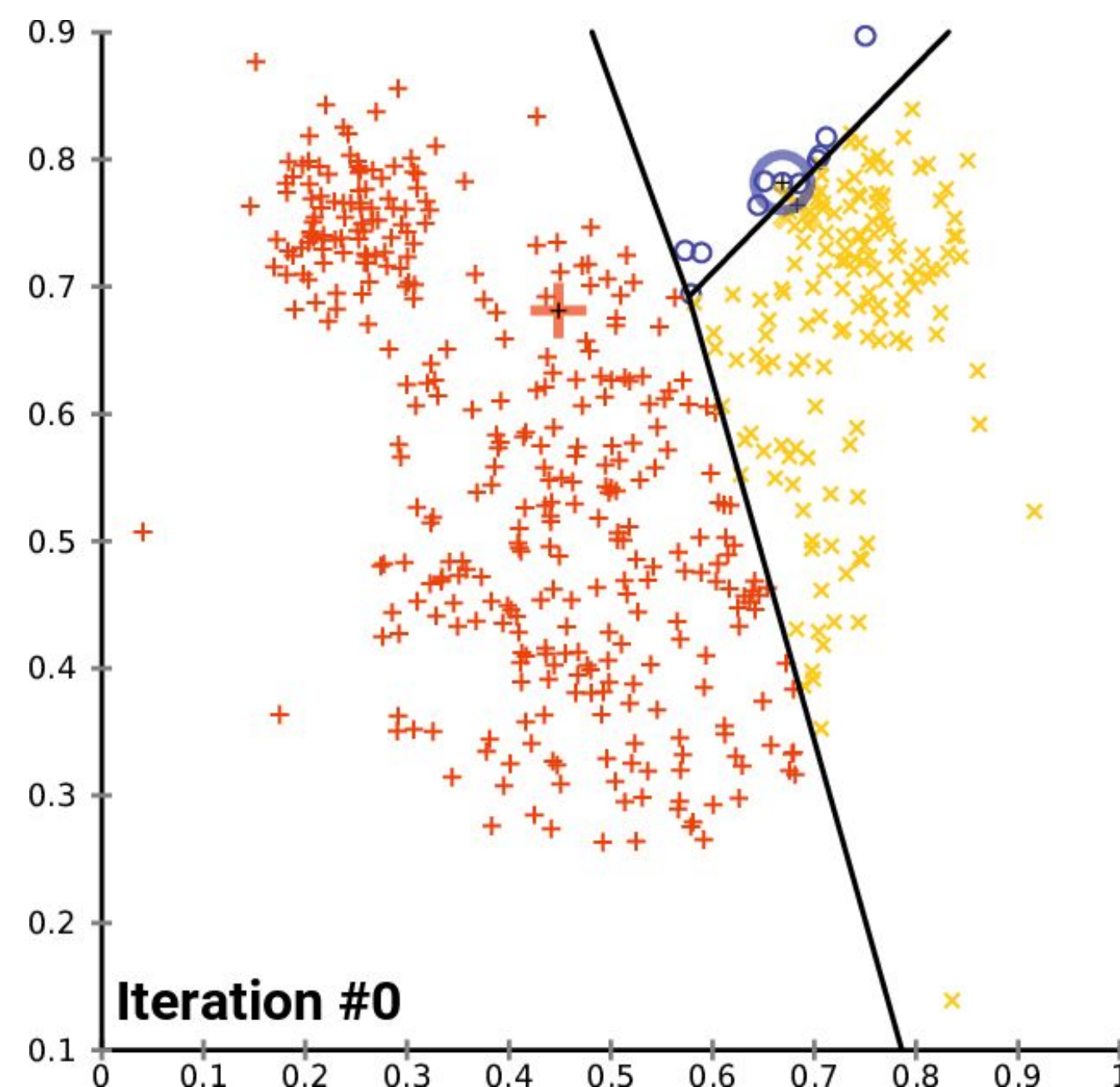


Para pensar:  
¿Qué es un cluster?

Imagen de [An Introduction to Statistical Learning](#)

# K-Means

- Encuentra los  $k$  **centroides** que minimizan la distancia entre cada punto del conjunto de datos y su centroide correspondiente.



1. Selecciona  $k$  puntos del conjunto de datos como centroides iniciales.
2. Asigna cada punto del conjunto de datos al centroide más cercano.
3. Recalcula los centroides de cada grupo utilizando la distancia media de todos los puntos que están asociados a ese centroide.
4. Repite los pasos 2 y 3 hasta que no haya cambios o se alcance un número máximo de iteraciones.

Source: Wikipedia

[LINK: StatQuest: K-means clustering](#)



# K-Means

---

- Si se conocen aproximadamente donde los centroides deberían estar (por ejemplo, se usó algún otro algoritmo de clustering antes) se puede inicializar el algoritmo con esos puntos como centroides iniciales.
- Dependiendo de los datos, los clusters encontrados pueden depender de las inicializaciones al azar de los centroides, se puede ejecutar varias veces con varias inicializaciones aleatorias (hiperparámetro `n_init`) y devolver el “mejor agrupamiento”.
- Esto nos lleva a pensar cómo elegimos un  $k$  adecuado y que significa un “buen agrupamiento”...

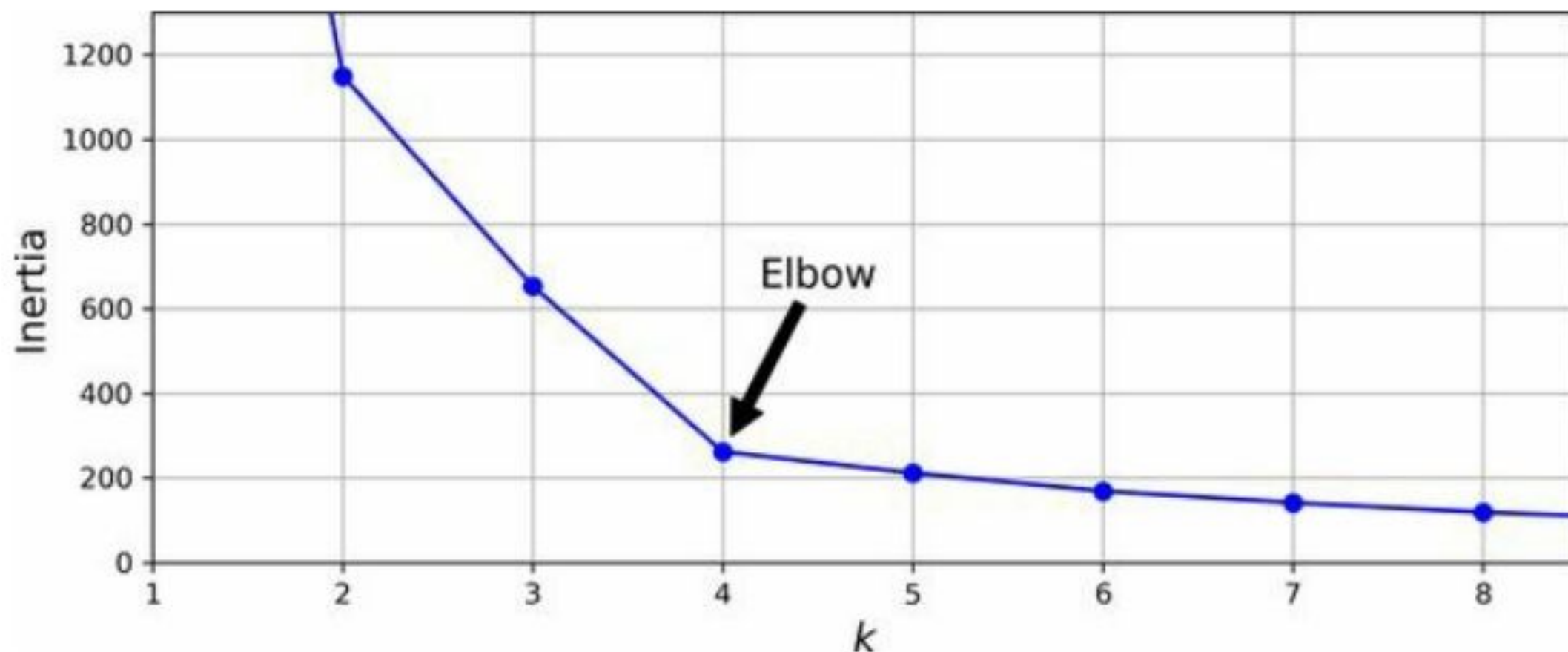
# K-Means

---

- Un criterio para definir el mejor agrupamiento usa como métrica la **inercia**: que es la suma de las distancias cuadradas de cada punto a su centroide más cercano. Cuanto menor es la inercia, más concentrados están los puntos alrededor de los centroides.

$$I = \sum_{j=1}^K \sum_{i=1}^{n_j} (x_i - \mu_j)^2$$

# K-Means



- Se prueba con diferentes  $k$  buscando el punto en el que la inercia *deja de disminuir tan rápido* (formando el famoso “codo”).
- Es una forma rápida de calcular pero un poco tosca (además de que siempre va a disminuir cuando aumenta  $k$ ), otra forma más costosa computacionalmente, pero más precisa, es usar la métrica de **la Silueta**.

Imágen del Gerón



# K-Means

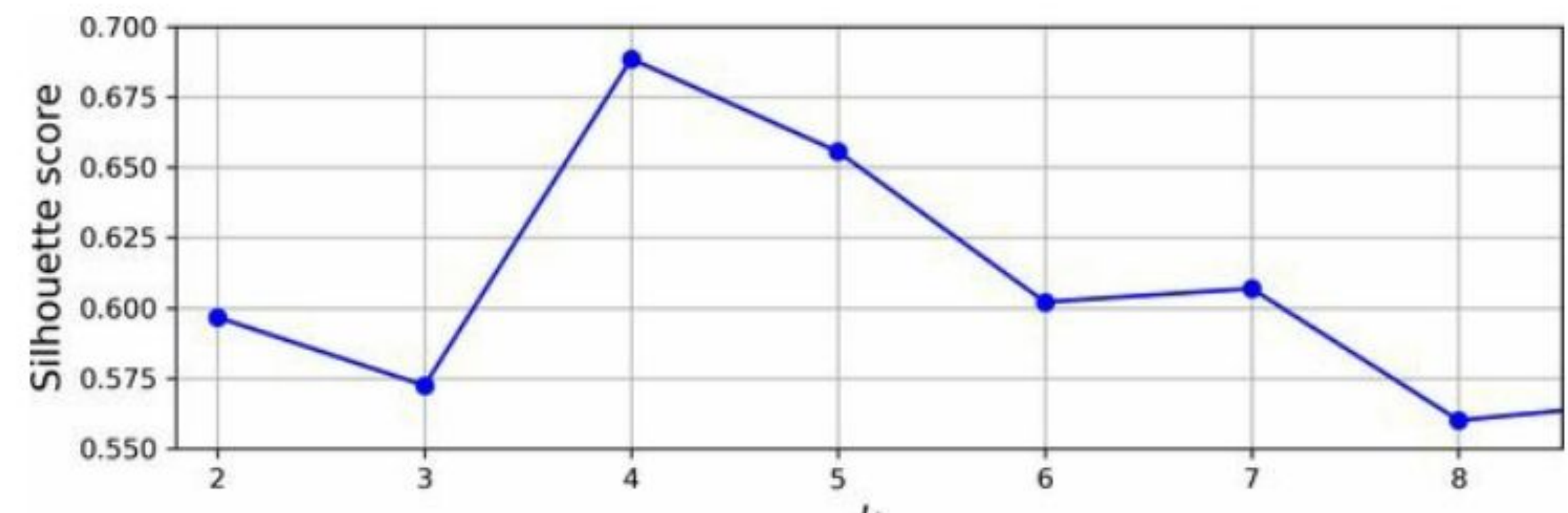
---

- **Silueta**, para cada punto  $i$ :
  - $a(i)$  es la distancia media a otros datos del mismo cluster (es la distancia media intra-cluster).
  - $b(i)$  es la distancia media al cluster más cercano, es decir, la distancia media a datos del siguiente cluster más cercano.

$$S(i) = \frac{(b(i) - a(i))}{\max(a(i), b(i))}$$

# K-Means

- Se calcula para todos los puntos y se hace el promedio.
- Puede variar entre -1 y +1: un coeficiente cercano a +1 significa que el dato está bien asignado dentro de su propio cluster y lejos de otros clusters, mientras que un coeficiente cercano a 0 significa que un dato está cerca del límite de un cluster.

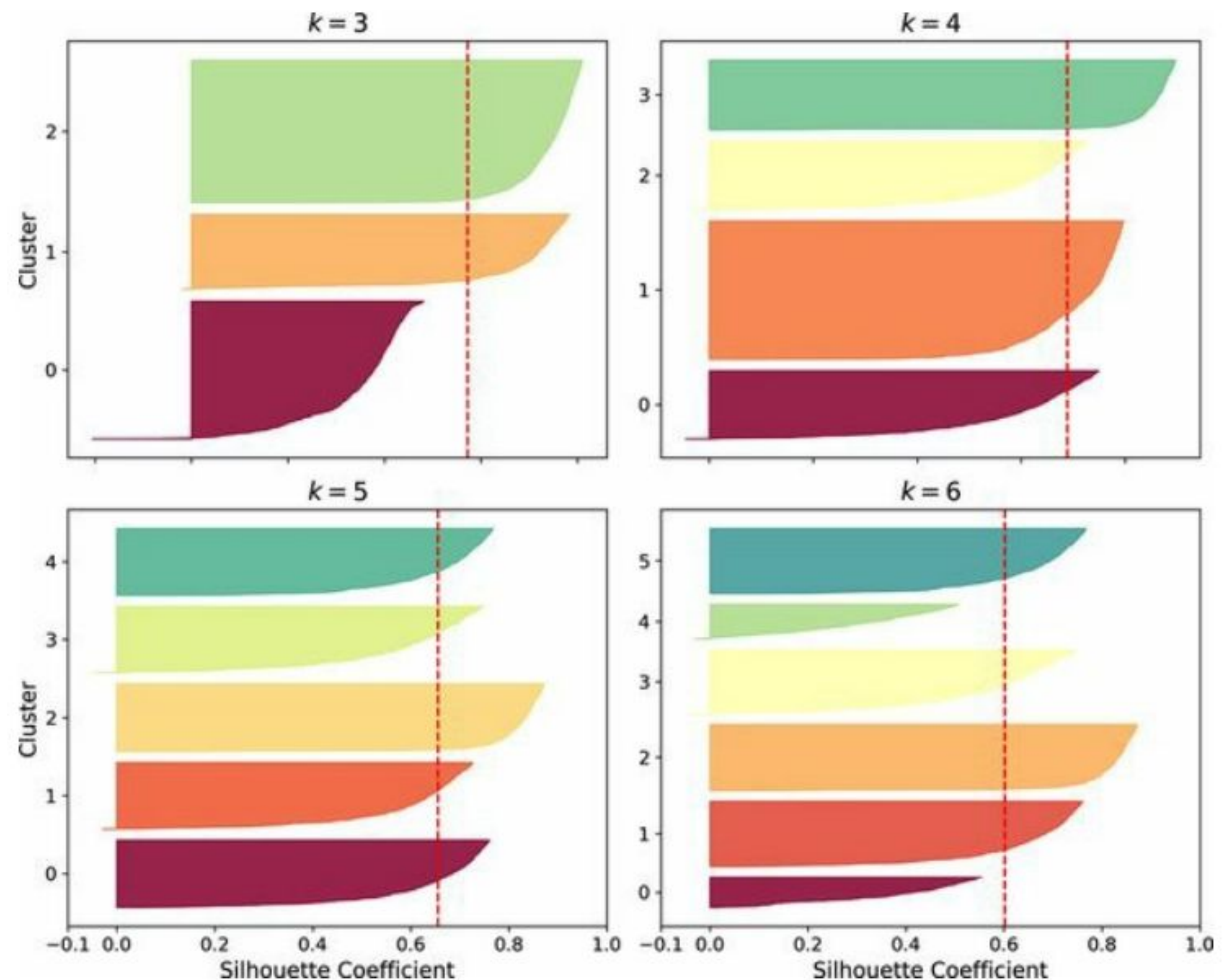


Imágen del Gerón

- Finalmente, valores negativos indican que el punto puede estar mal asignado, es decir, más cerca de otro cluster que del propio.

# K-Means

- Otra opción es mostrar el **diagrama de siluetas** que muestra el coeficiente de silueta de cada punto, ordenado por cluster y valor del coeficiente.
- La altura de cada forma indica el tamaño del cluster, y la anchura, los valores de silueta. La línea vertical marca el promedio general.
- Clusters con muchas instancias por debajo de esa línea se consideran de baja calidad, porque están demasiado cerca de otros.



(En este caso  $k=3$  o  $k=6$  son peores que  $k=4$  o  $k=5$ )

Imágen del Gerón



# K-Means

---

- Si bien K-Means es un método rápido y escalable tiene algunas limitaciones:
  - Es necesario ejecutar el algoritmo varias veces para evitar soluciones subóptimas.
  - Requiere especificar el número de clusters, lo cual puede ser complicado.
  - Tiene problemas con clusters de tamaños variables, densidades diferentes o formas no esféricas (para atacar algunos de estos problemas se puede probar con [modelos de mixturas gaussianas](#)).
- A pesar de estas limitaciones tiene algunas aplicaciones que vale la pena mencionar.

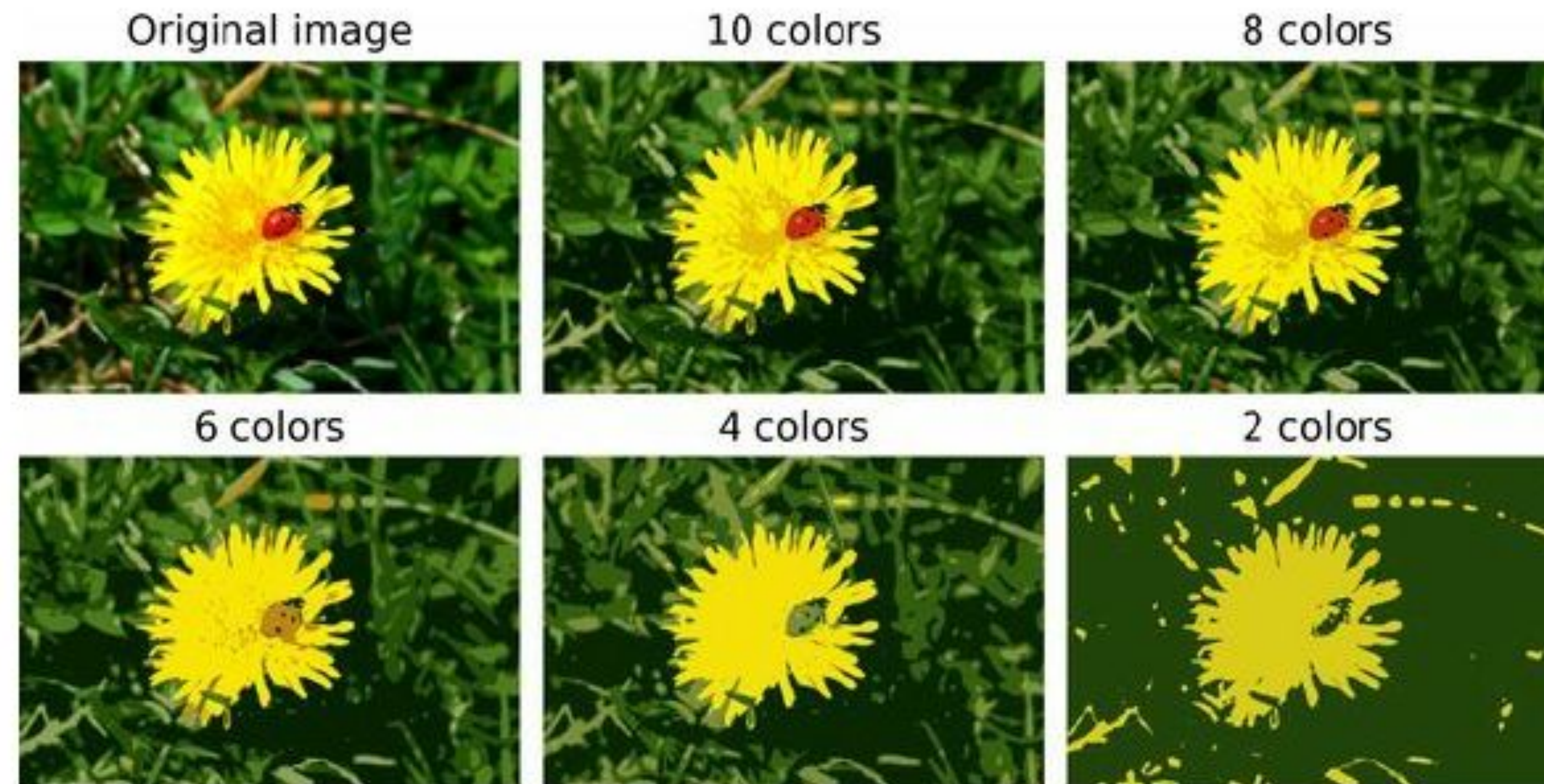
# Segmentación

---

- **Segmentación de imágenes**, consiste en dividir una imagen en regiones homogéneas.
  - **Por color:** agrupa píxeles con valores de color similares.
  - **Por espacio:** puede incluir también la posición (x, y) de los píxeles para formar regiones contiguas.
- Aunque hoy se usan redes convolucionales para segmentación avanzada, K-Means sigue siendo útil en problemas simples o como paso de preprocesamiento.



# Segmentación



Ejemplo de segmentación usando k-means con varios números de clusters de color

Imagen del Gerón



# Aprendizaje semi-supervisado

---

- Supongamos que tenemos el conjunto MNIST, pero no disponemos de etiquetas para todas las imágenes.
- Podríamos etiquetar una muestra al azar para entrenar el modelo, pero los resultados suelen ser pobres, ya que muchas de esas imágenes no representan bien la variedad del conjunto.
- Surge entonces la pregunta: ¿podemos elegir de manera más inteligente cuáles imágenes conviene etiquetar?



# Aprendizaje semi-supervisado

- Una estrategia consiste en aplicar K-Means para agrupar las imágenes en, por ejemplo, 50 clusters.
- Luego seleccionamos las imágenes más cercanas a cada centroide, que llamamos **imágenes representativas**.
- Etiquetamos solo esas imágenes y entrenamos el modelo con ellas, este enfoque suele lograr mejores resultados que etiquetar ejemplos al azar.
- Además, puede extenderse: las instancias cercanas a los centroides pueden etiquetarse automáticamente, e incluso descartar las más lejanas para evitar ruido. (Ver capítulo 8 del libro de Géron para más detalles.)

1	3	6	0	7	9	2	4	8	9
5	4	7	1	2	6	8	2	5	1
4	1	3	3	8	8	2	5	6	9
1	4	0	6	8	3	4	6	7	2
6	1	0	7	5	1	9	9	3	7

```
y_representative_digits = np.array([1, 3, 6, 0, 7, 9, 2, ..., 5, 1, 9, 9, 3, 7])
```

# Detección de anomalías

---

- En K-Means, los puntos se asignan a clusters según su cercanía al centroide.
  - **Los puntos más alejados del centroide pueden considerarse anómalos.**
- Se pueden analizar individualmente o definir un umbral de distancia para marcar posibles outliers.
- El coeficiente de silueta también ayuda a detectarlos:
  - Valores bajos o negativos indican puntos con poca cohesión en su cluster o más cercanos a otro.
  - Estos casos suelen corresponder a instancias atípicas o fronterizas.



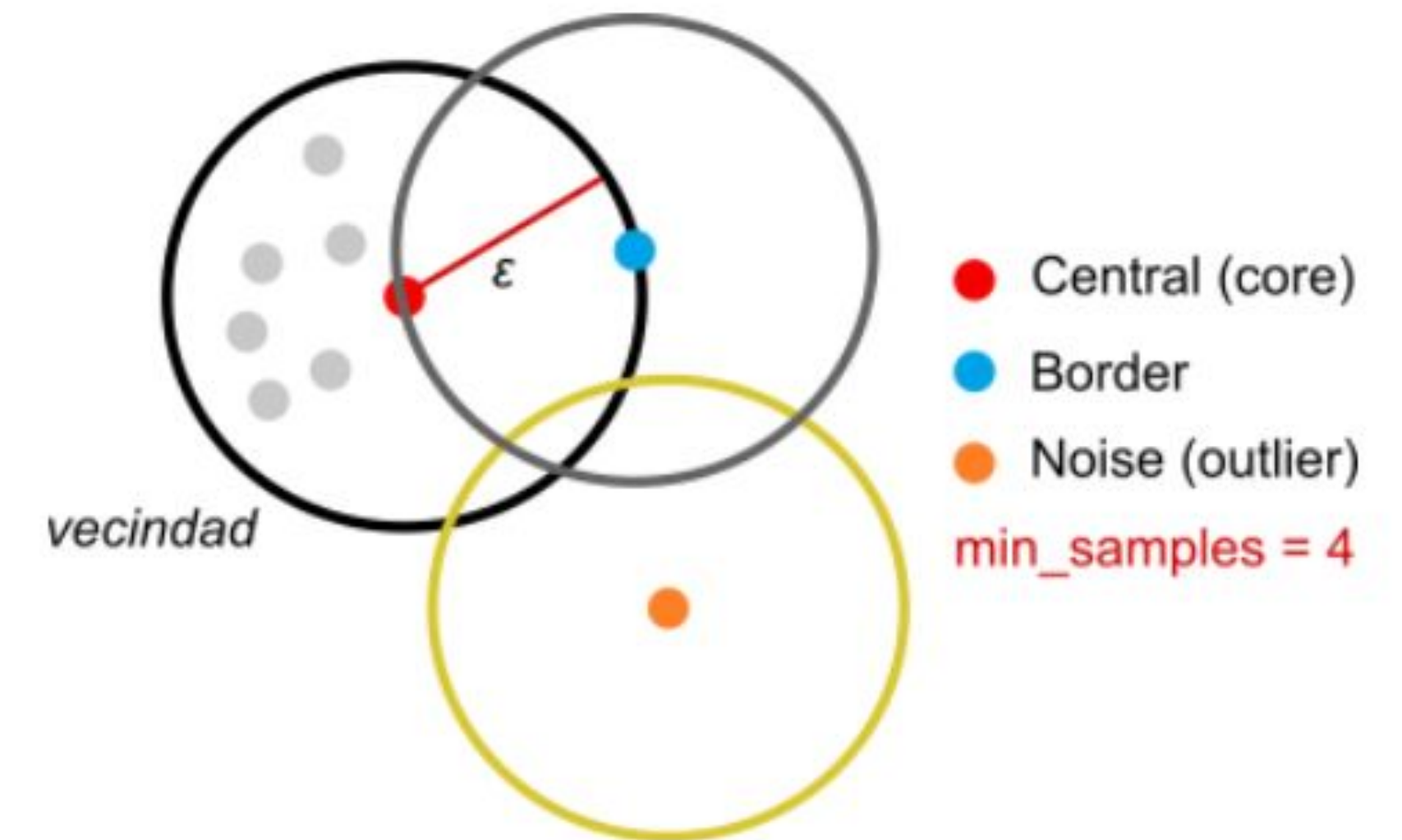
# Detección de anomalías

---

- Otra estrategia es comparar la distancia del punto al cluster más cercano:
  - **puntos lejos de todos los centroides son buenos candidatos a anomalías.**
- Estas instancias no se agrupan naturalmente con ningún conjunto de datos.
- Para cerrar, vale mencionar que existen métodos de clustering más sofisticados, como DBSCAN, que detectan outliers de forma explícita al identificar puntos aislados o en regiones de baja densidad.

# DBSCAN

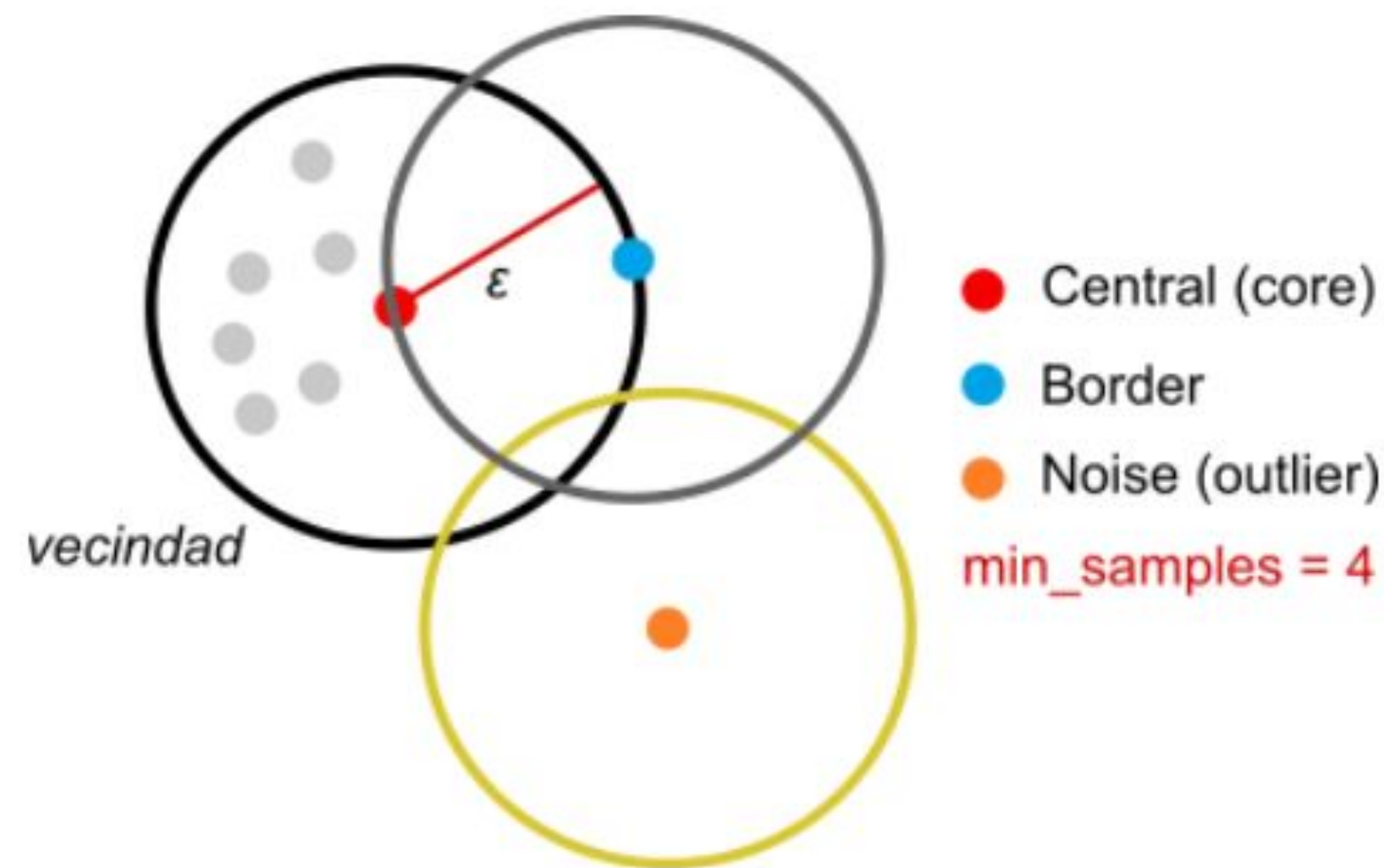
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise) es un algoritmo de clustering basado en la densidad de los datos.
- No requiere conocer el número de clusters a priori.
- Puede detectar clusters de forma arbitraria y distinguir ruido o anomalías.
- Se basa en dos hiperparámetros principales:
  - **$\epsilon$ : radio de vecindad**, define qué tan cerca deben estar dos puntos para considerarse vecinos.
  - **min\_samples**: mínimo número de puntos dentro de esa vecindad para que una instancia se considere núcleo (core).



[LINK: Clustering with DBSCAN, Clearly Explained!!!](#)



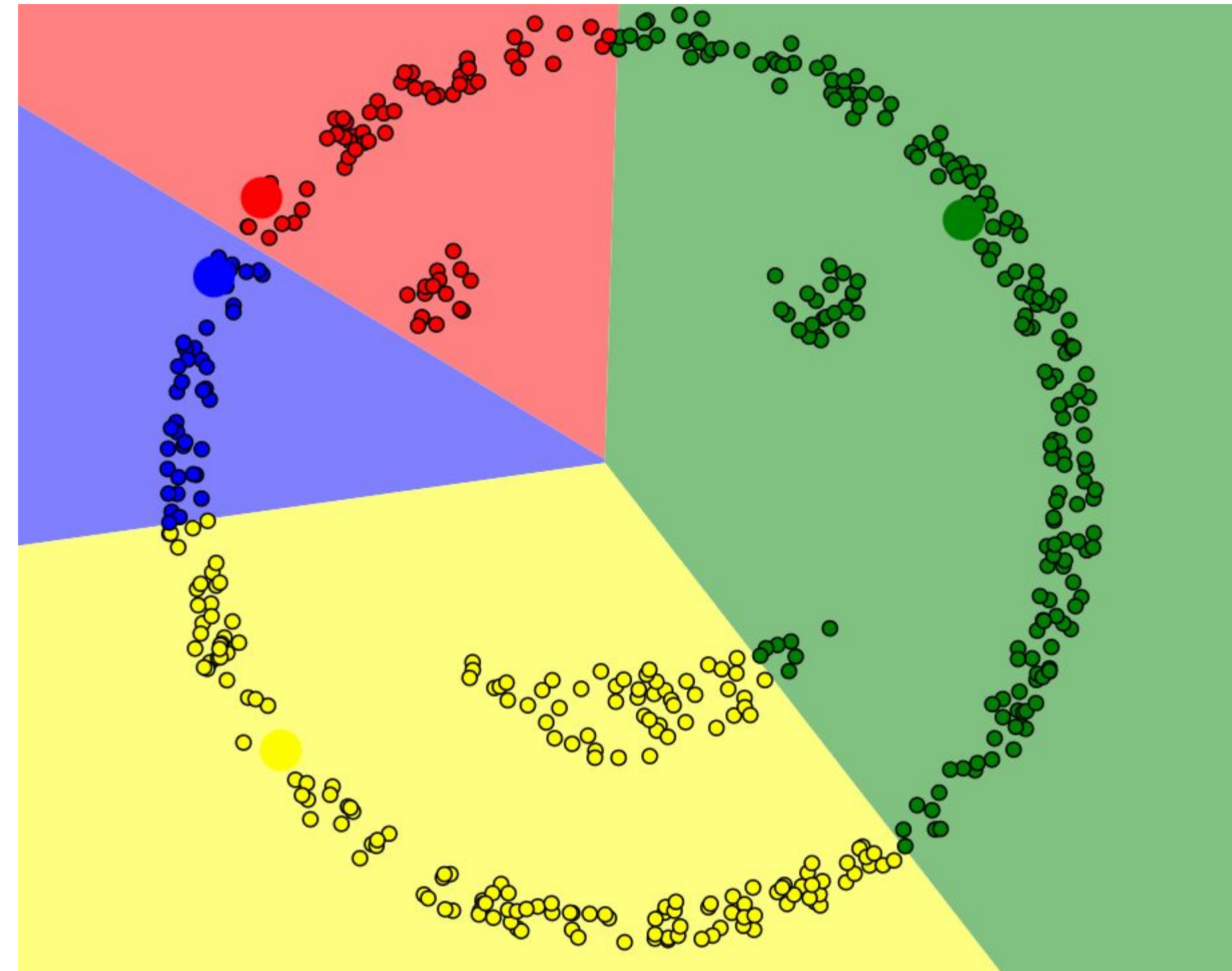
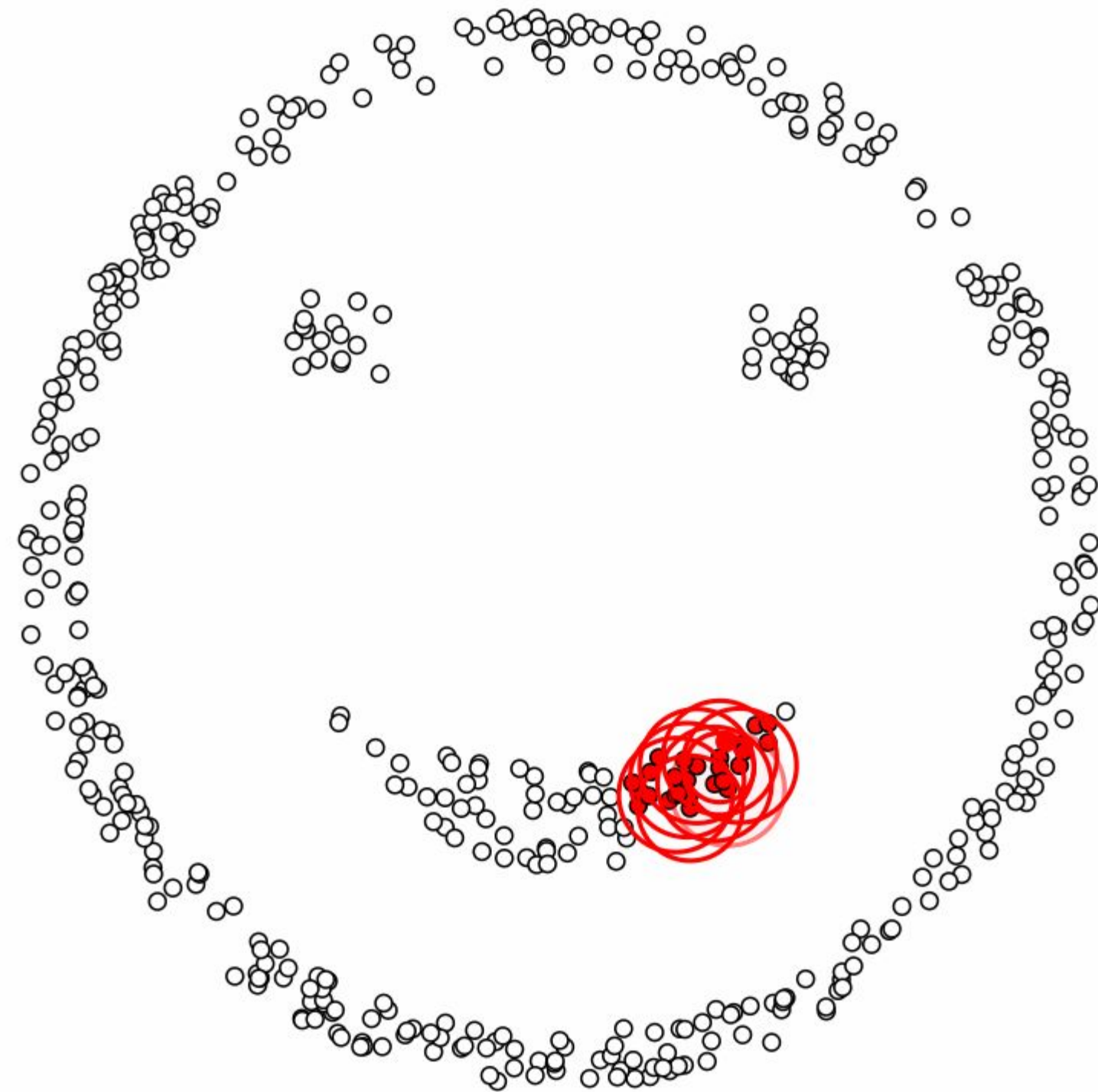
# DBSCAN



- Una instancia con al menos **min\_samples** vecinos dentro de su radio  $\epsilon$  se considera núcleo.
- Todos los puntos dentro de su vecindad pertenecen al mismo cluster.
- Si hay núcleos cuyas vecindades se solapan, se unen en un único cluster.
- Las instancias que no son núcleo y no están dentro de ninguna vecindad núcleo se consideran anomalías o ruido (*outliers*).
- DBSCAN es especialmente útil cuando los clusters tienen formas irregulares o densidades variables.

[LINK: Clustering with DBSCAN, Clearly Explained!!!](#)

# Comparación con K-Means



Fuente: [Understanding DBSCAN: A Practical Guide for Beginners with Business Applications](#)



# Parada técnica: Resumen hasta ahora

---

- **Aprendizaje no supervisado:** Búsqueda de patrones en datos para los que no tenemos etiquetas.
- **Reducción de la dimensionalidad:** Tener datos en muchas dimensiones puede tener sus complicaciones, tenemos algunas técnicas para buscar reducir esta dimensionalidad para usarlas como entradas para otros algoritmos o para visualizar.
- **Clustering:** Algoritmos que buscan agrupar datos similares, útiles para detección de anomalías, etiquetado, análisis de datos o incluso segmentación de imágenes.

