Guía de estudio - Aprendizaje No Supervisado

 ¡Hola! Te damos la bienvenida a esta nueva guía de estudio.

¿En qué consiste esta guía?

En esta guía vamos a adentrarnos en los algoritmos de machine learning, específicamente en el aprendizaje no supervisado y en los algoritmos de clustering dentro de este campo.

Explicaremos en detalle los algoritmos de clustering como técnica de aprendizaje no supervisado. Nos centraremos en diferentes algoritmos de clustering, su implementación práctica utilizando la biblioteca Scikit-learn (sklearn.cluster) en Python, y discutiremos las ventajas y desventajas de cada algoritmo. Además, abordaremos la validación de los resultados de clustering, tanto desde una perspectiva cuantitativa utilizando criterios de calidad numéricos como desde una perspectiva cualitativa basada en el juicio de expertos, entendiendo porque cada una de estas validaciones son importantes para determinar el éxito de un análisis o proyecto utilizando estás técnicas.

A lo largo de este documento, utilizaremos ejemplos reales en diferentes dominios de aplicación para ilustrar la aplicación práctica de los algoritmos de clustering. Estos ejemplos nos permitirán comprender cómo implementar y ajustar los algoritmos de clustering, evaluar la calidad de los resultados y extraer información valiosa de los datos no etiquetados.

En resumen, este documento tiene como objetivo proporcionar una guía completa para comprender, implementar y validar algoritmos de clustering en el contexto del aprendizaje no supervisado. Esperamos que este material sea de utilidad tanto para profesionales como para estudiantes que deseen explorar y aprovechar al máximo el poder del clustering en sus propias aplicaciones de análisis de datos.

¡Vamos con todo!

Tabla de contenidos

¡Comencemos!

Aprendizaje No Supervisado

El aprendizaje no supervisado es una rama fundamental en el campo del análisis de datos y la inteligencia artificial. A diferencia del aprendizaje supervisado, donde se dispone de datos etiquetados con información de entrada y salida, en el aprendizaje no supervisado se trabaja con datos sin etiquetar y el objetivo principal es descubrir información útil y relevante a partir de estos datos. Esto hace que los set de datos para entrenar estos modelos sean menos costosos, ya que no necesitan una respuesta al problema.

El aprendizaje no supervisado tiene muchas aplicaciones en una variedad de campos. Algunas de las aplicaciones más destacadas incluyen:

1. **Minería de Datos:** El aprendizaje no supervisado permite descubrir patrones y tendencias ocultas en grandes volúmenes de datos, lo que ayuda en la toma de decisiones y la generación de conocimiento útil.
2. **Análisis de Imágenes:** La reducción de dimensionalidad y el clustering se utilizan para la clasificación automática de imágenes, la segmentación de objetos y la extracción de características relevantes.
3. **Bioinformática:** El aprendizaje no supervisado se aplica en el análisis de secuencias de ADN, la clasificación de proteínas y la identificación de patrones en datos biológicos.
4. **Reconocimiento de Patrones:** El aprendizaje no supervisado se utiliza en la detección de fraudes, la detección de intrusiones en sistemas de seguridad y el reconocimiento de patrones en datos no estructurados.

El aprendizaje no supervisado abarca diversas tareas, entre las cuales dos de las más destacadas son la reducción de dimensionalidad y el clustering

Reducción de dimensionalidad

La reducción de dimensionalidad se refiere a la técnica utilizada para disminuir la cantidad de variables o características en un conjunto de datos. Esto se logra al proyectar los datos de alta dimensionalidad en un espacio de menor dimensión mientras se conserva la mayor cantidad de información relevante posible. La reducción de dimensionalidad es útil cuando se trabaja con conjuntos de datos con un gran número de características, ya que puede ayudar a eliminar el ruido, mejorar la eficiencia computacional y facilitar la visualización de los datos. Algunos algoritmos populares de reducción de dimensionalidad incluyen el Análisis de Componentes Principales (PCA) y el Análisis de Discriminante Lineal (LDA).

Clustering

El clustering se enfoca en agrupar datos similares en conjuntos o clústeres, donde los elementos dentro de cada clúster son más similares entre sí que con los elementos de otros clústeres. El objetivo del clustering es descubrir la estructura inherente en los datos y

obtener información sobre grupos o segmentos en los conjuntos de datos no etiquetados. El clustering es ampliamente utilizado en diversas aplicaciones, como la segmentación de clientes en marketing, la detección de anomalías en la seguridad de la red, la agrupación de documentos en minería de texto, entre otros.

Nos vamos a centrar en los algoritmos de clustering, explicando los diferentes algoritmos, sus ventajas y desventajas, las formas de de implementarlos y los mecanismos para validar la calidad de estos mismos.

Clustering, áreas y algoritmos

Clustering es una técnica de análisis de datos en el aprendizaje no supervisado que tiene como objetivo agrupar objetos similares en conjuntos llamados clusters. Un cluster es una colección o conjunto de objetos que comparten características similares entre sí y difieren de otros objetos en el conjunto de datos. En otras palabras, el clustering busca identificar grupos intrínsecos en los datos sin tener información previa sobre las categorías o etiquetas a las que pertenecen los objetos.

En el contexto del análisis de datos y el aprendizaje no supervisado, el término "cluster" se utiliza para describir conjuntos de objetos que han sido agrupados mediante técnicas de clustering, como K-means, Fuzzy CMean, DBSCAN o clustering jerárquico. Los clusters se crean mediante algoritmos que buscan maximizar la similitud intra-cluster y minimizar la similitud inter-cluster, lo que permite agrupar objetos similares y separar objetos diferentes.

A continuación se muestran algunos ejemplos en diferentes industrias:

Industria	Ejemplo
Minería	**Clustering de datos geológicos:** en la industria minera, se puede aplicar clustering para identificar áreas geográficas con características similares en términos de contenido de minerales, estructuras geológicas o composición del suelo. Esto puede ayudar a identificar ubicaciones potenciales para la exploración de minerales.
Minería	**Clustering de patrones de perforación:** en la exploración y producción de minas, se pueden utilizar técnicas de clustering para agrupar patrones de perforación con características similares. Esto puede ayudar a identificar patrones geológicos o anomalías en las muestras de perforación y mejorar la eficiencia en la toma de decisiones.
Retail	**Clustering de clientes:** en la industria minorista, el clustering se utiliza para agrupar clientes según su comportamiento de compra, preferencias de productos o características demográficas. Esto permite a las empresas personalizar ofertas, estrategias de marketing y

	programas de fidelización para diferentes segmentos de clientes.
Retail	**Clustering de productos:** en el ámbito del retail, se puede aplicar clustering para agrupar productos según sus características, como categoría, precio, popularidad o características técnicas. Esto ayuda en la gestión de inventario, recomendaciones de productos y planificación de promociones.
Telecomunicaciones	**Clustering de patrones de uso de servicios:** en el sector de las telecomunicaciones, el clustering se utiliza para agrupar usuarios según sus patrones de uso de servicios, como llamadas, mensajes o uso de datos. Esto puede ayudar a identificar segmentos de usuarios con necesidades similares y ofrecer planes de servicios personalizados.
Telecomunicaciones	**Clustering de torres celulares:** en la infraestructura de telecomunicaciones, se puede aplicar clustering para agrupar torres celulares según su ubicación, carga de tráfico o características técnicas. Esto ayuda en la planificación de la red, optimización de la cobertura y distribución eficiente de recursos.
Salud	**Clustering de perfiles de pacientes:** en el campo de la salud, se puede aplicar clustering para agrupar perfiles de pacientes según características clínicas, historial médico o resultados de pruebas. Esto puede ayudar a identificar grupos de pacientes con condiciones similares, apoyar la toma de decisiones médicas y personalizar el tratamiento.
Salud	**Clustering de genes o proteínas:** en la investigación genética o biomédica, se pueden aplicar técnicas de clustering para agrupar genes o proteínas según su expresión, funciones o interacciones. Esto ayuda en la identificación de patrones genéticos, clasificación de enfermedades y descubrimiento de nuevos objetivos terapéuticos.
Banca	**Segmentación de clientes:** los bancos pueden utilizar técnicas de clustering para segmentar a sus clientes en grupos homogéneos con base en características demográficas, comportamiento financiero o preferencias de productos y servicios. Esto permite personalizar estrategias de marketing, productos y servicios para cada segmento, así como desarrollar campañas de fidelización específicas.
Banca	**Detección de fraude:** el clustering se puede utilizar en el análisis de transacciones bancarias para detectar patrones de comportamiento anómalos y sospechosos

	que puedan indicar actividades fraudulentas. Al agrupar transacciones similares, se pueden identificar grupos que difieran significativamente del comportamiento típico, lo que ayuda a detectar y prevenir fraudes.
Educación	**Agrupación de estudiantes:** en el ámbito educativo, se pueden aplicar técnicas de clustering para agrupar estudiantes en función de sus características académicas, intereses, estilos de aprendizaje o rendimiento en exámenes. Esto puede ayudar a los educadores a personalizar el enfoque educativo, identificar necesidades específicas y ofrecer apoyo individualizado a los estudiantes.
Educación	**Análisis de rendimiento de escuelas:** las técnicas de clustering se pueden utilizar para agrupar escuelas o instituciones educativas en función de su rendimiento académico, recursos disponibles, características demográficas de los estudiantes o indicadores socioeconómicos. Esto proporciona información valiosa para comparar y evaluar el desempeño de las escuelas, así como para identificar áreas de mejora y diseñar políticas educativas efectivas.

Tabla 1 . Ejemplos de clustering en diferentes industrias

Diagrama de Voronoi

Un diagrama de Voronoi es una representación gráfica de cómo un espacio se divide en regiones basadas en la proximidad a un conjunto de puntos de referencia. Cada región del diagrama de Voronoi está asociada a uno de los puntos de referencia y contiene todos los puntos del espacio que están más cerca de ese punto en particular que de cualquier otro punto de referencia.

En el contexto del clustering o análisis espacial, el diagrama de Voronoi es útil para visualizar cómo los puntos se agrupan alrededor de los centroides o puntos representativos de un conjunto de datos. Cada región del diagrama de Voronoi representa un cluster y muestra las áreas del espacio que son más cercanas a su respectivo centroide.

Para crear un diagrama de Voronoi se aplican los siguientes pasos:

1. Seleccionar un conjunto de puntos de referencia (por ejemplo, los centroides de los clusters) que actuarán como centros de las regiones.
2. Calcular las distancias de todos los puntos del espacio a cada punto de referencia.
3. Asignar cada punto del espacio al punto de referencia más cercano, lo que crea regiones alrededor de cada punto de referencia.
4. Trazar las fronteras que separan las regiones adyacentes.

En el contexto del clustering, cada región del diagrama de Voronoi representa un cluster, y los puntos dentro de cada región están más cerca del centroide correspondiente que de cualquier otro centroide.

K - means

El algoritmo K-Means es uno de los métodos más utilizados en el campo del clustering, una técnica del aprendizaje no supervisado que busca agrupar conjuntos de datos similares en clústeres. El algoritmo K-Means es conocido por su simplicidad y eficiencia en la agrupación de datos, lo que lo hace ampliamente aplicable en diversas áreas, desde el análisis de datos hasta la segmentación de clientes

El algoritmo K-means es un enfoque iterativo que busca asignar puntos de datos a clusters mediante la minimización de la suma de las distancias al cuadrado. Es ampliamente utilizado debido a su simplicidad y eficiencia en grandes conjuntos de datos, aunque puede requerir ciertos ajustes para obtener resultados óptimos.

Algoritmo

A continuación, se describe el proceso general del algoritmo K-means:

1. **Paso de inicialización:**
    a. Selecciona el número K de clusters deseados.
    b. Inicializa los centroides de los K clusters de manera aleatoria o utilizando algún otro método, como el algoritmo de inicialización K-means++.
2. **Paso de asignación:** Asigna cada punto de datos al cluster cuyo centroide esté más cerca. Esto se basa en la distancia euclidiana u otras métricas de distancia.
3. **Paso de actualización:** Calcula los nuevos centroides de los clusters al encontrar la media de los puntos de datos asignados a cada cluster.
4. Repetición de los pasos 2 y 3:
    a. Repite los pasos de asignación y actualización hasta que se cumpla algún criterio de convergencia. Esto puede ser cuando no hay cambios significativos en la asignación de puntos a clusters o cuando se alcanza un número máximo de iteraciones.
5. **Resultados:**
    a. Los centroides finales representan los puntos centrales de los clusters.
    b. Los puntos de datos se agrupan en K clusters basados en la asignación final de los centroides.

Es importante tener en cuenta que el algoritmo K-means puede converger a mínimos locales subóptimos y puede ser sensible a la inicialización de los centroides. Para mitigar estos problemas, se pueden probar múltiples inicializaciones y ejecuciones del algoritmo o utilizar técnicas de evaluación para seleccionar la mejor solución.

Ventajas y desventajas

El algoritmo K-means tiene sus ventajas y desventajas. Aquí hay una descripción de algunas de ellas, junto con posibles soluciones a las desventajas:

Ventajas

1. Simplicidad: K-means es un algoritmo simple y fácil de implementar. No requiere una cantidad excesiva de configuración o parámetros.

2. Eficiencia: Es computacionalmente eficiente y puede manejar grandes conjuntos de datos.

3. Escalabilidad: Es adecuado para conjuntos de datos con un número significativo de dimensiones y características.

Desventajas

1. Sensibilidad a la inicialización: Los resultados del clustering pueden variar dependiendo de la inicialización de los centroides. Esto puede llevar a la convergencia en mínimos locales subóptimos.

   Solución: Una solución común es realizar múltiples inicializaciones aleatorias y ejecuciones del algoritmo y seleccionar la solución con la mejor función objetivo o utilizando medidas de evaluación del clustering.

2. Requiere especificar el número de clusters (K) de antemano: Es necesario conocer o estimar el número óptimo de clusters antes de aplicar el algoritmo.

   Solución: Se pueden utilizar métodos de selección de K, como la técnica del codo (elbow method) o el índice de silueta, para encontrar el número óptimo de clusters basado en medidas de calidad.

3. Sensible a los datos atípicos y ruido: Los puntos de datos anómalos o ruidosos pueden afectar la formación de clusters y la asignación incorrecta de puntos.

   Solución: Se pueden aplicar técnicas de preprocesamiento de datos, como la detección de outliers o el filtrado de ruido, para mitigar su impacto en el algoritmo K-means.

4. No adecuado para clusters de forma irregular o de tamaño muy diferente: K-means tiende a generar clusters de forma esférica y de tamaño similar.

   Solución: Para abordar este problema, se pueden utilizar variantes del algoritmo K-means, como el K-means en espacios de características más altas o técnicas de

clustering más avanzadas, como el DBSCAN (Density-Based Spatial Clustering of Applications with Noise), que puede detectar clusters de forma arbitraria y tamaño variable.

Mientras que el algoritmo K-means es simple y eficiente, tiene limitaciones en cuanto a la sensibilidad a la inicialización y la necesidad de especificar el número de clusters de antemano. Sin embargo, muchas de estas desventajas pueden mitigarse mediante el uso de estrategias apropiadas de inicialización, selección de K y aplicando técnicas de preprocesamiento de datos adecuadas.

Método del Codo

El método del codo (elbow method) es una técnica comúnmente utilizada para determinar el número óptimo de clusters (K) en el algoritmo de clustering. La idea principal detrás del método del codo es identificar el valor de K donde se produce un cambio significativo en la variabilidad explicada por los clusters.

El procedimiento general del método del codo es el siguiente:

1. Ejecutar el algoritmo K-means para un rango de valores de K.
2. Calcular la suma de las distancias al cuadrado dentro de los clusters, para cada valor de K. Esta métrica mide la variabilidad dentro de los clusters.
3. Graficar el valor de K en el eje x y la métrica intra cluster en el eje y.
4. Observar el gráfico y buscar el punto donde se produce un "codo" pronunciado o una disminución significativa en la métrica intracluster. Este punto indica el número óptimo de clusters según el método del codo.

Fuzzy C-Means (FCM)

El algoritmo Fuzzy C-Means (FCM) es una técnica de clustering que se basa en la teoría de conjuntos difusos. A diferencia del algoritmo K-Means, que asigna puntos de datos a clústeres de manera binaria, el FCM asigna grados de membresía difusos a cada punto, lo que permite una mayor flexibilidad y representación de la incertidumbre en los datos. El FCM es ampliamente utilizado en aplicaciones donde los puntos de datos pueden pertenecer a múltiples clústeres simultáneamente.

Es importante tener en cuenta que el algoritmo FCM requiere la especificación de un parámetro de "fuzziness" que controla la difusión o borrosidad de las asignaciones de membresía. Valores más altos de este parámetro hacen que las asignaciones sean más difusas, permitiendo que los puntos de datos tengan grados de pertenencia más equilibrados entre los clusters.

El algoritmo FCM se puede implementar utilizando bibliotecas de Python como scikit-learn o la librería fuzzy-c-means. Estas implementaciones proporcionan funciones convenientes

para calcular los centroides, actualizar los grados de pertenencia y obtener los resultados finales del clustering difuso.

Algoritmo

A continuación, se presenta una descripción general del algoritmo Fuzzy C-means:

1. **Inicialización**: Se selecciona el número de clusters (K) y se inicializan aleatoriamente los centroides y los grados de pertenencia para cada punto de datos. Los grados de pertenencia son valores entre 0 y 1 que representan la membresía relativa de un punto a cada clúster.

2. **Cálculo de los centroides:** Se calculan los centroides ponderados utilizando los grados de pertenencia. Cada punto contribuye al cálculo del centroide de cada cluster según su grado de pertenencia.

3. **Actualización de los grados de pertenencia:** Se actualizan los grados de pertenencia de cada punto de datos utilizando una función de pertenencia basada en la distancia a los centroides. Cuanto más cerca esté un punto de un centroide en comparación con los otros centroides, mayor será su grado de pertenencia a ese cluster.
   La fórmula de actualización de grados de pertenencia es la siguiente:

$$u_{ij} = \left(\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{ik}} \right)^{\frac{2}{m-1}} \right)^{-1}$$

   donde:
   u_{ij} : Es pertenencia de la muestra i al cluster j
   d_{ij} : Distancia de la muestra i al cluster j
   m: Parámetro de "fuzziness"

4. **Iteración**: Se repiten los pasos 2 y 3 hasta que se alcance un criterio de convergencia. El criterio puede ser un número máximo de iteraciones, una pequeña mejora en los valores de los centroides o un cambio mínimo en los grados de pertenencia.

5. **Resultados**: Al finalizar las iteraciones, se obtienen los centroides finales y los grados de pertenencia de los puntos de datos. Estos resultados proporcionan información sobre la membresía relativa de cada punto a cada cluster.

Cuando se utiliza FCM, el parámetro de fuzziness se elige de antemano y se mantiene constante durante la ejecución del algoritmo. Un valor típico para m es 2, lo que indica que cada punto de datos puede pertenecer completamente a un solo cluster. A medida que el valor de m aumenta, los grados de pertenencia se vuelven más difusos, permitiendo que los puntos de datos tengan asignaciones parciales a varios clusters.

La interpretación del parámetro de fuzziness (m) es la siguiente:

- **Para m = 1:** La pertenencia es binaria, es decir, un punto de datos pertenece completamente a un cluster o no pertenece a él en absoluto (asignación crisp). En este caso, FCM se reduce a un algoritmo de clustering duro como K-means.
- **Para m > 1:** La pertenencia es difusa, lo que significa que los puntos de datos pueden tener grados de pertenencia parciales a múltiples clusters. Cuanto mayor sea el valor de m, más difusas serán las asignaciones. Esto permite que los puntos de datos se consideren parcialmente miembros de varios clusters, lo que refleja la incertidumbre en la pertenencia.

La elección del valor de m depende del dominio del problema y de la naturaleza de los datos. Un valor bajo de m (cercano a 1) produce particiones más nítidas y más cercanas a un clustering duro, mientras que un valor alto de m (por ejemplo, 2 o más) permite asignaciones más difusas y flexibles.

Es importante destacar que el valor de m no se puede determinar de manera objetiva y no existe un valor óptimo universal. En su lugar, se recomienda experimentar con diferentes valores de m y evaluar el rendimiento del clustering utilizando métricas de validación y evaluación, como el índice de Dunn o el coeficiente de partición fuzzy. Estas métricas pueden ayudar a determinar el valor de m que produce los resultados de clustering más satisfactorios para un problema y conjunto de datos específicos.

Ventajas y desventajas

El algoritmo Fuzzy C-means (FCM) tiene ventajas y desventajas que vale la pena tener en cuenta:

Ventajas

1. **Flexibilidad en asignación de pertenencia:** A diferencia de los algoritmos de clustering tradicionales, FCM permite asignar grados de pertenencia difusos a los puntos de datos, lo que refleja la incertidumbre en la asignación a clusters. Esto es especialmente útil cuando los puntos de datos tienen características ambiguas o pueden pertenecer a múltiples grupos simultáneamente.

2. **Tolerancia al ruido y atipicidad**: FCM es menos sensible a los valores atípicos y al ruido en los datos, ya que utiliza grados de pertenencia en lugar de asignaciones binarias. Esto permite que los puntos de datos se consideren parcialmente miembros de diferentes clusters, incluso si están cerca de los límites de los grupos.

3. **Mayor información sobre similitudes:** Al proporcionar grados de pertenencia difusos, FCM puede revelar información sobre la similitud relativa entre los puntos de datos.

Esto puede ser útil en aplicaciones donde se necesita medir la cercanía o la relación entre diferentes grupos.

Desventajas

1. **Sensibilidad a la inicialización**: El rendimiento del algoritmo FCM puede depender en gran medida de la inicialización de los centroides y los grados de pertenencia. Una mala inicialización puede conducir a soluciones subóptimas o a la convergencia a puntos de referencia locales.

2. **Necesidad de ajustar el parámetro de fuzziness**: El parámetro de fuzziness (m) en FCM controla la difusión de los grados de pertenencia y afecta la forma en que los puntos de datos se asignan a los clusters. En la práctica, encontrar el valor óptimo de m puede ser un desafío y puede requerir ajustes y experimentación.

3. **Mayor complejidad computacional**: El algoritmo FCM es más computacionalmente costoso que los algoritmos de clustering tradicionales, como K-means, debido a la necesidad de calcular y actualizar los grados de pertenencia en cada iteración. Esto puede ser un problema en grandes conjuntos de datos o cuando se requiere una ejecución rápida.

Es importante considerar estas ventajas y desventajas al seleccionar el algoritmo de clustering más adecuado para una tarea específica. En algunos casos, FCM puede ser la mejor opción cuando se requiere flexibilidad y tolerancia al ruido, pero también es importante tener en cuenta sus limitaciones y las peculiaridades del conjunto de datos en cuestión.

Clustering Jerárquico

Clustering jerárquico es un método de agrupamiento que crea una jerarquía de clusters de manera recursiva, donde los clusters se agrupan en subgrupos más pequeños o se fusionan en grupos más grandes. Este enfoque se basa en la idea de que los objetos similares se agrupan juntos formando estructuras jerárquicas.

A diferencia de otros algoritmos de clustering, el clustering jerárquico no requiere un número predefinido de clústeres y permite explorar las relaciones entre los datos a diferentes niveles de granularidad. Este algoritmo es especialmente útil cuando se busca una visión global y detallada de la estructura de los datos.

Algoritmo

El algoritmo de clustering jerárquico puede ser de dos tipos: aglomerativo y divisivo.

1. **Clustering jerárquico aglomerativo:**
    a. Comienza con cada punto de datos como un cluster individual.

   b. En cada paso, los clusters más cercanos se fusionan en un nuevo cluster.
   c. Este proceso se repite hasta que todos los puntos de datos estén en un solo cluster o hasta que se cumpla un criterio de detención.

2. **Clustering jerárquico divisivo:**
   a. Comienza con todos los puntos de datos en un solo cluster.
   b. En cada paso, el cluster actual se divide en subclusters más pequeños.
   c. Este proceso se repite hasta que cada punto de datos esté en su propio cluster o hasta que se cumpla un criterio de detención.

Los criterios de enlace utilizados en el algoritmo de clustering jerárquico determinan cómo se mide la distancia o similitud entre los clusters. Algunos de los criterios de enlace comunes son:

- **Enlace completo (Complete linkage)**: Mide la distancia máxima entre todos los pares de puntos de datos de diferentes clusters.

$$d_{single}(G,\ H)\ =\ \max_{i\in G,\ j\in H}\ (d_{ij})$$

- **Enlace simple (Single linkage)**: Mide la distancia mínima entre todos los pares de puntos de datos de diferentes clusters.

$$d_{single}(G,\ H)\ =\ \min_{i\in G,\ j\in H}\ (d_{ij})$$

- **Enlace promedio (Average linkage)**: Calcula la distancia promedio entre todos los pares de puntos de datos de diferentes clusters.

$$d_{single}(G,\ H)\ =\frac{1}{n_G n_H}\ \sum_{i\in G,\ j\in H}\ (d_{ij})$$

- **Enlace de Ward (Ward's linkage)**: Minimiza la suma de las diferencias cuadradas dentro del cluster al fusionar dos clusters.

$$d_{ward}(G,\ H)\ =\ ||\ G\ -\ H||^{2}$$

Dendrogramas

Un dendrograma es una representación gráfica utilizada en el clustering jerárquico para mostrar la estructura de los clústeres formados durante el proceso de agrupamiento. Tiene varias características importantes:

1. **Estructura jerárquica:** Un dendrograma muestra la estructura jerárquica de los clústeres, lo que significa que se organiza de manera ascendente desde los clústeres individuales hasta el clúster que contiene todos los datos. Cada nivel en el dendrograma representa un nivel de agrupamiento.

2. **Eje vertical:** El eje vertical del dendrograma representa la medida de similitud o distancia utilizada en el algoritmo de clustering jerárquico. A medida que se sube en el eje vertical, los clústeres se fusionan o dividen en función de su similitud.

3. **Altura o longitud de las ramas:** La altura o longitud de las ramas en el dendrograma indica la medida de disimilitud o distancia entre los clústeres. Cuanto más larga sea la rama, mayor será la diferencia entre los clústeres que se están fusionando o dividiendo.

4. **Nodos y hojas:** Los nodos en el dendrograma representan los clústeres formados en cada nivel del agrupamiento. Los nodos internos indican la fusión o división de clústeres, mientras que las hojas representan los clústeres individuales o puntos de datos.

5. **Corte o umbral:** Para determinar el número óptimo de clústeres, se puede establecer un umbral o corte en el dendrograma. Al cortar el dendrograma en un nivel determinado, se obtienen los clústeres deseados. Esto proporciona flexibilidad en la selección del número de clústeres.

6. **Interpretación:** Los dendrogramas permiten una interpretación visual de la estructura de los clústeres. Pueden revelar la presencia de clústeres principales, subgrupos y patrones complejos en los datos. La forma y la altura de las ramas en el dendrograma proporcionan información sobre la similitud o disimilitud entre los clústeres.

7. **Visualización y análisis:** Los dendrogramas son útiles para visualizar y analizar grandes conjuntos de datos. Proporcionan una visión global de la estructura de clústeres y permiten una exploración detallada de los patrones y relaciones entre los datos.

En resumen, los dendrogramas son herramientas poderosas para visualizar la estructura jerárquica de los clústeres en el clustering jerárquico. Proporcionan información valiosa para comprender las relaciones y patrones en los datos, y permiten la selección del número óptimo de clústeres.

Ventajas y desventajas

El algoritmo cluster jerárquicos tiene ventajas y desventajas que vale la pena tener en cuenta como los siguientes:

Ventajas

1. No se necesita especificar el número de clusters de antemano.

2. Proporciona una visualización natural de la estructura de los datos a través del dendrograma.
3. Permite identificar tanto clusters globales como locales en los datos.
4. No depende de suposiciones sobre la forma de los clusters, lo que lo hace más flexible en comparación con otros métodos de clustering.

Desventajas

1. Requiere más tiempo de cómputo en comparación con otros algoritmos de clustering, especialmente cuando se trata de conjuntos de datos grandes.
2. La complejidad del dendrograma puede dificultar la interpretación, especialmente en conjuntos de datos con muchos puntos.
3. No es adecuado para conjuntos de datos con formas de cluster no esféricas o con diferentes tamaños de cluster.

En general, el clustering jerárquico es un enfoque versátil y útil en el análisis de datos. Sin embargo, es importante considerar las ventajas y desventajas específicas en relación con los requisitos y características de los datos en cada caso particular.

Validación de los modelos de clustering

Validar la calidad de los modelos de clustering es una tarea crucial en el aprendizaje no supervisado. La evaluación de los resultados del clustering nos permite comprender la eficacia y la coherencia de los clústeres identificados, así como la calidad del modelo en general. Sin embargo, validar los modelos de clustering presenta desafíos únicos debido a la falta de etiquetas o información de referencia en el proceso de entrenamiento.

La importancia de la validación de la calidad de modelos de clustering radica en varios aspectos. En primer lugar, nos permite evaluar la coherencia y consistencia de los clusters generados. Un buen modelo de clustering debería ser capaz de identificar clusters significativos y distinguir entre patrones reales y ruido. Además, la validación también ayuda a seleccionar el número óptimo de clústeres y a determinar la adecuación del algoritmo de clustering utilizado para un conjunto de datos específico. Esto nos permite obtener resultados más confiables y útiles en aplicaciones del mundo real.

No obstante, la validación de modelos de clustering presenta desafíos debido a la falta de etiquetas o respuestas correctas predefinidas. No podemos utilizar métricas tradicionales de evaluación supervisada, como la precisión o el F1-score, ya que no tenemos información sobre las clases o categorías reales de los datos. Además, la calidad de los resultados del clustering es subjetiva y depende de la interpretación y el contexto del problema.

Algunas de las razones por las cuales la validación de un modelo de clustering puede ser difícil:

1. **Ausencia de etiquetas:** En el aprendizaje no supervisado, no hay etiquetas disponibles para medir el rendimiento del modelo. No se sabe de antemano a qué grupo o categoría pertenece cada instancia de datos, lo que dificulta la comparación directa entre las asignaciones de cluster del modelo y las etiquetas verdaderas.
2. **Subjetividad de los resultados:** La interpretación de los resultados de clustering puede ser subjetiva y depende del criterio o percepción del analista. Diferentes personas pueden agrupar los datos de manera diferente según su enfoque o criterios seleccionados, lo que dificulta establecer una medida objetiva de la calidad del modelo.
3. **Evaluación intrínseca:** Al no tener etiquetas, la evaluación del modelo se basa en medidas intrínsecas o internas que consideran la estructura y coherencia de los grupos generados. Estas medidas pueden ser útiles para comparar diferentes ejecuciones del modelo o ajustar parámetros, pero no proporcionan una evaluación absoluta o externa del rendimiento del modelo.

Para abordar estos desafíos, existen métodos de validación de modelos de clustering que se dividen en dos categorías principales: métodos cuantitativos y métodos cualitativos.

Métodos Cuantitativos

Los métodos cuantitativos utilizan medidas numéricas y estadísticas para evaluar la calidad del clustering. Algunos de los métodos más comunes son los siguientes:

1. **Distancia Intra Cluster (SSE):** La distancia intra cluster mide la coherencia de los objetos dentro de un mismo grupo o clúster. Cuanto menor sea la distancia intracluster, mayor será la cohesión dentro del cluster. Se busca que los objetos dentro del mismo cluster estén lo más cerca posible unos de otros.

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} (x - m_i)^2$$

2. **Coeficiente de Silhouette:** Calcula la similitud intracluster y la disimilitud intercluster para cada punto de datos. Proporciona una medida de cuán bien se agrupan los datos en clusters compactos y separados.

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

a = distancia promedio de i a los puntos del propio cluster
b = mínima distancia promedio de i a los punteros de otro cluster

3. **Índice de Davies-Bouldin:** Mide la dispersión dentro de los clústeres y la separación entre clústeres. Un valor más bajo indica una mejor calidad de clustering.

$$DB = \frac{1}{k}\sum_{i=1}^{k} \max_{i != j} R_{ij}$$

donde:

s_i = promedio de distancia de cada punto del cluster i al centroide del cluster.

d_{ij} = distancia del centroide del cluster i al centroide del cluster j.

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

Estas medidas cuantitativas proporcionan una evaluación objetiva de la calidad del clustering y pueden utilizarse para comparar diferentes modelos o ajustar parámetros.

Métodos Cualitativos

Los métodos cualitativos se basan en la interpretación y el juicio de expertos para evaluar la calidad del clustering. Estos métodos incluyen:

1. **Evaluación visual:** Los expertos pueden inspeccionar visualmente los resultados del clustering, examinar los dendrogramas o diagramas de dispersión, y evaluar la coherencia y la interpretabilidad de los clústeres identificados.

2. **Evaluación basada en dominio experto :** Los expertos en el dominio pueden utilizar su conocimiento y comprensión del problema para evaluar la utilidad y relevancia de los clústeres identificados. Pueden determinar si los clústeres tienen sentido desde la perspectiva del problema y si proporcionan información valiosa. Los expertos pueden realizar las siguientes actividades:
   - Revisar y evaluar visualmente los clusters generados en función de su conocimiento y experiencia.
   - Interpretar y asignar etiquetas o categorías a los clusters identificados.
   - Evaluar la coherencia y relevancia de los grupos generados en relación con los objetivos del análisis.

   La validación por expertos proporciona una perspectiva subjetiva y basada en el conocimiento experto, lo que puede ser especialmente valioso en dominios específicos donde la interpretación y la relevancia de los clusters son críticas. Sin embargo, puede haber cierto grado de subjetividad y variabilidad en las evaluaciones realizadas por diferentes expertos.

3. **Evaluación de estabilidad:** Los métodos de bootstrap o de remuestreo se pueden utilizar para evaluar la estabilidad de los clústeres y la consistencia de los resultados del clustering. Esto implica ejecutar el algoritmo de clustering varias veces con diferentes muestras de datos y analizar la estabilidad de los clústeres resultantes.

Es importante tener en cuenta que la validación de modelos de clustering es un proceso iterativo y depende en gran medida del contexto y los objetivos del problema. Combinar métodos cuantitativos y cualitativos puede proporcionar una evaluación más completa y confiable de la calidad del clustering.

Actividad guiada: Clustering con Python

KMeans y método del codo

1. Importamos las librerías más importantes y el dataset **wine**. Comenzaremos con KMeans, y para simplificar consideraremos solo las 3 primeras columnas

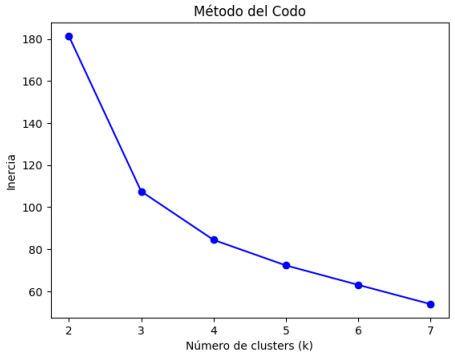
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_wine
from sklearn.cluster import KMeans
import seaborn as sns
from sklearn.metrics import silhouette_score
from sklearn.metrics import davies_bouldin_score
from sklearn.metrics import pairwise_distances


# Cargar el conjunto de datos wine. Para simplificar utilizaremos solo 3 columnas
wine = load_wine()
data = pd.DataFrame(wine.data)
X = data.iloc[:, 0:3]
```

2. Vamos a utilizar el método del codo para determinar el número óptimo de clusters.

```
#Definimos una lista de posibles valores, desde 2 hasta 8
k_values = range(2, 8)
# Inicializar listas para almacenar las métricas
inertia_values = []
# Realizar clustering con diferentes valores de k y calcular las métricas
for k in k_values:
kmeans = KMeans(n_clusters=k, random_state=42)
kmeans.fit(X)
labels = kmeans.labels_
# Calcular la inercia
inertia_values.append(kmeans.inertia_)
# Graficar el método del codo utilizando la inercia
plt.plot(k_values, inertia_values, 'bo-')
```

```
plt.xlabel('Número de clusters (k)')
plt.ylabel('Inercia')
plt.title('Método del Codo')
plt.show()
```

El gráfico nos indica que para k = 3 se produce una disminución sensible de la inercia



3. Aplicamos ahora el algoritmo de Kmeans

```
# Realizar clustering con el valor óptimo de k (elegido por el método del
codo)
optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
kmeans.fit(X)labels = kmeans.labels_
X['clusters'] = labels #agregamos a la data una columna con las etiquetas de
los clusters


# Graficar los clusters en el espacio reducido
plt.scatter(X.iloc[:, 1], X.iloc[:, 2], c=labels)
plt.title('Clusters K-means (k=3)')
plt.show()
plt.show()
```

4. Calculamos los indicadores de Silhouette y Davies Boludin.

```
# Calcular la puntuación de silueta
silhouette_scores = silhouette_score(X, labels)
# Calcular el índice de Dunn
dunn_index = davies_bouldin_score(X, labels)
print(f'Silhoutte Score : {silhouette_scores}')
print(f'Davies Bouldin Score : {dunn_index}')
```

Fuzzy C Means

5. Veremos ahora el algoritmo Fuzzy CMeans. Para ello es necesario cargar la biblioteca skfuzzy

```
!pip install scikit-fuzzy #podría ser necesario instalar la biblioteca
import skfuzzy as fuzz
```

6. Vamos a determinar el valor del parámetro fuzzy m. Para ello probaremos con una lista de valores desde 1.2 hasta 3.1, incrementando cada 0.1

```
# Definir una lista de posibles valores de m
m_values = np.arange(1.1, 3.1, 0.1)

# Inicializar listas para almacenar las métricas
silhouette_scores_fuzzy = []
davies_bouldin_scores_fuzzy = []
```

```
A partir de esto realizaremos Clustering Fuzzy con diferentes valores de m,
y calcularemos las métricas respectivas
for m in m_values:
   cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(X.T, 3, m, error=0.005,
maxiter=1000)
   labels = np.argmax(u, axis=0)

# Calculamos las métricas
   silhouette_scores_fuzzy.append(silhouette_score(X, labels))
   dunn_index = davies_bouldin_score(X, labels)
   davies_bouldin_scores_fuzzy.append(dunn_index)
```

En este código tenemos:

- **cntr**: Es una matriz que contiene los centroides finales de los clusters encontrados por el algoritmo Fuzzy C-Means. Cada fila representa un centroide y cada columna representa una característica.
- **u:** Es una matriz que contiene los grados de pertenencia de cada punto a cada cluster. Cada fila representa un punto y cada columna representa un cluster. Los valores en u son números entre 0 y 1 que representan la probabilidad de que un punto pertenezca a cada cluster.

- **u0:** Es una matriz que contiene los grados de pertenencia iniciales de cada punto a cada cluster. Al inicio del algoritmo, estos valores se inicializan aleatoriamente o de alguna otra manera.
- **d:** Es una matriz que contiene las distancias entre los puntos y los centroides finales de los clusters. Cada fila representa un punto y cada columna representa un cluster.
- **jm:** Es el valor de la función objetivo del clustering fuzzy. Representa el valor de la función de costo que se intenta minimizar durante el proceso de clustering.
- **p:** Es un valor que representa el exponente utilizado en la función de costo para el clustering fuzzy.
- **fpc:** Es el índice de partición fuzzy. Representa la fracción de puntos que se encuentran correctamente asignados en los clusters finales.
- **error=0.005:** Es el criterio de parada del algoritmo FCM. El algoritmo continuará iterando hasta que la diferencia en el valor de la función objetivo entre dos iteraciones consecutivas sea menor que el valor de error. En este caso, el algoritmo se detendrá cuando la diferencia en la función objetivo sea menor o igual a 0.005.
- **maxiter=1000:** Es el número máximo de iteraciones permitidas para el algoritmo FCM. Si el algoritmo no converge después de maxiter iteraciones, se detendrá. En este caso, el algoritmo se detendrá después de 1000 iteraciones si no ha convergido antes.

A partir del coeficiente de Silhouette nos quedaremos con el valor óptimo para **m**

```
optimal_index = np.argmax(silhouette_scores_fuzzy)
optimal_m = m_values[optimal_index]
```

7. Realizaremos ahora el clustering con los valores obtenidos, y graficaremos

```
optimal_cntr, optimal_u, optimal_u0, optimal_d, optimal_jm, optimal_p,
optimal_fpc = fuzz.cluster.cmeans(X.T, 3, optimal_m, error=0.005,
maxiter=1000)
optimal_labels = np.argmax(optimal_u, axis=0)

# Graficar los resultados en 2D
plt.figure(figsize=(10, 4))
plt.subplot(121)
plt.scatter(X.iloc[:, 1], X.iloc[:, 2], c=wine.target, cmap='viridis')
plt.title('Clasificación real')
plt.subplot(122)
plt.scatter(X.iloc[:, 1], X.iloc[:, 2], c=optimal_labels, cmap='viridis')
plt.title('Clustering Fuzzy C-means')
plt.show()

# Imprimir el valor óptimo de m
print(f"Valor óptimo de m: {optimal_m}")
```

8. Graficaremos las métricas, para observar cómo se estabilizan.

```
plt.figure(figsize=(8, 4))
plt.plot(m_values, silhouette_scores_fuzzy, 'bo-', label='Puntuación de
silueta')
plt.plot(m_values, davies_bouldin_scores_fuzzy, 'ro-', label='Índice de
Dunn')
plt.xlabel('Valor de m')
plt.ylabel('Métrica')
plt.title('Evaluación de Clustering Fuzzy C-means')
plt.legend()
plt.show()
```

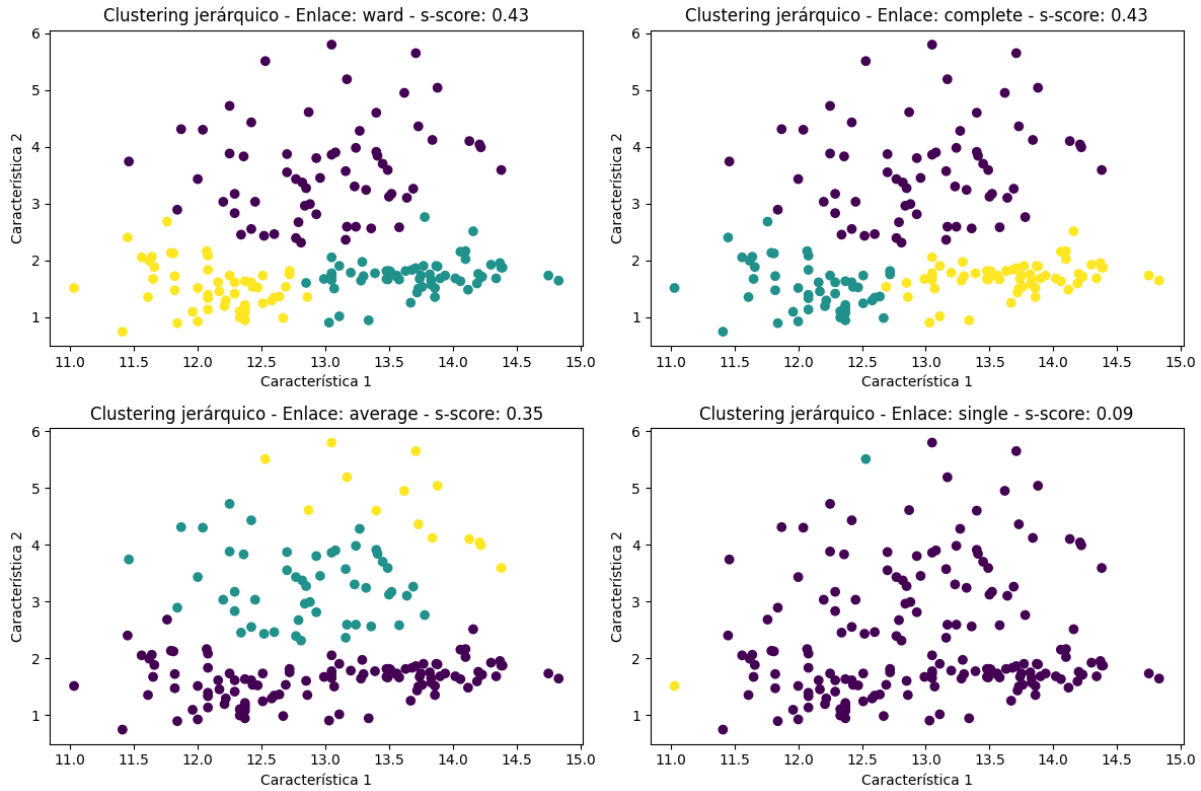
Cluster jerárquicos

9. Veremos ahora los cluster jerárquicos, para las cuales necesitaremos importar más funciones

```
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import linkage, dendrogram
```

10. Realizaremos clustering jerárquico utilizando diferentes métodos de enlace, indicados en una lista.

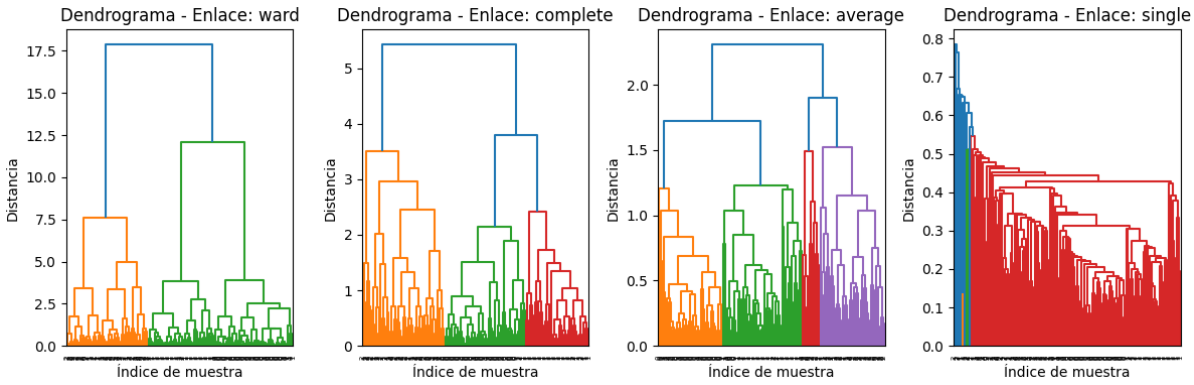
```
linkage_methods = ['ward', 'complete', 'average', 'single']

plt.figure(figsize=(12, 8))
for i, method in enumerate(linkage_methods):
# Realizar clustering jerárquico con el método de enlace actual
clustering = AgglomerativeClustering(n_clusters=3, linkage=method)
labels = clustering.fit_predict(X)
sc = silhouette_score(X, labels)

# Graficar los puntos en un diagrama de dispersión para las dos primeras
columnas
plt.subplot(2, 2, i+1)
plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=labels, cmap='viridis')
plt.xlabel('Característica 1')
plt.ylabel('Característica 2')
plt.title(f'Clustering jerárquico - Enlace: {method} - s-score: {sc:.2f}')

plt.tight_layout()
plt.show()
```



Dendogramas

11. Podemos ahora ver los dendogramas

```
plt.figure(figsize=(12, 4))
for i, method in enumerate(linkage_methods):
# Realizar clustering jerárquico con el método de enlace actual
   Z = linkage(X, method)

# Convertir la matriz de enlace a tipo float
   Z = Z.astype(float)

# Graficar el dendrograma
   plt.subplot(1, 4, i+1)
   dendrogram(Z, labels=wine.target)
   plt.xlabel('Índice de muestra')
   plt.ylabel('Distancia')
   plt.title(f'Dendrograma - Enlace: {method}')

plt.tight_layout()
plt.show()
```



12. Finalmente, analizaremos el mejor cluster obtendo

```
#cluster con mejor método
clustering = AgglomerativeClustering(n_clusters=3, linkage='ward')
labels = clustering.fit_predict(X)

# Analizar los clusters obtenidos
for cluster in range(3):
   cluster_indices = np.where(labels == cluster)[0]
   cluster_samples = wine.data[cluster_indices]
   cluster_target = wine.target[cluster_indices]
   cluster_name = wine.target_names[cluster]
   print(f'Cluster {cluster}: {cluster_name}')
   print(f'Número de muestras: {len(cluster_samples)}')
 print(f'Características más repres: {np.mean(cluster_samples,
axis=0)}')
   print(f'Etiquetas reales en cluster: {np.unique(cluster_target)}')
  print('---')
```

Puedes observar este código en acción, probarlo y modificarlo en el archivo **02 - Algoritmos de clustering**

Preguntas de proceso

Reflexiona:

- ¿Cuáles son algunas métricas comunes utilizadas para evaluar la calidad de un modelo de clustering?
- ¿Cuál es la diferencia entre la validación interna y la validación cuantitativa y cualitativa?
- ¿Cómo se puede determinar el número óptimo de clusters en un conjunto de datos utilizando técnicas de validación?
- ¿Cuál es la importancia de la validación por expertos en la evaluación de la calidad de los clusters generados por un modelo de clustering?
- ¿Qué algoritmos de clustering conoces?

Referencias bibliográficas

1. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
2. Python Data Science Handbook: https://jakevdp.github.io/PythonDataScienceHandbook
3. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
4. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html

¡Continúa aprendiendo y practicando!