

# Prácticas de Aprendizaje Automático

## Práctica 2

### Experimentación con agrupamiento y detección de anomalías no supervisada

Pablo Mesejo y Salvador García

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial



UNIVERSIDAD  
DE GRANADA



# Práctica 2

- Ejercicio 1: Optimizando k-means y DBSCAN (4/10)
- Ejercicio 2: Problema de clustering (5/10)
- Ejercicio 3: Detección de anomalías (1/10)
- **Casuística próxima a la realidad:** te llega un problema y... ¿cómo lo resuelves?
  - **Análisis del Problema, Preprocesado** de los Datos, **Ajuste, Validación, y Discusión** de Resultados

# Fundamentos Teóricos

## Técnicas de aprendizaje no supervisado

- **Clustering**
  - Agrupar instancias similares en *clusters*.
- **Detección de anomalías y valores atípicos**
  - Aprender qué aspecto tienen los datos "normales" y, a partir de ahí, **detectar los casos anómalos**.
  - Estos casos se denominan anomalías ("outliers"), mientras que los casos normales se denominan "inliers".
- **Estimación de la densidad**
  - Estimar la *función de densidad de probabilidad* (PDF) del proceso aleatorio que ha generado el conjunto de datos.
  - Se utiliza habitualmente como parte de la detección de anomalías: **es probable que los casos situados en regiones de baja densidad sean anomalías**. También es útil para el análisis y la visualización de datos.

# Clustering

- Tarea de identificar instancias similares y asignarlas a **clusters** (conglomerados), o **grupos de instancias similares**.
- Al igual que en clasificación, cada instancia se asigna a un grupo. Sin embargo, a diferencia de la clasificación, el **clustering es una tarea no supervisada**.

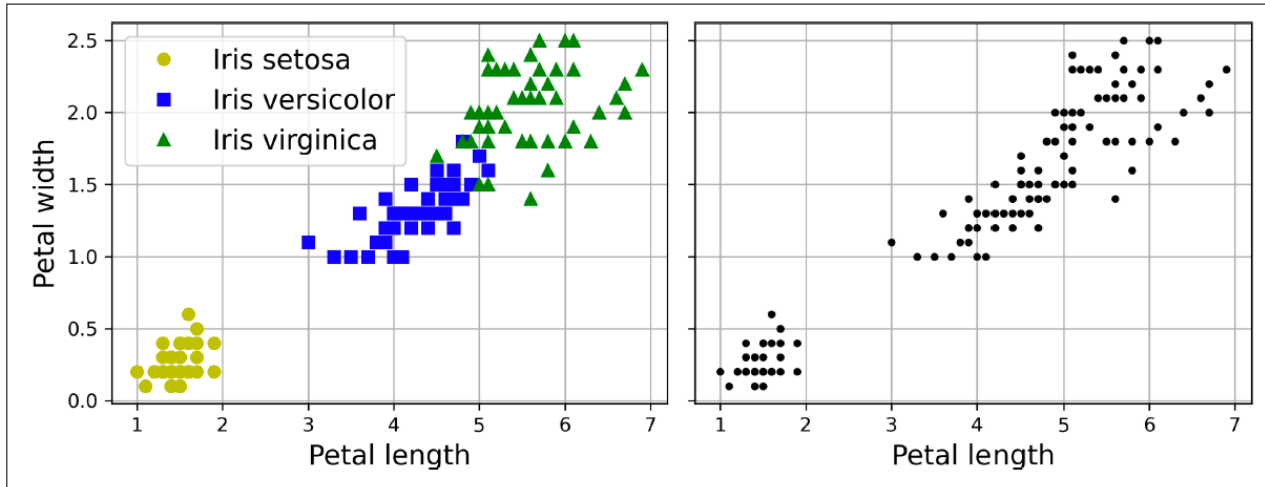


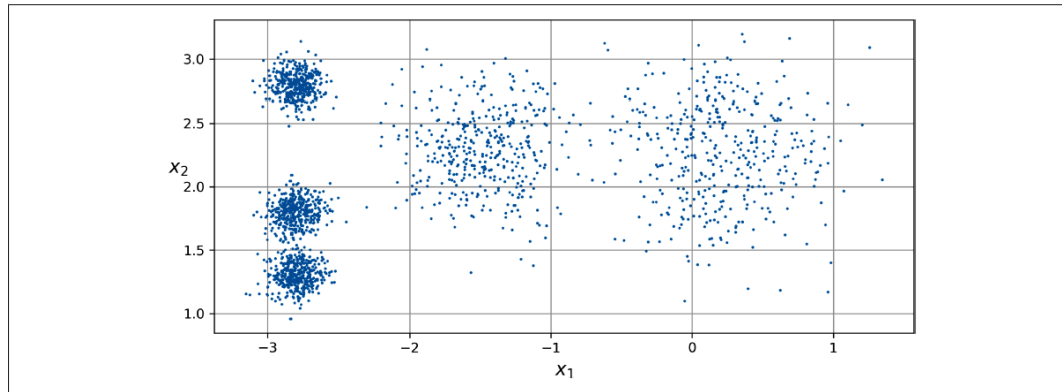
Figure 9-1. Classification (left) versus clustering (right)

# Aplicaciones del clustering

- Segmentación de clientes
- Análisis de datos
- Reducción de dimensionalidad
  - Una vez que se ha agrupado un conjunto de datos, suele ser posible medir la *afinidad* de cada instancia con cada cluster. **La afinidad es cualquier medida de lo bien que encaja una instancia en un cluster.** El vector de características  $x$  de cada instancia puede sustituirse por el vector de sus afinidades con los clusters. Si hay  $k$  conglomerados/clusters, este vector es *k-dimensional*.
- Ingeniería de características (*feature engineering*)
- Detección de anomalías
- Aprendizaje semi-supervisado
- Motores de búsqueda
- Segmentación de imágenes
  - Agrupando los píxeles según su color y sustituyendo después el color de cada píxel por el color medio de su agrupación, es posible reducir considerablemente el número de colores diferentes de una imagen (*color quantization*).

# k-means

- Considera el conjunto de datos sin etiquetar representado aquí: puedes ver claramente cinco grupos de instancias.



- El algoritmo k-means es un algoritmo sencillo capaz de agrupar este tipo de conjuntos de datos de forma muy rápida y eficaz, a menudo en solo unas pocas iteraciones.
- Propuesto por Stuart Lloyd en los laboratorios Bell en 1957. No se publicó hasta 1982. En 1965, Edward W. Forgy había publicado prácticamente el mismo algoritmo, por lo que a veces se hace referencia a k-means como el algoritmo Lloyd-Forgy.

Lloyd, Stuart. "Least squares quantization in PCM." *IEEE transactions on information theory* 28.2 (1982): 129-137.

Forgy, Edward W. "Cluster analysis of multivariate data: efficiency versus interpretability of classifications." *biometrics* 21 (1965): 768-769.

Steinhaus, Hugo. "Sur la division des corps matériels en parties." *Bull. Acad. Polon. Sci* 1.804 (1956): 801.

# k-means

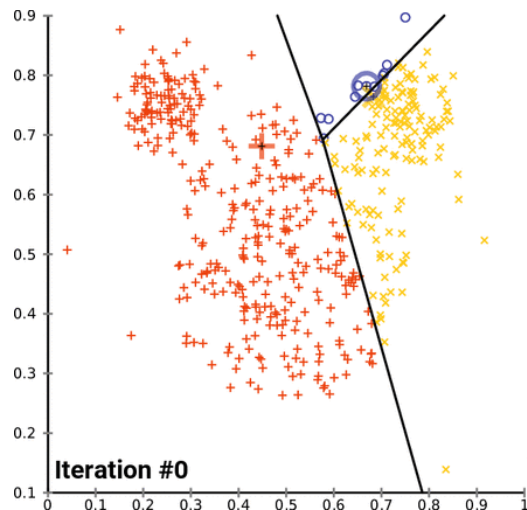
- El algoritmo k-means es uno de los algoritmos de clustering más rápidos y sencillos.

- 1) **Se inicializan  $k$  centroides aleatoriamente:** por ejemplo, se eligen al azar  $k$  instancias distintas del conjunto de datos y se colocan los centroides en sus ubicaciones.
- 2) **Se repite hasta convergencia** (es decir, hasta que los centroides dejen de moverse):
  - **Asignar cada instancia al centroide más cercano.**
  - **Actualiza los centroides para que sean la media de las instancias que se les asignan.**

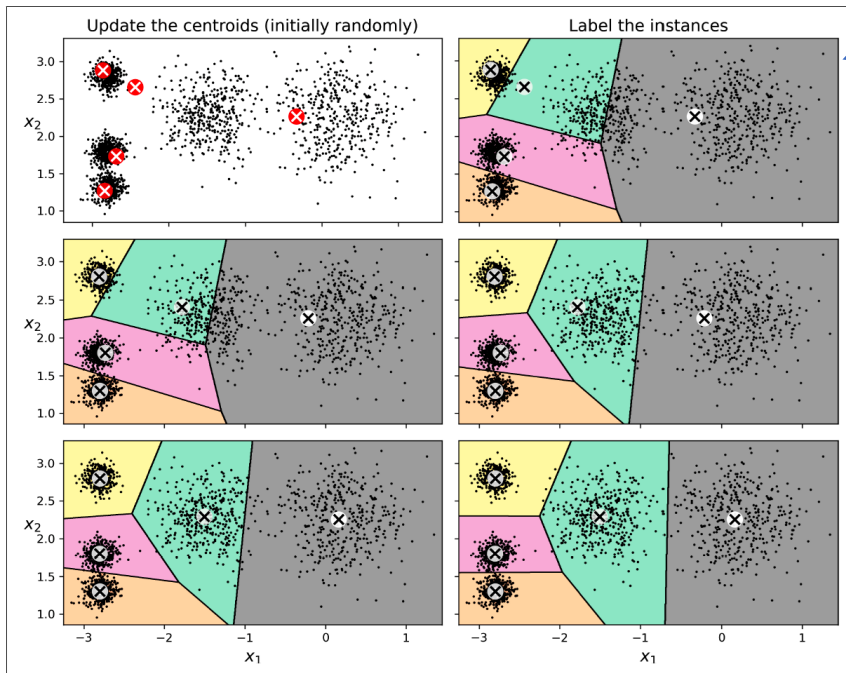
- La complejidad computacional del algoritmo suele ser **lineal con respecto al número de instancias  $m$ , el número de conglomerados/clusters  $k$  y el número de dimensiones  $n$ .**
  - Sin embargo, esto solo es cierto cuando los datos tienen una estructura de conglomerados. Si no es así, en el peor caso, **la complejidad puede aumentar exponencialmente con el número de instancias.** En la práctica, esto rara vez ocurre, y k-means suele ser uno de los algoritmos de clustering más rápidos.
- La clase `KMeans` de scikit-learn utiliza por defecto una técnica de inicialización optimizada. Para obtener el algoritmo original de K-Means, debe establecer `init="random"` y `n_init=1`. Véase <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

# k-means

Ejemplo de ejecución del algoritmo k-means:



Otro ejemplo de ejecución del algoritmo k-means (en este caso, durante 3 iteraciones) para ver cómo se mueven los centroides:

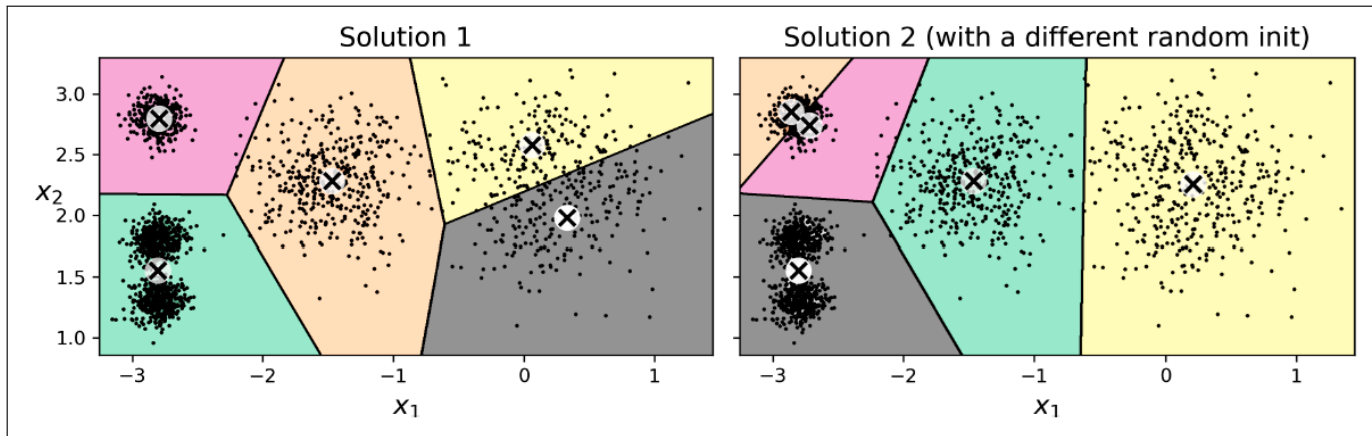


La partición del espacio en clusters genera una [teselación de Voronoi](#).



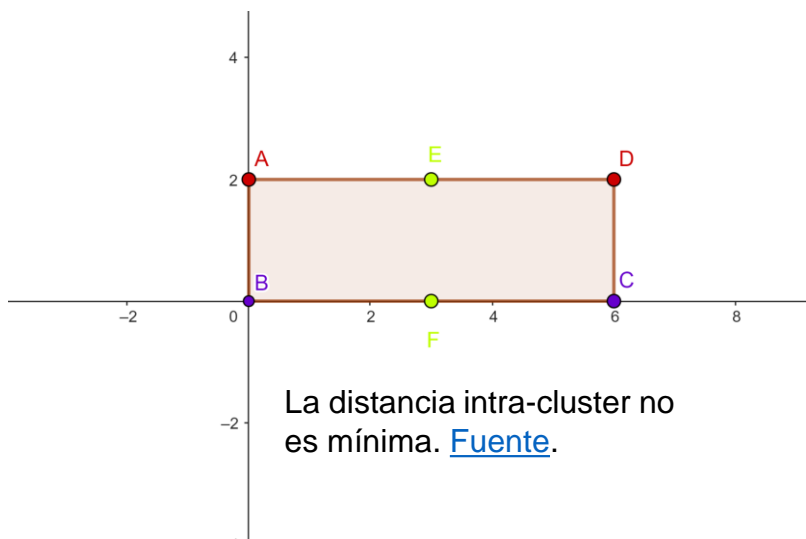
# k-means

- Aunque **se garantiza que el algoritmo converge**, puede que no converja a la solución correcta (es decir, **puede que converja a un óptimo local**):
  - que lo haga o no depende de la inicialización del centroide.
- La siguiente figura muestra dos soluciones subóptimas a las que puede converger el algoritmo si no se tiene suerte con el paso de inicialización aleatoria.

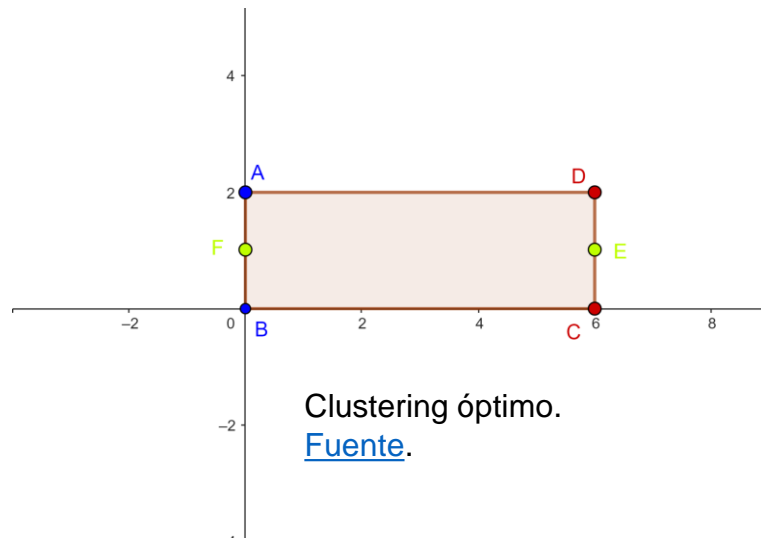


# k-means

- El objetivo de k-means es minimizar la inercia o varianza intra-cluster (es decir, la suma de las distancias al cuadrado desde cada punto a su centroide). <https://scikit-learn.org/stable/modules/clustering.html#k-means>



VS



# k-means++

- Proceso de **inicialización más inteligente** que seleccionar **centroides** aleatorios.
- Mejora: seleccionar centroides distantes entre sí. **Esto hace que k-means converja en menos ocasiones a una solución subóptima.**
- El trabajo de Arthur y Vassilvitskii (2007) demostró que el cálculo adicional necesario para **el paso de inicialización más inteligente** merece la pena porque **permite reducir drásticamente el número de veces que hay que ejecutar el algoritmo para encontrar la solución óptima.**

Arthur and Vassilvitskii. "k-means++: The advantages of careful seeding." *Soda*. Vol. 7. 2007.  
<https://theory.stanford.edu/~sergei/papers/kMeansPP-soda.pdf>

# k-means++

El algoritmo de inicialización de *k-means++* funciona así:

1. El primer centroide  $\mathbf{c}^{(1)}$  es elegido uniformemente al azar del conjunto de datos.
2. Para cada instancia  $\mathbf{x}^{(i)}$  todavía no seleccionada, calculamos  $D(\mathbf{x}^{(i)})$ , la distancia entre dicha instancia  $\mathbf{x}^{(i)}$  y el centroide más cercano ya elegido.
3. Escogemos una instancia  $\mathbf{x}^{(i)}$  como nuevo centroide,  $\mathbf{c}^{(i)}$ , con probabilidad proporcional a  $D(\mathbf{x}^{(i)})^2$

$$D(\mathbf{x}^{(i)})^2 / \sum_{j=1}^m D(\mathbf{x}^{(j)})^2$$

Cuanto mayor sea esta distancia, más posibilidades hay de escoger esta instancia como centroide

Esta distribución de probabilidad busca que **las instancias más alejadas de los centroides ya elegidos tengan muchas más probabilidades de ser seleccionadas como nuevos centroides iniciales.**

4. Repetimos los pasos 2 y 3 hasta que se hayan elegido los  $k$  centroides.

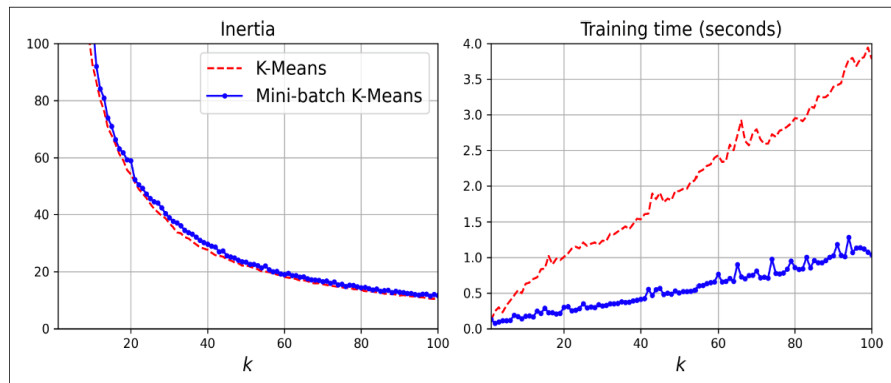
La clase `KMeans` de scikit-learn utiliza este método de inicialización por defecto.  
<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

# k-means acelerado

- Charles Elkan propuso en 2003 otra mejora del algoritmo k-means. En algunos conjuntos de datos grandes con muchos clusters, **el algoritmo puede acelerarse evitando muchos cálculos de distancia innecesarios**.
- Elkan lo consiguió explotando la **desigualdad triangular** (es decir, que una línea recta es siempre la distancia más corta entre dos puntos) y llevando un **registro de los límites inferior y superior de las distancias entre instancias y centroides**.
- Sin embargo, el algoritmo de Elkan no siempre acelera el entrenamiento y, dependiendo del problema concreto, a veces incluso puede ralentizarlo.
- Para probarlo en scikit-learn, se puede indicar `algorithm="elkan"` como hiperparámetro.

# Mini-batch k-means

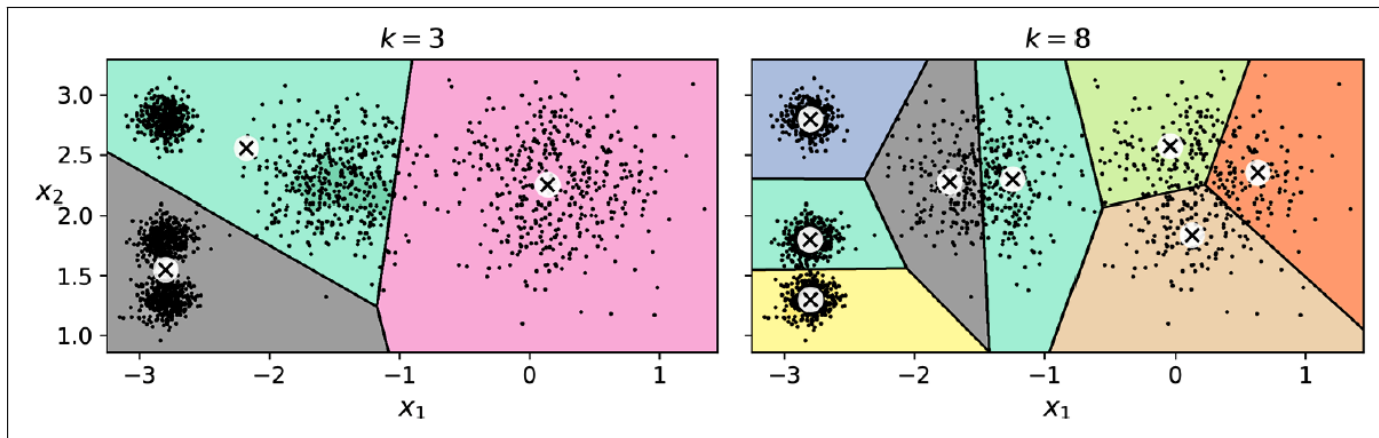
- En lugar de utilizar el conjunto de datos completo, **el algoritmo puede utilizar minilotes, moviendo los centroides solo ligeramente en cada iteración.**
- **Esto acelera la ejecución del algoritmo** (normalmente x3-x4) y permite agrupar conjuntos de datos enormes que no caben en la memoria.
- Aunque el algoritmo **mini-batch k-means es mucho más rápido que el algoritmo k-means normal, su inercia suele ser ligeramente peor.**
  - Recordemos que la inercia es la suma de las distancias al cuadrado entre las instancias y sus centroides más cercanos. Nos interesa que la inercia sea pequeña, dado que buscamos minimizar la distancia intra-cluster.



# Encontrar el número óptimo de clusters

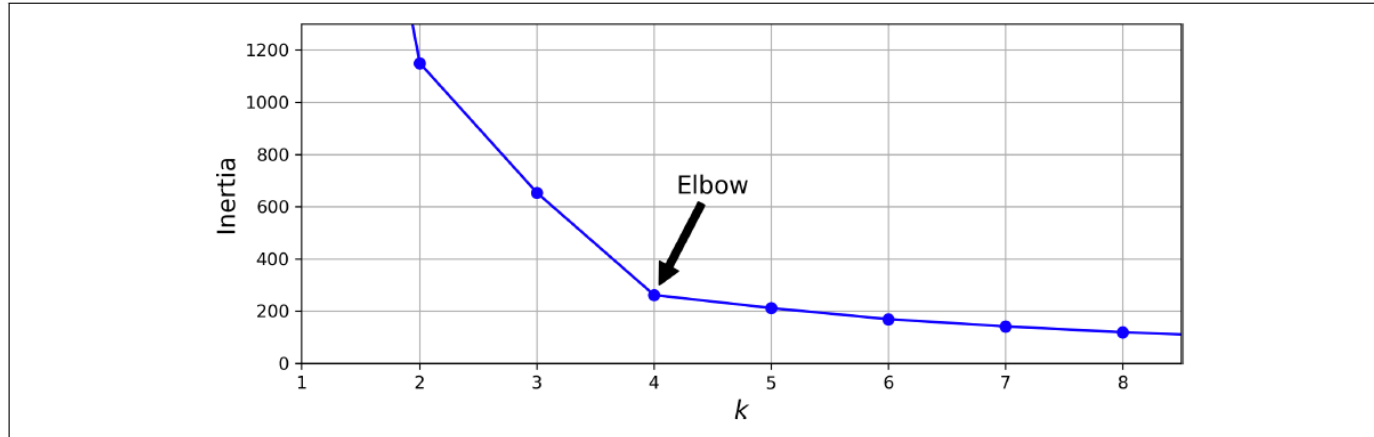
En general, **no será fácil saber cómo establecer  $k$** , y el resultado puede ser bastante malo si se establece en un valor incorrecto.

Como se puede ver aquí, para este conjunto de datos, establecer  $k$  en 3 u 8 da como resultado modelos manifiestamente mejorables.



# Encontrar el número óptimo de clusters

- **La inercia** no es una buena métrica de rendimiento cuando se trata de elegir  $k$  porque **sigue bajando a medida que aumentamos  $k$** .
- Cuantos más conglomerados haya, más cerca estará cada instancia de su centroide más próximo y, por tanto, menor será la inercia. Tracemos **la inercia en función de  $k$** . Al hacerlo, **la curva suele contener un punto de inflexión llamado *elbow* (codo)**.



En otras palabras, buscamos un equilibrio entre minimizar la inercia y minimizar el número de clusters/ $k$



# Encontrar el número óptimo de clusters

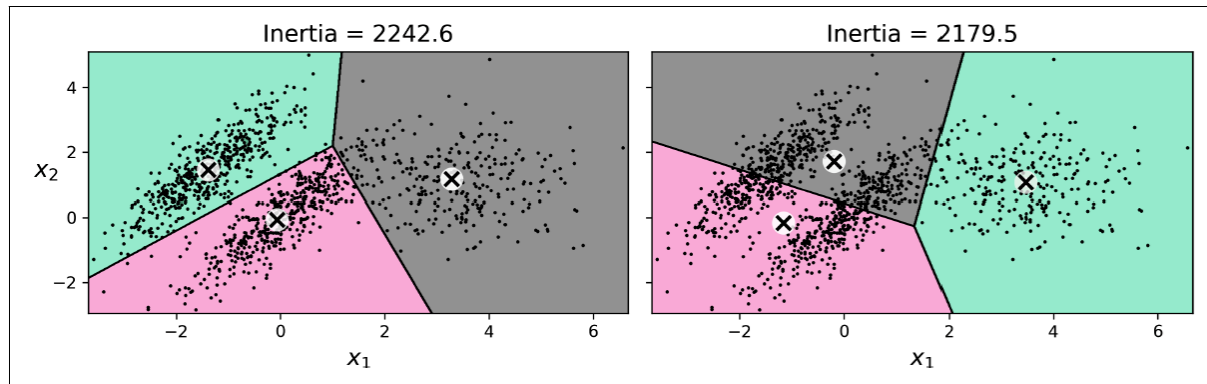
- La técnica del codo es bastante tosca. Un enfoque más preciso (pero más costoso computacionalmente) es utilizar la **puntuación de silueta** (*Silhouette score*), que es el **coeficiente de silueta medio de todas las instancias**.
- El coeficiente de silueta de una instancia es igual a  $(b - a) / \max(a, b)$ , donde
  - $a$  es la distancia media de una instancia a las demás instancias del mismo cluster (es decir, la distancia media intra-cluster)
  - $b$  es la distancia media de una instancia al cluster más cercano (es decir, la distancia media a las instancias del cluster más cercano, definido como aquel que minimiza  $b$ , excluyendo el cluster propio de la instancia).
- El coeficiente de silueta puede variar entre -1 y +1.
  - Un coeficiente cercano a **+1** **significa que la instancia está bien dentro de su propio cluster** y lejos de otros clústeres.
  - Un coeficiente cercano a **0** **significa que está cerca del límite de un cluster**.
  - Un coeficiente cercano a **-1** **significa que la instancia puede haber sido asignada a un cluster equivocado**.

# Límites de k-means

- k-means no es perfecto.
  - Es **necesario ejecutar el algoritmo varias veces** para evitar soluciones subóptimas → ¡la **inicialización** es clave!
  - **Hay que especificar el número de clusters.**
  - A nivel práctico, **no se comporta muy bien** cuando los clusters tienen **tamaños variables, densidades diferentes o formas no esféricas.**

Aquí mostramos cómo k-means agrupa un conjunto de datos que contiene tres clusters elipsoidales de diferentes dimensiones, densidades y orientaciones.

Ninguna de estas dos soluciones es buena. **La solución de la izquierda** es mejor, pero **asigna el 25% del cluster central (forma elíptica) al cluster de la derecha**. **La solución de la derecha es peor, aunque su inercia sea menor.**



# DBSCAN



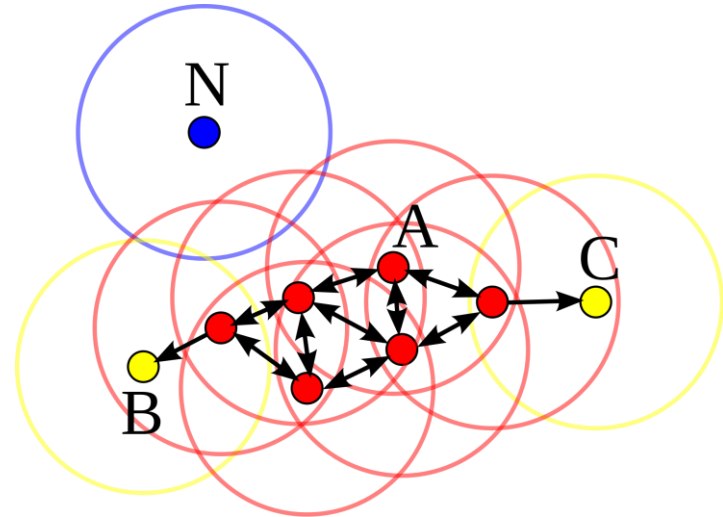
- El algoritmo de *agrupamiento espacial basado en densidad de aplicaciones con ruido* (DBSCAN) **define los clusters como regiones continuas de alta densidad.**
  - Para cada instancia, **el algoritmo cuenta cuántas instancias se encuentran a una pequeña distancia  $\epsilon$  de ella.**
    - Esta región se denomina *vecindad*  $\epsilon$  de la instancia.
  - **Si una instancia tiene al menos *min\_muestras* instancias en su vecindad  $\epsilon$  (incluida ella misma), se considera una *instancia central* (core point).**
    - Las instancias centrales son aquellas que se encuentran en regiones densas.
  - **Todas las instancias en la vecindad de un *core point* pertenecen al mismo cluster.**
  - Si una instancia no es central y no tiene ninguna en su vecindad → **anomalía.**
- Este algoritmo funciona bien si todos los clusters están bien separados por regiones de baja densidad.

# DBSCAN



- Ejemplo de funcionamiento:

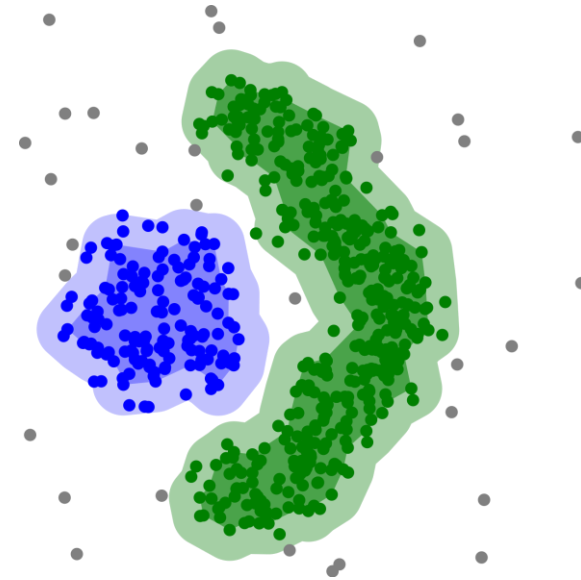
- **El punto A y todos los demás en rojo son instancias centrales** (*core points*), dado que en su vecindad tienen, al menos, 4 instancias (**min\_muestras**).
- $\epsilon$  es el radio de cada punto (dentro del cual contamos el número de instancias).
- **Todos los puntos en rojo** son alcanzables (*reachable*) entre ellos, por lo que **conforman un mismo cluster**.
- Los puntos B y C, en amarillo, no son instancias centrales, pero **son alcanzables desde A**, por lo que **también pertenecen al cluster**.
- **N no es alcanzable, y se considera un outlier**.



<https://en.wikipedia.org/wiki/DBSCAN#/media/File:DBSCAN-Illustration.svg>

# DBSCAN

- DBSCAN es un algoritmo sencillo pero potente, capaz de identificar cualquier número de conglomerados de cualquier forma. **Es robusto frente a valores atípicos y solo tiene dos hiperparámetros ( $\epsilon$  y  $\min\_samples$ ).**
- **DBSCAN puede tener dificultades:**
  - si la densidad varía significativamente entre los conglomerados, o
  - si no hay una región de densidad suficientemente baja alrededor de algunos conglomerados
- Su complejidad computacional es aproximadamente  $O(m^2n)$ , por lo que no se adapta bien a grandes conjuntos de datos.  $m$ : número de instancias;  $n$ : número de dimensiones
- También existe DBSCAN jerárquico (**HDBSCAN**):
  - Suele ser mejor que DBSCAN a la hora de encontrar clústeres de distintas densidades.
  - <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.HDBSCAN.html>



<https://en.wikipedia.org/wiki/DBSCAN#/media/File:DBSCAN-density-data.svg>

# Otros algoritmos de clustering

- **Clustering aglomerativo**

- Se construye una **jerarquía de clusters de abajo a arriba**. En cada iteración, se conecta el par de clusters más cercano (empezando por instancias individuales).
- Este enfoque **puede capturar clusters de diversas formas; también produce un árbol de clusters flexible e informativo** (donde las hojas son las instancias individuales) y puede utilizarse con cualquier distancia entre pares.
- **Puede escalar bien a un gran número de instancias si se proporciona una matriz de conectividad**, que es una matriz  $m \times m$  dispersa que indica qué pares de instancias son vecinos (por ejemplo, devuelta por `sklearn.neighbors.kneighbors_graph()`). Sin una matriz de conectividad, el algoritmo no se adapta bien a grandes conjuntos de datos.
- <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html>

- **BIRCH**

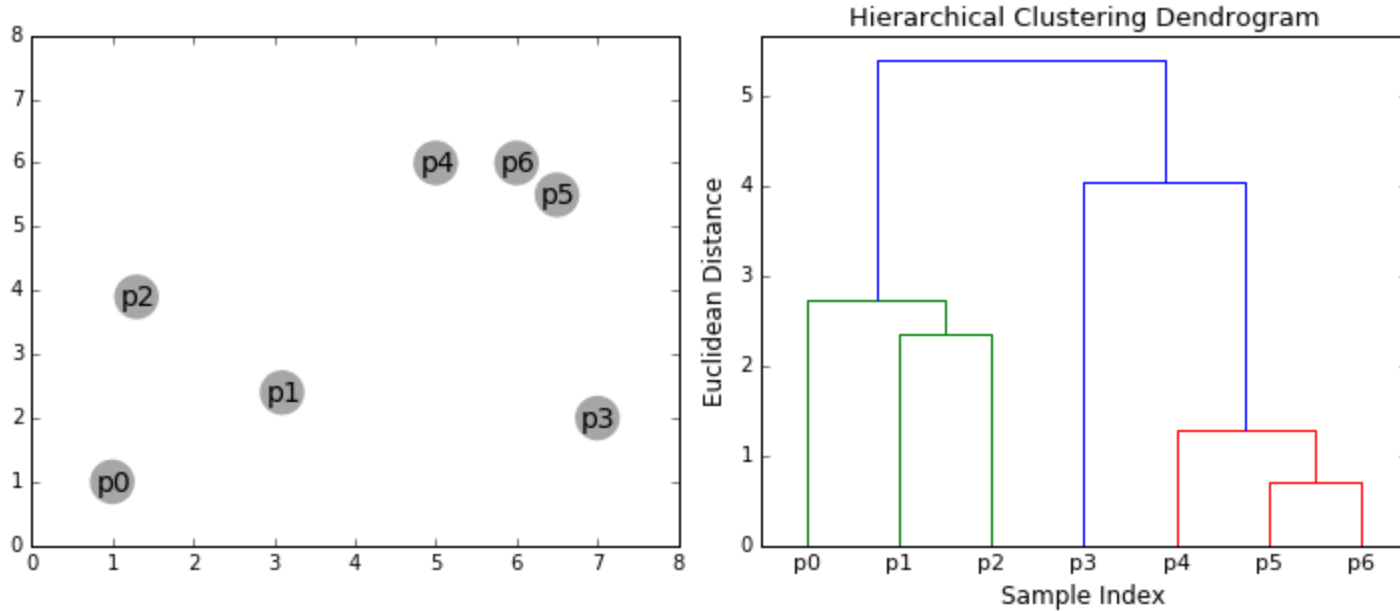
- El algoritmo BIRCH (*balanced iterative reducing and clustering using hierarchies*) se diseñó específicamente para **conjuntos de datos muy grandes**, y puede ser más rápido que k-means por lotes, con resultados similares, siempre que el número de características no sea demasiado grande (<20).
- Durante el entrenamiento, **construye una estructura de árbol que contiene la información suficiente para asignar rápidamente cada nueva instancia a un cluster**, sin tener que almacenar todas las instancias en el árbol.
- <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.Birch.html>

Zhang, Ramakrishnan and Livny. "BIRCH: an efficient data clustering method for very large databases." *ACM sigmod record* 25.2 (1996): 103-114.

Murtagh and Contreras. "Algorithms for hierarchical clustering: an overview." *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2.1 (2012): 86-97.

Murtagh and Contreras. "Algorithms for hierarchical clustering: an overview, II." *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 7.6 (2017): e1219.

# Otros algoritmos de clustering

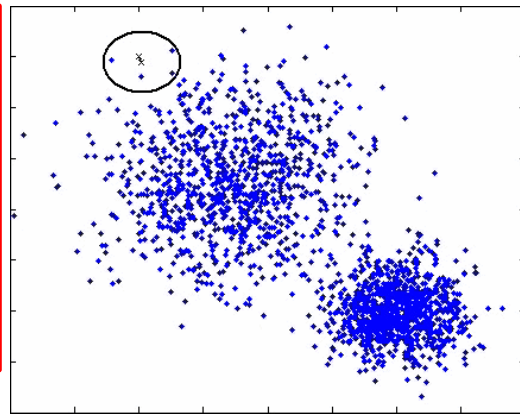


<https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/>

# Otros algoritmos de clustering

- **Mean-shift**

- 1) colocamos un círculo centrado en cada instancia.
- 2) para cada círculo, calculamos la media de todas las instancias situadas dentro de él y desplazamos el círculo de modo que quede centrado en la media (de ahí lo de *mean-shift*).
- 3) Iteramos este último paso hasta que todos los círculos dejan de moverse. Todas las instancias cuyos círculos se asientan en el mismo lugar (o lo suficientemente cerca) se asignan al mismo cluster.



[Fuente.](#)

- Desplazamos los círculos en la dirección de mayor densidad, hasta que cada uno de ellos haya encontrado un máximo local de densidad. Como DBSCAN, se basa en la estimación de densidad local.
- Mean-shift puede encontrar cualquier número de clusters de cualquier forma, y tiene solo un **hiperparámetro** (el radio de los círculos, llamado **bandwidth**).
- Por desgracia, su complejidad computacional es  $O(m^2n)$ , por lo que no es adecuado para grandes conjuntos de datos.  $m$ : número de instancias;  $n$ : número de dimensiones
- <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MeanShift.html>



# Otros algoritmos de clustering

- **Affinity Propagation**

- Las instancias intercambian mensajes entre sí repetidamente hasta que cada una ha elegido a otra instancia (o a sí misma) para que la represente. Estas instancias elegidas se denominan *exemplars*. Cada ejemplar y todas las instancias que lo eligieron forman un cluster.
- En la práctica, el paso de mensajes se reduce a actualizar dos matrices:
  - “Responsibility” ( $r(i, k)$ ): cómo de bien  $x_k$  serviría como ejemplar de  $x_i$ .
  - “Availability” ( $a(i, k)$ ): cómo de apropiado sería  $x_i$  para escoger  $x_k$  como su ejemplar.
- La propagación por afinidad tiende a elegir ejemplares situados cerca del centro de los conglomerados, de forma parecida a k-means.
  - Pero, a diferencia de k-means (y al igual que, p.ej., mean-shift), **no hay que elegir un número de clusters de antemano**: se determina durante el entrenamiento.
- La propagación por afinidad puede manejar bien clusters de diferentes tamaños.
- Este algoritmo tiene una complejidad computacional de  $O(m^2)$ , por lo que no es adecuado para grandes conjuntos de datos.  $m$ : número de instancias.
- <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AffinityPropagation.html>

# Otros algoritmos de clustering

- **Clustering espectral**

- Se crea una matriz de similitud/afinidad entre instancias y una representación (*embedding*) de baja dimensión a partir de ella.
- Luego, utiliza otro algoritmo de clustering en este espacio de baja dimensión (la implementación de Scikit-Learn utiliza k-means).
- El clustering espectral puede capturar estructuras de clustering complejas, y también puede utilizarse para segmentar grafos (por ejemplo, para identificar clusters de amigos en una red social).
- No se adapta bien a un gran número de instancias y no se comporta bien cuando los clusters tienen tamaños muy diferentes.
- <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html>

Ng, Jordan and Weiss. "On spectral clustering: Analysis and an algorithm." *Advances in neural information processing systems* 14 (2001).

Shi and Malik. "Normalized cuts and image segmentation." *IEEE Transactions on pattern analysis and machine intelligence* 22.8 (2000): 888-905.

# Medidas de calidad del clustering

## Silueta (*Silhouette index*)

- ❑ Cuantifica cuán similar es un objeto a su propio cluster (cohesión) en comparación con otros clusters (separación).
- ❑ Para cada instancia, se calcula la distancia con todos los objetos de su cluster (a), la distancia con los objetos del cluster más cercano (b), y luego, el ratio entre ambas.
- ❑ En  $[-1, 1]$ : un valor alto indica que el objeto está bien emparejado con su propio cluster.

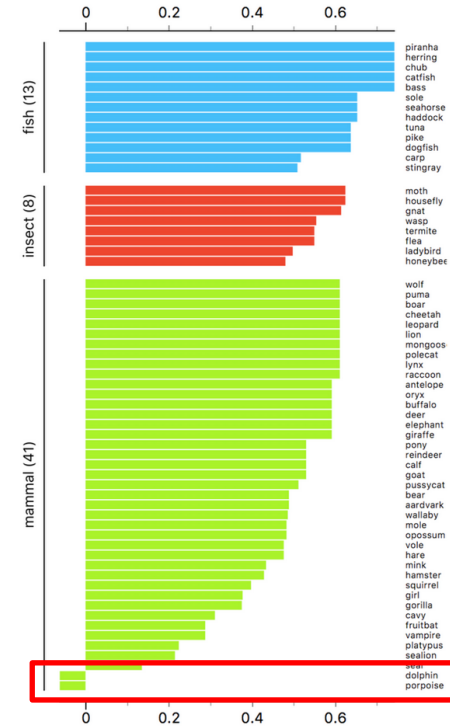
Distancia media entre  $i$  y todos los otros puntos en el mismo cluster.

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j)$$

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j)$$

Mínima distancia media entre  $i$  y todos los puntos en cualquier otro cluster.

$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$

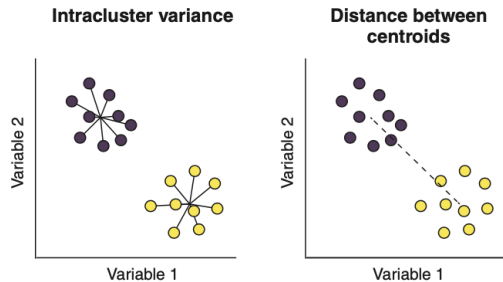


[https://en.wikipedia.org/wiki/Silhouette\\_\(clustering\)#/media/File:Silhouette-plot-orange.png](https://en.wikipedia.org/wiki/Silhouette_(clustering)#/media/File:Silhouette-plot-orange.png)

# Medidas de calidad del clustering

## Davies-Bouldin index

- ❑ Cuantifica la “separabilidad media” de un cluster frente a la del cluster más cercano.
- ❑ Desviación típica dentro de los clústeres (dispersión o *scatter*) dividido por la separación entre centroides.
- ❑ Mejor cuanto más pequeño: queremos alta separación entre clusters y baja variabilidad intra-cluster.



H.I. Rhys. *Machine Learning with R, the tidyverse, and mlr*. Manning, 2020.

Separación entre los centroides de los cluster  $j$  y  $k$

$$\text{scatter}_k = \left( \frac{1}{n_k} \sum_{i \in k} (x_i - c_k)^2 \right)^{1/2}$$

$$\text{separation}_{j,k} = \left( \sum_{1 \leq j \leq k}^N (c_j - c_k)^2 \right)^{1/2}$$

$$\text{ratio}_{j,k} = \frac{\text{scatter}_j + \text{scatter}_k}{\text{separation}_{j,k}}$$

$R_k$  es el máximo  $\text{ratio}_{j,k}$

Es decir, para cada cluster  $k$ , nos quedamos con el máximo  $\text{ratio}_{j,k}$

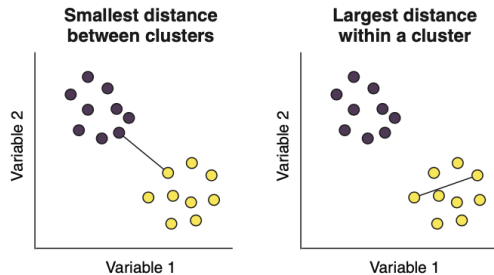
$$DB = \frac{1}{N} \sum_{k=1}^N R_k$$

$N$  es el número de clusters

# Medidas de calidad del clustering

## Dunn index

- ❑ **Ratio entre la distancia mínima entre puntos de diferentes clusters y el diámetro máximo de los clusters.**
- ❑ Mejor cuanto más alto: buscamos baja distancia máxima entre objetos del mismo clúster y elevada distancia entre clusters.



H.I. Rhys. *Machine Learning with R, the tidyverse, and mlr*. Manning, 2020.

$$\text{Dunn} = \frac{\min_{1 \leq i < j \leq N} \delta(c_i, c_j)}{\max_{1 \leq k \leq N} \Delta(c_k)}$$

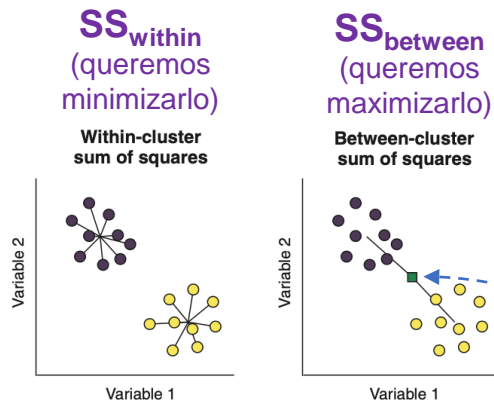
$\Delta(c_k)$  mide el diámetro del cluster  $k$ . Puede calcularse como la distancia máxima entre los objetos del cluster.

$\delta(c_i, c_j)$  es la distancia entre cada par de clusters  $c_i, c_j$

# Medidas de calidad del clustering

## Estadística Pseudo F (Calinski–Harabasz index)

- ❑ **Ratio entre las distancias “entre-clusters” (cómo de bien están separados los clusters entre ellos) y la distancia “intra-cluster” (cómo de cohesionados están los clusters internamente).**
- ❑ Mejor cuanto más alto.



H.I. Rhys. *Machine Learning with R, the tidyverse, and mlr*. Manning, 2020.

$$\text{Pseudo F} = \frac{SS_{\text{between}} / (k - 1)}{SS_{\text{within}} / (n - k)}$$

$n$  es el número de observaciones,  $k$  es el número de clusters

where  $SS_{\text{between}}$  and  $SS_{\text{within}}$  are calculated as

$$SS_{\text{between}} = \sum_k^N n_k (c_k - c_g)^2$$

$$SS_{\text{within}} = \sum_k^N \sum_{i \in k}^{n_k} (x_i - c_k)^2$$

$N$  es el número de clusters

$n_k$  es el número de objetos en el cluster  $k$

$c_k$  es el centroide de  $k$

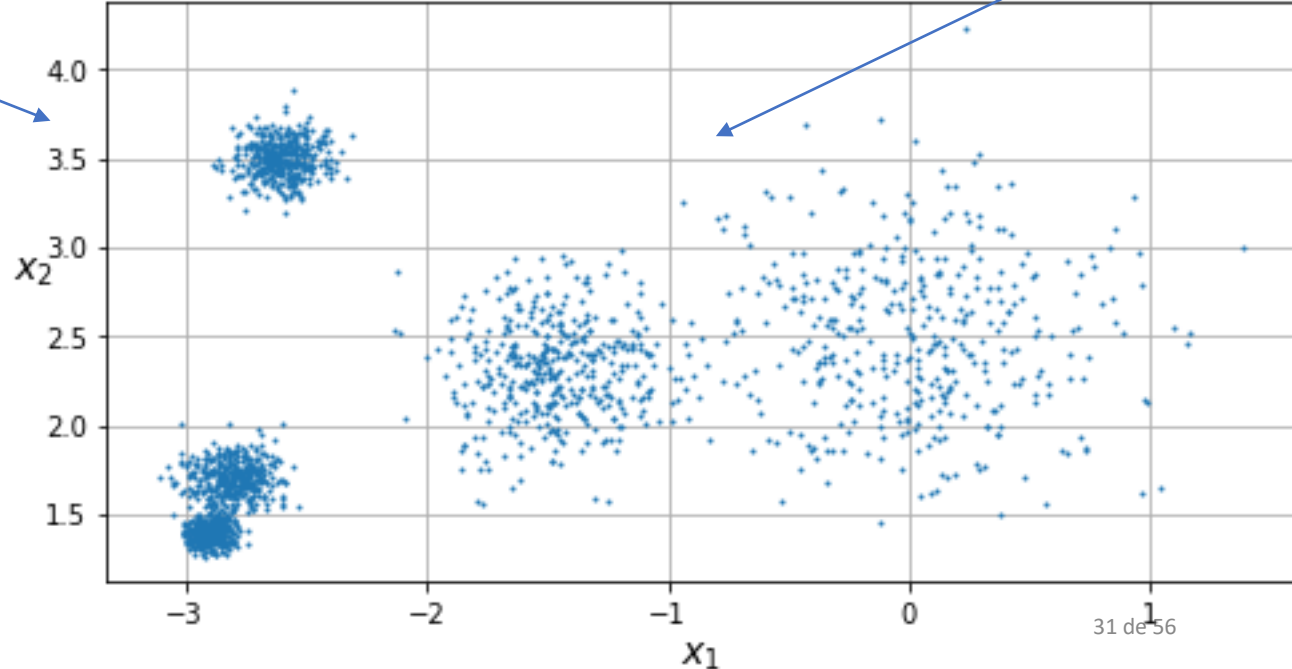
$c_g$  es el super-centroide (centroide global de los datos)

# Ejercicio 1

- Apartado A: Optimizando k-means (3 puntos)

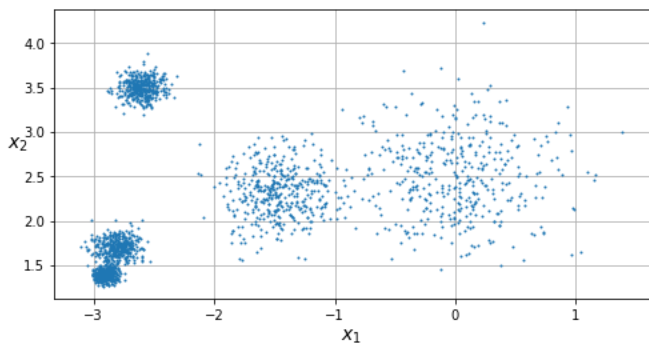
Generamos un conjunto sintético con 5 manchas y diferente densidad. Dos de ellas están un poco solpadas

Emplearemos el algoritmo k-means (`kMeans()`) con distintas configuraciones para optimizar el número de clusters con inercia y/o medida silueta.



# Ejercicio 1

- Apartado A: Optimizando k-means (3 puntos)



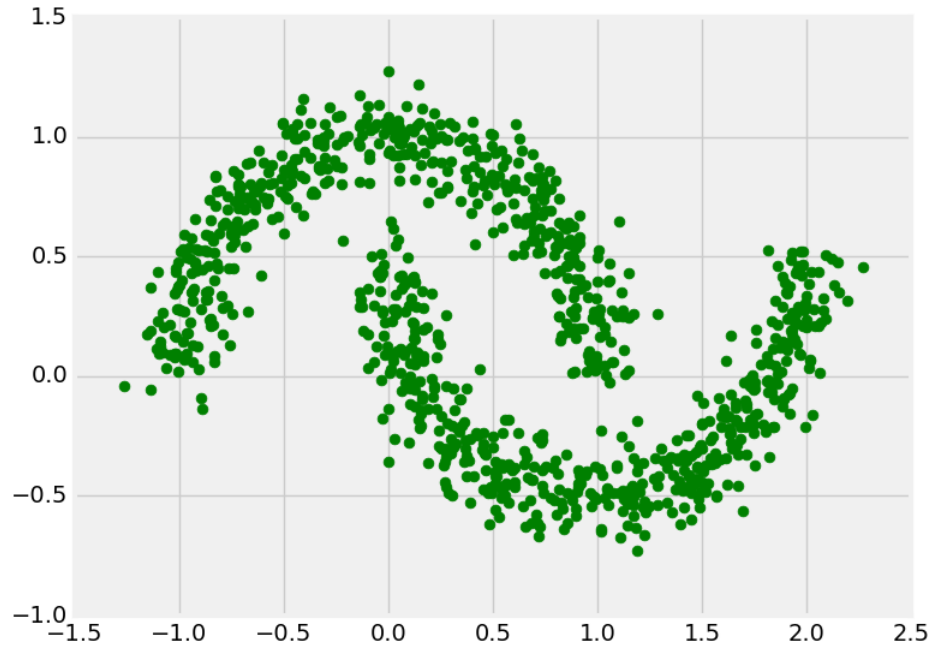
Tenéis que:

- Experimentar con los hiperparámetros `n_init`, `init` y `algorithm`.
- Asignar nuevos ejemplos a clusters obtenidos.
- Dibujar los bordes de decisión de los clusters obtenidos.
- Inicializar k-means a mano, poniendo centros definidos por el usuario.
- Calcular y representar gráficamente inercias y puntuación de siluetas para decidir qué agrupamiento es el mejor de acuerdo al número de clusters necesario.
- Utilizar k-means sobre un problema de imágenes pequeño y ver los resultados de diferentes configuraciones.



# Ejercicio 1

- Apartado B: utilización de **DBSCAN** (1 punto)



Generamos un conjunto sintético de 1000 puntos 2D con la función `make_moons`. Este será el conjunto de datos a usar para DBSCAN.

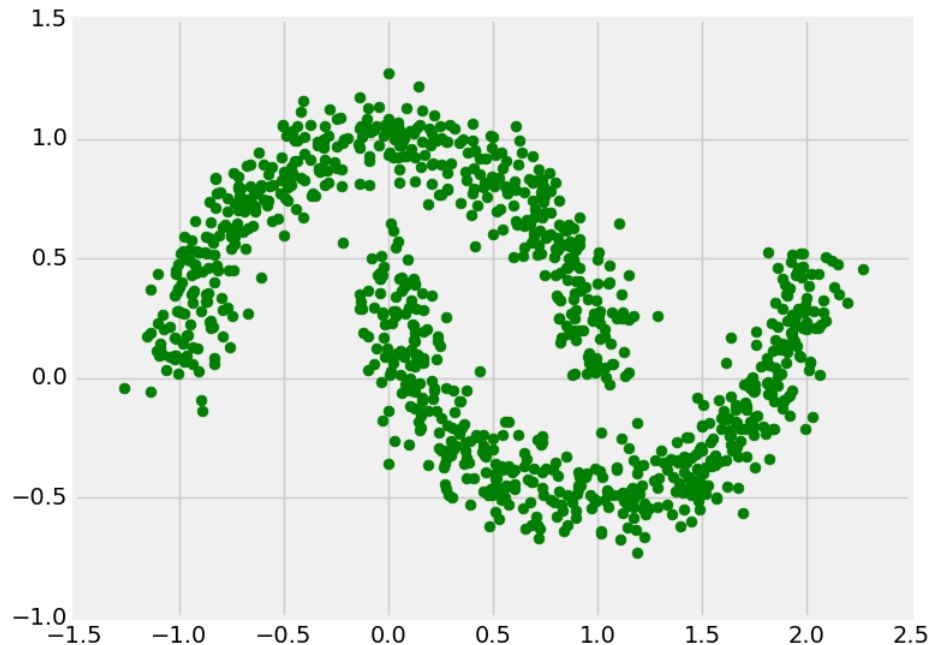
# Ejercicio 1

- Apartado B: utilización de DBSCAN (1 punto)



Tenéis que:

- Experimentar con el hiperparámetro  $\epsilon$ .
- Asignar nuevos ejemplos a clusters obtenidos.
- Dibujar los clusters obtenidos por varias configuraciones.
- Ejecutar un kNN partiendo de los puntos core y dibujar las fronteras de decisión obtenidas.



# Ejercicio 2

Datos tabulares:

– Problema de clustering

<https://archive.ics.uci.edu/dataset/406/anuran+calls+mfccs>

7195 ejemplos y 22 atributos, y 60 clases *ground truth*



# Detección de anomalías en Machine Learning

---

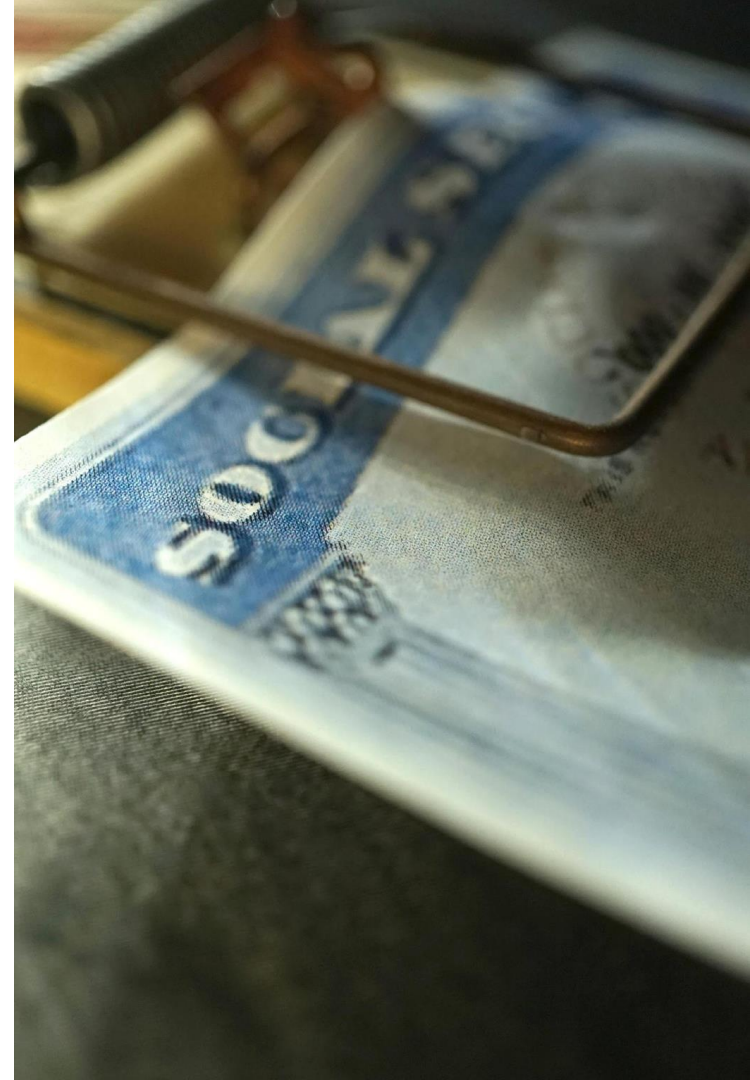
La detección de anomalías es la **identificación de patrones raros o no comunes en los datos**. Las anomalías pueden indicar fallos en los dispositivos, fraudes, errores humanos, entre otros.

Es una tarea importante en diferentes aplicaciones de *machine learning*.



# Aplicaciones de la detección de anomalías

- Detección de **fraude** en tarjetas de crédito
- Identificación de **fallos** en dispositivos electrónicos
- Detección de **intrusiones** en sistemas de seguridad
- Identificación de **enfermedades raras** en el campo de la medicina



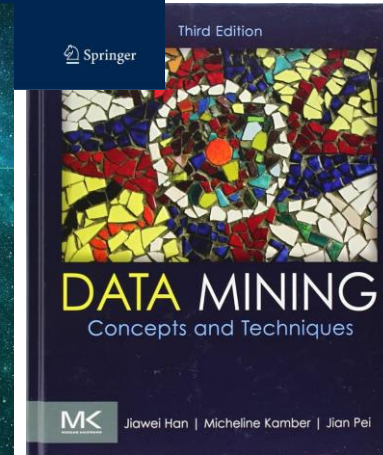
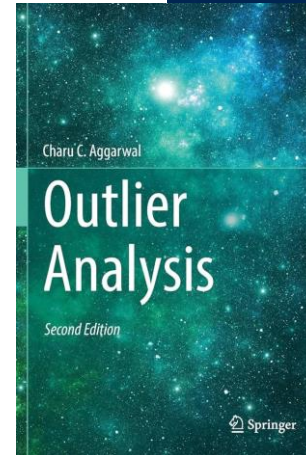
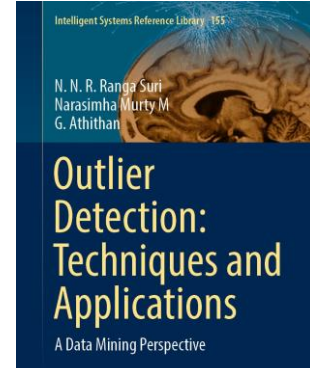
# TÉCNICAS DE DETECCIÓN DE ANOMALÍAS

- Métodos basados en **distancias** (k-NN)
- Métodos basados en **densidad** (LOF)
- Métodos basados en **modelos** (OCSVM)
- Métodos basados en ***ensembles*** (IsolationForest)

# Detección de anomalías

## Key Challenges

- Definir una región normal representativa es un reto
- El límite entre el comportamiento normal y el atípico no suele ser preciso
- Además, lo que se considera comportamiento normal evoluciona y cambia
- El concepto exacto de valor atípico varía también según el ámbito de aplicación
- Los datos pueden contener ruido 🗨️



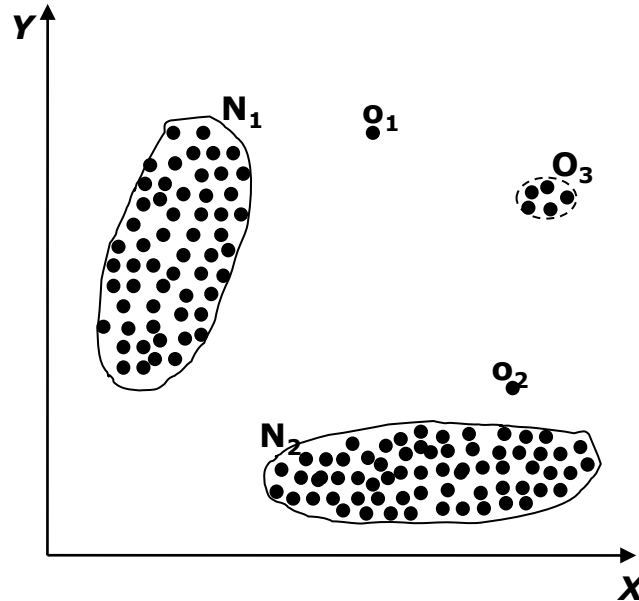


# Detección de anomalías

¿Qué son las anomalías?

Anomalías puntuales 

- Una instancia individual es anómala con respecto al resto de datos



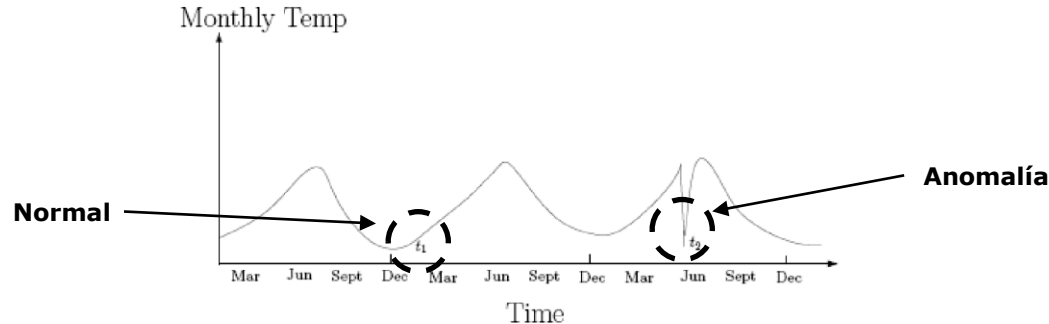


# Detección de anomalías

## ¿Qué son las anomalías?

### Anomalías contextuales

- Un dato individual es **anómalo dentro de un contexto**
- Requiere una noción de contexto
- También se denominan anomalías condicionales

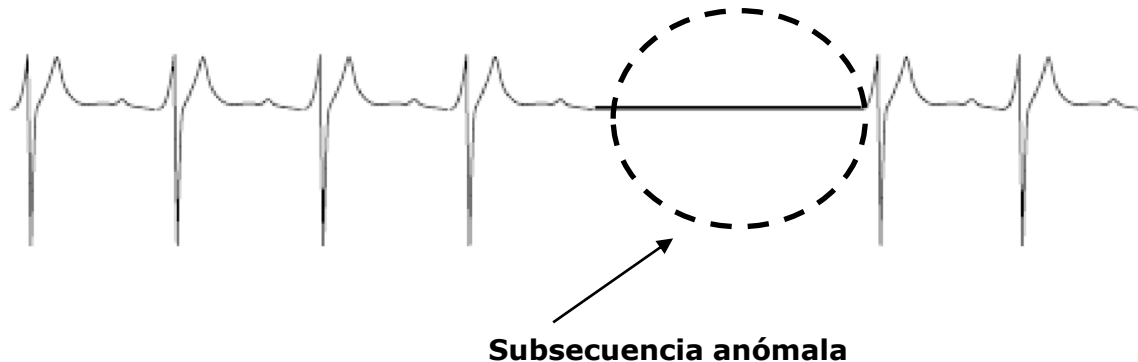


# Detección de anomalías

¿Qué son las anomalías?

**Anomalías colectivas** 

- Una **colección de instancias relacionadas es anómala**
- **Requiere una relación** (secuencial, espacial) entre instancias de datos
- Las instancias individuales dentro de una anomalía colectiva no son anómalas por sí mismas



# Detección de anomalías

Técnicas basadas en el vecino más próximo

- ***Suposición clave:*** los puntos normales tienen vecinos cercanos, mientras que las anomalías se sitúan lejos de otros puntos
- Enfoque general en dos pasos
  1. Calcula la vecindad de cada instancia/ejemplo
  2. Analiza la vecindad para determinar si la instancia es anómala o no

# Detección de anomalías

## Técnicas basadas en el vecino más próximo

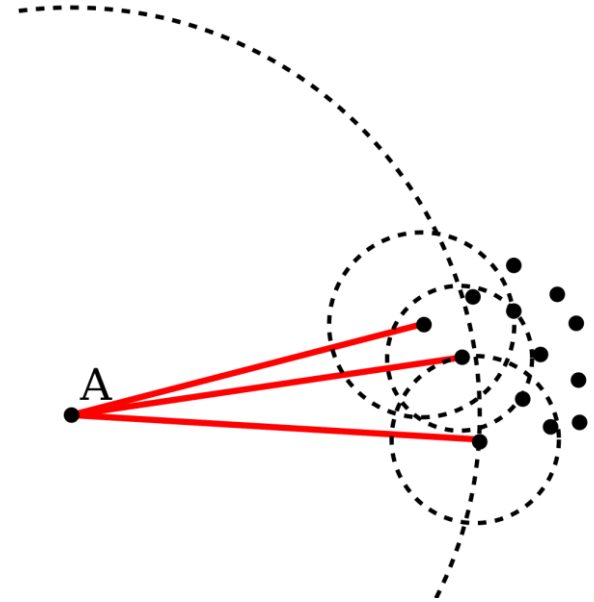
- *Enfoque del vecino más próximo (NN)*
  - Para cada punto de datos  $d$ , calcula la distancia al  $k$ -ésimo vecino más próximo  $d_k$
  - Ordena todos los puntos de datos según la distancia  $d_k$
  - **Los valores atípicos son los puntos que tienen mayor distancia  $d_k$  y, por tanto, están situados en los vecindarios más dispersos**
  - Normalmente, **los puntos con una distancia  $d_k$  superior al  $n\%$  se identifican como valores atípicos**
    - $n, k$  – hiperparámetros
  - No es adecuado para conjuntos de datos que tienen modos con densidad variable

# Detección de anomalías

## Técnicas basadas en densidad

### Local Outlier Factor (LOF)

- El algoritmo LOF comparte conceptos e intuiciones con DBSCAN, y **compara la densidad local de una instancia con la de sus vecinos** para detectar valores atípicos.
- Si la **densidad de una instancia es significativamente menor que la de sus vecinos**, se considera un **valor atípico**.
- <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html>

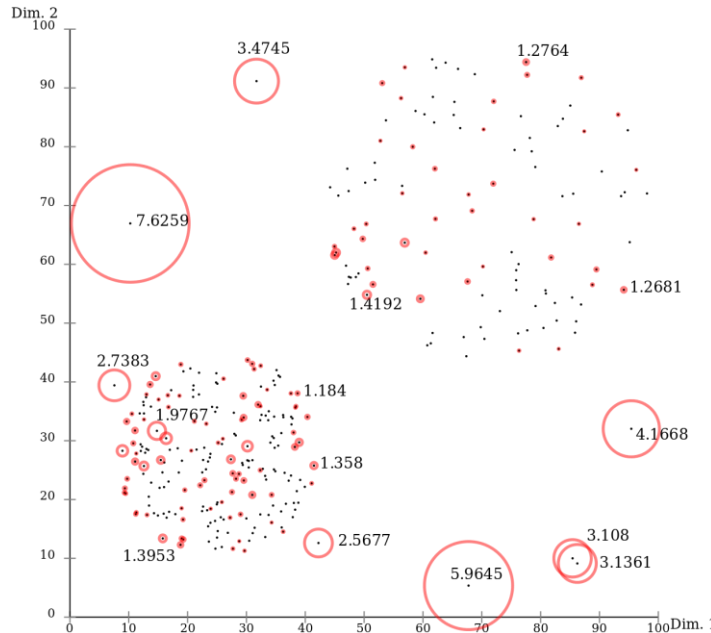


Idea clave de LOF: comparar la densidad local de un punto con la densidad de sus vecinos. En el ejemplo, **A tiene una densidad mucho más baja que sus vecinos**, y operamos con una  $k\text{-distance}()$  con  $k=3$ .  $k\text{-distance}(A)$ : distancia de la instancia A al  $k$  vecino más cercano. [Fuente](#).

# Detección de anomalías

## Técnicas basadas en densidad

### Local Outlier Factor (LOF)



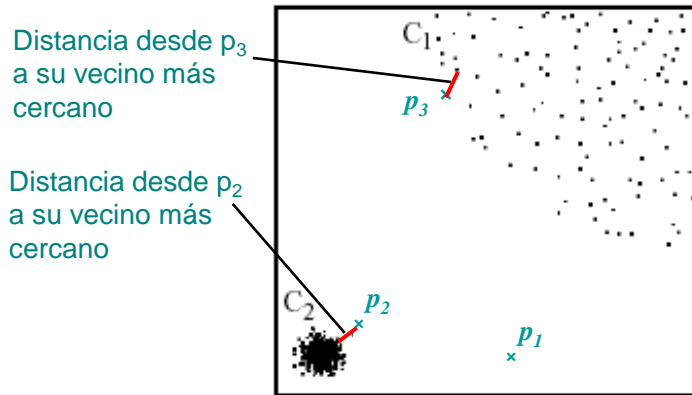
Los scores más elevados se corresponden con puntos más aislados, habiendo dos clusters principales (uno arriba a la derecha, y otro abajo a la izquierda).

[Fuente.](#)

# Detección de anomalías

## Técnicas basadas en densidad

### Ventajas de las técnicas basadas en la densidad



Al no tener en cuenta distancias, **un punto a una "pequeña" distancia de un cluster muy denso puede ser considerado un outlier.**

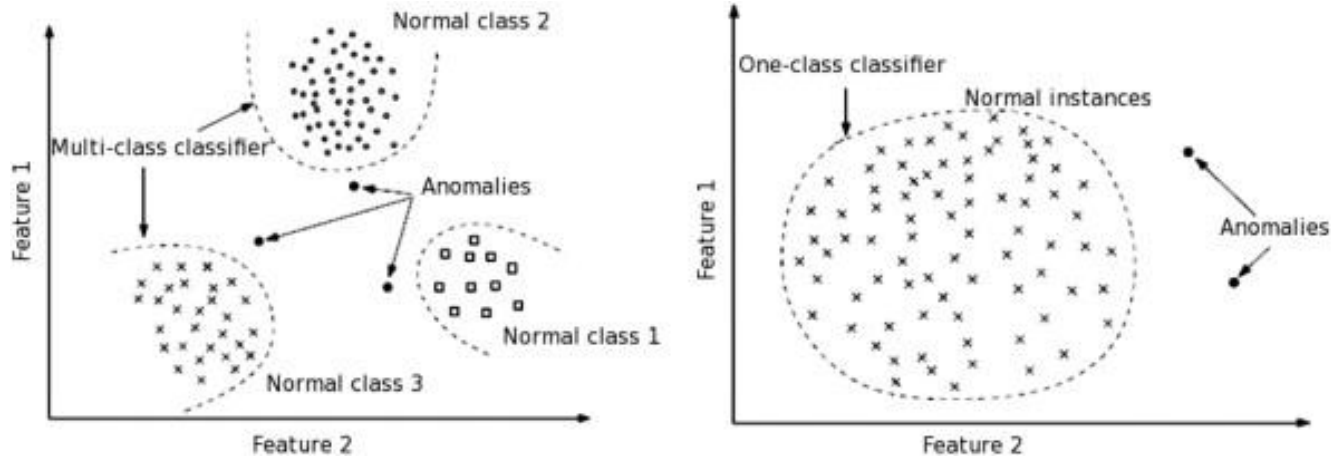
En el **enfoque NN**,  $p_2$  **no se considera un valor atípico**, mientras que el enfoque **LOF** **considera que tanto  $p_1$  como  $p_2$  son valores atípicos.**

El enfoque **NN** **puede llegar a considerar  $p_3$  como valor atípico**, pero el enfoque **LOF** **no lo hace.**

# Detección de anomalías

## One-class Classification

### Several classes vs One-class classification



Schölkopf, Bernhard, et al. "Support vector method for novelty detection." *Advances in neural information processing systems* 12 (1999).  
Schölkopf, Bernhard, et al. "Estimating the support of a high-dimensional distribution." *Neural computation* 13.7 (2001): 1443-1471.  
Tax and Duin. "Support vector data description." *Machine learning* 54 (2004): 45-66.  
<https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html>



# Detección de anomalías

## One-class Classification

- One-Class SVM (OCSVM)
  - Encuentra el **hiperplano óptimo** para **separar la clase objetivo del origen con el máximo margen**
  - Utiliza la **hiperesfera mínima** para **encerrar la clase objetivo**

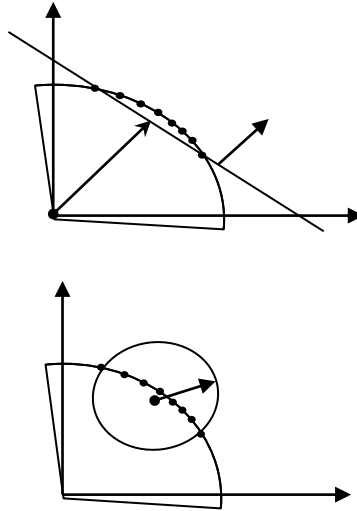
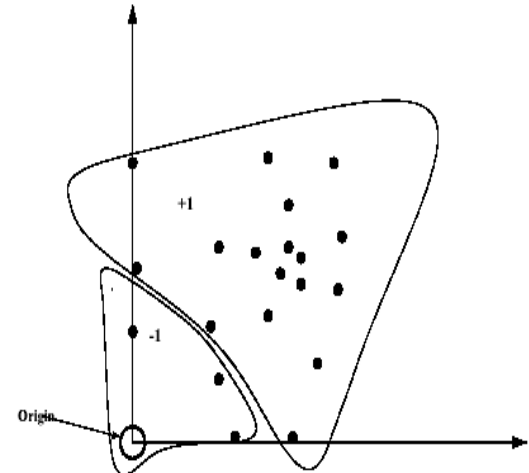



Figure 1: One-class SVM



One-Class SVM Classifier. The origin is the only original member of the second class.

En lugar de un hiperplano que separa dos clases maximizando el margen, intenta  **minimizar el radio de una hiperesfera que incluya la mayor parte de los ejemplos** (de ahí lo de *one-class*, dado que explícitamente solo trabajamos con una). Scikit-learn indica que es un método de “[unsupervised outlier detection](#)”. Tú no usas etiquetas durante el *fit*, solo le pasas los datos de entrenamiento (sin salidas deseadas).

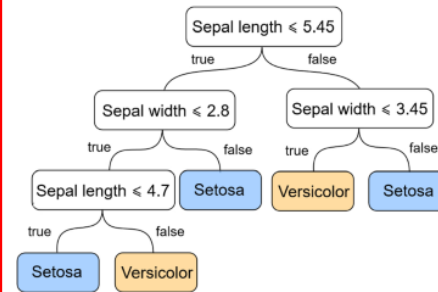
# Detección de anomalías

## Técnicas basadas en ensembles

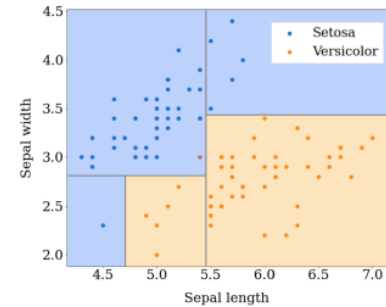
### Algoritmo Isolation Forest

- Funciona particionando recursivamente el conjunto de datos utilizando árboles de aislamiento.
- Se basa en que **las anomalías son más fáciles de aislar y tienen caminos más cortos en los árboles.**
- Es un método que no estima la densidad y funciona bien con grandes volúmenes de datos.
- <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html>

#### Recordatorio sobre árboles de decisión (DTs):



(a) Tree visualization



(b) Partitioning visualization

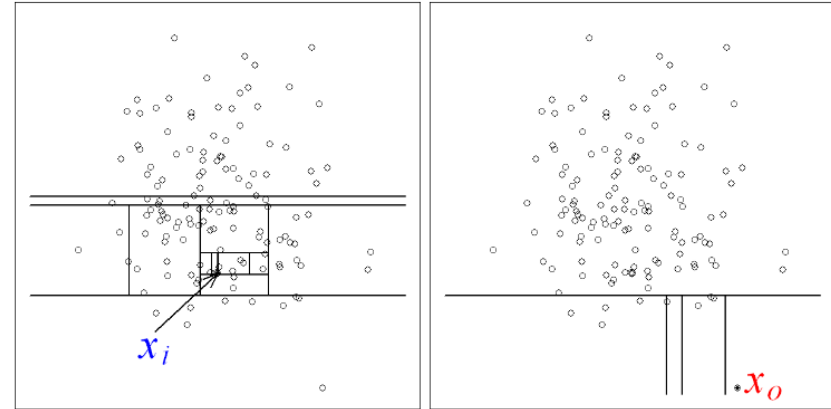
Los DTs realizan una partición no lineal del espacio de características. [Fuente](#).

# Detección de anomalías

## Técnicas basadas en ensembles

### Algoritmo Isolation Forest

- Funciona particionando recursivamente el conjunto de datos utilizando árboles de aislamiento.
- Se basa en que **las anomalías son más fáciles de aislar y tienen caminos más cortos en los árboles.**
- Es un método que no estima la densidad y funciona bien con grandes volúmenes de datos.
- <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html>



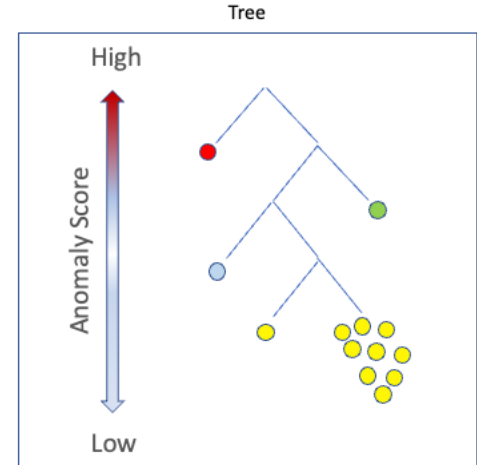
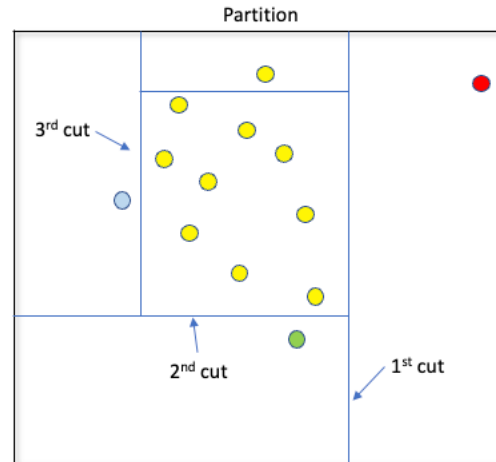
Las anomalías son más susceptibles al aislamiento. Un punto normal (como  $x_i$ ), en este ejemplo, ha requerido 12 particiones aleatorias para ser aislado. En cambio, una anomalía, como  $x_0$ , requiere solo 4 particiones para ser aislada.

# Detección de anomalías

## Algoritmo Isolation Forest



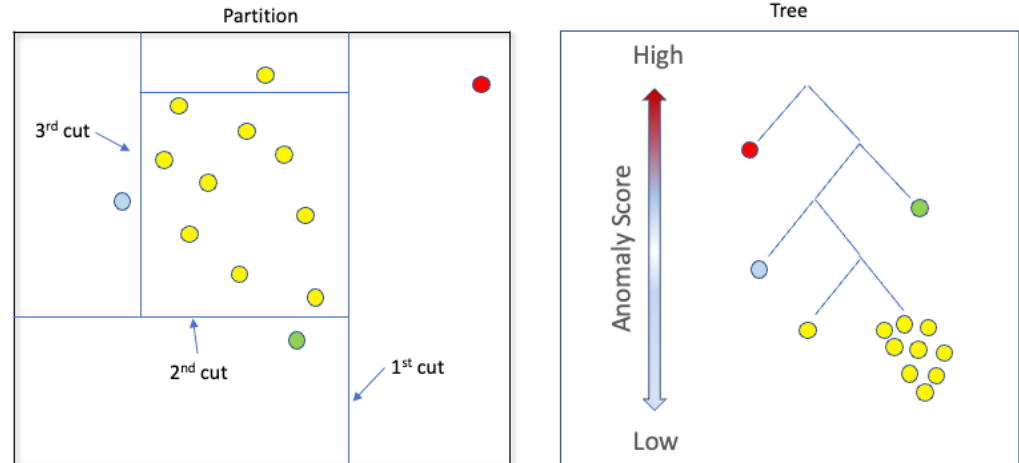
1. **Construcción de árboles de aislamiento:** El algoritmo crea **múltiples árboles de aislamiento** (*Isolation Trees* o *iTrees*) para la construcción del bosque. **Para cada árbol, se selecciona de manera aleatoria un subconjunto de los datos junto con un subconjunto de las características.**
2. **Particionamiento de datos:** En cada nodo del árbol, **se elige aleatoriamente una característica y un valor de división** (por eso es un método muy rápido). **Los datos se dividen en dos nodos hijos según si son menores o mayores que el valor de división.** Este proceso se repite hasta que el *datapoint* está aislado o se alcanza la profundidad máxima del árbol.



# Detección de anomalías

## Algoritmo Isolation Forest

- Cálculo del puntaje de anomalía:** El score de anomalía de un punto se basa en la profundidad promedio a la que se aísla en los diferentes árboles. Un score más alto indica una mayor probabilidad de que el punto de datos sea una anomalía.
- Evaluación de resultados:** Finalmente, los puntos con puntajes de anomalía que superan un umbral definido se consideran anomalías.



# Ejercicio 3

Datos tabulares:

Se proporciona un ejemplo del uso de IsolationForest en un conjunto sintético.



– Problema de detección de anomalías

<https://odds.cs.stonybrook.edu/satellite-dataset/>

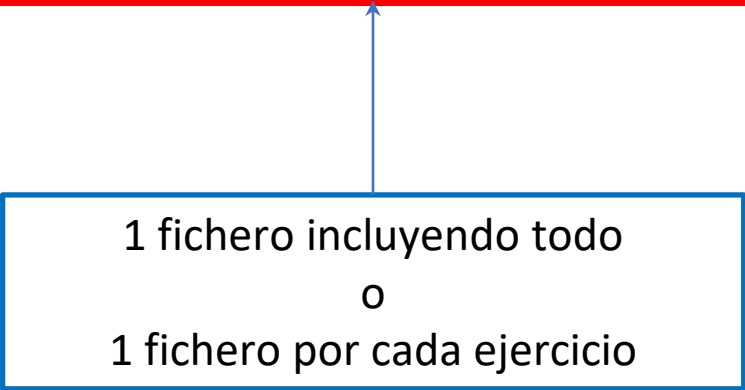


6435 ejemplos, 36 atributos.

# Entrega

.ipynb = Código, informe y resultados integrados en un Colab notebook

1 fichero incluyendo todo  
o  
1 fichero por cada ejercicio

A blue rectangular box containing the text '1 fichero incluyendo todo', 'o', and '1 fichero por cada ejercicio'. A blue arrow points upwards from the top center of this box to the bottom center of the red box above it.

Subid la entrega a PRADO, a la actividad creada para ello.

Fecha de entrega: 19 de Mayo

# Referencias online recomendadas

- <https://scikit-learn.org/stable/modules/clustering.html>
- [https://scikit-learn.org/stable/modules/outlier\\_detection.html](https://scikit-learn.org/stable/modules/outlier_detection.html)
- <https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/>



# Prácticas de Aprendizaje Automático

## Práctica 2

### Experimentación con agrupamiento y detección de anomalías no supervisada

Pablo Mesejo y Salvador García

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial



UNIVERSIDAD  
DE GRANADA

