Selección óptima de algoritmos de clustering e hiperparámetros: un enfoque basado en métricas de calidad

Álvaro Jesús Bernal Caunedo
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
alvbercau@alum.us.es - alvarobc2412@gmail.com

Álvaro González Frías
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
alvgonfri@alum.us.es

***Resumen*—El objetivo de este estudio es realizar una comparación exhaustiva de diez algoritmos de clustering. Se utilizarán tres métricas de evaluación comúnmente utilizadas en el campo del clustering para dicha comparación. El objetivo es determinar qué algoritmo de clustering es más efectivo en función de los hiperparámetros utilizados, haciendo uso del mismo conjunto de datos.**

Se llevó a cabo una evaluación detallada de los diez algoritmos de clustering utilizando las métricas Silhouette, Davies-Bouldin y Calinski-Harabasz. Los resultados revelaron que los algoritmos de clustering dependen significativamente del conjunto de datos y el objetivo de la clasificación. En general, se observó que K-means, Mini-batch K-means y Spectral Clustering obtuvieron los mejores resultados en términos de las tres métricas para el conjunto de datos dado. Sin embargo, se observó que algunos algoritmos, como DBSCAN, presentaban resultados subóptimos. En general, este estudio proporciona una guía útil para la selección de los hiperparámetros más apropiados según el algoritmo de clustering utilizado.

***Palabras clave*—Clustering, Birch, DBSCAN, K-means, Mini-batch K-means, Mean Shift, OPTICS, Mixture of Gaussians, Partition around medoids, UPGMA, Silhouette coefficient, Calinski-Harabasz Index, Davies-Bouldin Index.**

I. Introducción

El clustering, o agrupamiento de datos, es una técnica fundamental en el campo de la minería de datos y el aprendizaje automático. Su objetivo principal es identificar patrones y estructuras ocultas en conjuntos de datos, dividiéndolos en grupos homogéneos, también conocidos como clusters. Estos grupos permiten una comprensión más profunda de los datos, así como la extracción de información útil para la toma de decisiones.

A lo largo de los años, se han desarrollado diversos algoritmos de clustering, cada uno con sus propias fortalezas y debilidades. Sin embargo, la selección de un algoritmo de clustering adecuado para un conjunto de datos específico sigue siendo un desafío. Diferentes algoritmos pueden funcionar mejor en diferentes situaciones, dependiendo de las características de los datos y los objetivos del análisis. Además, muchos algoritmos de clustering requieren la especificación de hiperparámetros, que son configuraciones que afectan el rendimiento y la calidad de los resultados obtenidos.

En este contexto, surge la necesidad de una metodología que permita seleccionar los algoritmos de clustering más adecuados y los valores óptimos de los hiperparámetros. Esta selección debe basarse en métricas de calidad que evalúen la coherencia interna de los clusters, la separación entre ellos y la estabilidad de los resultados.

En este artículo científico, presentamos un enfoque novedoso para la selección óptima de algoritmos de clustering e hiperparámetros, basado en métricas de calidad. Nuestra propuesta busca plantear una posible solución al problema de seleccionar el algoritmo de clustering más apropiado en un contexto determinado.

En las siguientes secciones, indicaremos los diferentes algoritmos de clustering que se han tenido en cuenta, presentaremos las métricas de calidad que utilizamos para evaluar los resultados del clustering, describiremos en detalle nuestro enfoque y mostraremos los resultados de experimentos realizados en un conjunto de datos de referencia.

En resumen, este artículo contribuye al campo del clustering al proponer una nueva metodología para la selección óptima de algoritmos e hiperparámetros. Nuestra propuesta tiene el potencial de mejorar la eficiencia y la calidad de los análisis de clustering, permitiendo una comprensión más profunda de los datos y un mejor apoyo en la toma de decisiones en diversas áreas de aplicación.

II. Preliminares

En esta sección se definen aspectos relacionados con el marco teórico, que constituye la base conceptual sobre la cual se fundamenta esta investigación. Con este fin, se presentan las teorías y los conceptos clave que sustentan el estudio, proporcionando una comprensión profunda del tema y estableciendo un contexto para el análisis posterior.

En primer lugar, haremos referencia a algunos trabajos que guardan algún tipo de relación con la presente investigación, y que pueden ser de utilidad para adquirir un mayor conocimento en el campo del clustering:

- *Algoritmos de agrupamiento:* Éste es un estudio realizado sobre diferentes técnicas de agrupamiento, las que se encuentran dentro del campo de estudio del Reconocimiento de Patrones, haciendo hincapié en las técnicas basadas en densidad (DBSCAN, K-means y CURE) [1].
- *Comparación de diferentes algoritmos de clustering en la estimación de coste en el desarrollo de software:* En este trabajo se plantea, como mejora del proceso de estimación, segmentar la base de datos ISBSG en diferentes grupos de proyectos mediante la utilización de tres algoritmos de agrupamiento diferentes: COBWEB, EM, y k-means, de manera que para cada uno de estos grupos (formados por proyectos homogéneos entre sí) se obtenga una relación matemática diferente [2].

Seguidamente, se describen los diez algoritmos que han sido empleados para realizar la comparativa:

A. BIRCH

El algoritmo de clustering BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) es un método de agrupamiento que utiliza una estructura de árbol llamada CF-tree (Clustering Feature tree) para representar los datos de manera compacta y facilitar el proceso de agrupamiento. Consta de dos fases principales: construcción del árbol y refinamiento de los clusters. Además, cuenta con dos hiperparámetros clave: el *Branching Factor* (Factor de ramificación) y el *Threshold* (Umbral de distancia) [3].

En la fase de construcción del árbol, BIRCH recorre los datos de entrada de manera iterativa. En cada iteración, se selecciona un objeto y se actualiza la estructura del CF-tree para incorporar el nuevo objeto. El CF-tree mantiene información resumida sobre los datos de cada subcluster, empleada para calcular los distintos centroides.

En la fase de refinamiento, BIRCH realiza una búsqueda en el árbol para identificar los clusters finales. Durante este proceso, se realiza una optimización iterativa para mejorar la calidad de los clusters. Esto se logra mediante la actualización de los centroides y la reasignación de puntos de datos si es necesario.

El Branching Factor (B) determina el número máximo de hijos que puede tener un nodo en el árbol CF-tree. Un valor bajo de B implica una menor ramificación en el árbol CF-tree, lo que resulta en clusters más compactos. En contraste, un valor alto de B permitirá una mayor ramificación, lo que puede dar lugar a clusters más pequeños y dispersos en el árbol CF-tree.

El Threshold (T) establece el umbral de distancia utilizado para decidir si un subárbol representa un cluster válido. Si la distancia entre un objeto y el centroide de un subcluster es menor que el umbral, el objeto se considera parte del cluster. Un valor más bajo de T indica un umbral de distancia más estricto, lo que resulta en clusters más compactos y densos. Por el contrario, un valor más alto de T permite una mayor variabilidad en la distancia y puede generar clusters más dispersos y más grandes.

B. DBSCAN

DBSCAN estima la densidad contando el número de puntos en una vecindad de radio fijo y considera que dos puntos están conectados si se encuentran dentro de la vecindad del otro. El punto A se denomina un punto central si la vecindad de radio Eps contiene al menos puntos MinPts, es decir, la densidad en la vecindad debe superar un cierto umbral. Un punto Q es directamente alcanzable por la densidad desde un punto central P si Q está dentro de la vecindad de P con radio Eps. Dos puntos P y Q se denominan conectados por densidad si existe un tercer punto O desde el cual tanto P como Q son alcanzables por la densidad [4].

Un cluster es un conjunto de puntos conectados por densidad que es maximal con respecto a la densidad-alcanzabilidad. El ruido se define como el conjunto de puntos de la base de datos que no pertenecen a ninguno de sus clústeres. La tarea de agrupación es encontrar todos los clusters con respecto a los parámetros Eps y MinPts en un conjunto de datos dado. Para su implementación se ha hecho uso del método DBSCAN de la librería *Scikit-learn*, usando como hiperparámetros Eps y MinPts (citados anteriormente).

C. K-means

El algoritmo k-means es un método ampliamente utilizado en el campo del agrupamiento de datos e información. Su objetivo es agrupar un conjunto de datos no etiquetados en k clusters, donde k es un parámetro predefinido. Se basa en la minimización de la distancia entre los puntos de datos y los centroides de los clusters a los que pertenecen. A continuación, se describe su funcionamiento paso a paso: [5]

1) *Inicialización:* Seleccionamos k puntos como centroides iniciales. Esto se puede hacer de varias maneras, como elegir aleatoriamente k puntos del conjunto de datos o utilizar algún criterio de selección más sofisticado.
2) *Asignación de puntos a clusters:* Cada punto de datos se asigna al cluster cuyo centroide esté más cerca según alguna medida de distancia, generalmente la distancia euclidiana. Esto se repite para todos los puntos de datos, creando así agrupaciones iniciales.
3) *Actualización de los centroides:* Una vez que todos los puntos se han asignado a clusters, se recalcula la posición del centroide de cada cluster como el promedio de todos los puntos que pertenecen a ese cluster. Este paso actualiza la posición de los centroides.
4) *Repetición de pasos 2 y 3:* Los pasos de asignación y actualización se repiten iterativamente hasta que se cumpla algún criterio de parada, como que los centroides ya no cambien significativamente o que se alcance un número máximo de iteraciones.

El algoritmo converge hacia una solución donde los puntos dentro de cada cluster son similares entre sí y diferentes de los puntos en otros clusters. Sin embargo, es importante tener en cuenta que el resultado final puede depender de la inicialización de los centroides y de la elección del valor de k.

D. Mini-Batch K-means

El algoritmo de clustering Mini-batch K-means es una versión del algoritmo K-means que puede utilizarse en lugar del algoritmo K-means cuando se trabaja con clustering en conjuntos de datos enormes. A veces funciona mejor que el algoritmo K-means estándar cuando se trabaja con conjuntos de datos enormes porque no itera sobre todo el conjunto de datos. Crea lotes aleatorios de datos que se almacenan en la memoria y, a continuación, recoge un lote aleatorio de datos en cada iteración para actualizar los clusters.

En el primer paso, las muestras se extraen aleatoriamente del conjunto de datos para formar un mini-lote. Posteriormente, dichas muestras se asignan al centroide más cercano y éste se actualiza tomando el promedio de transmisión de la muestra y todas las muestras anteriores asignadas a ese centroide. Esto se realiza hasta que se alcanza la convergencia o un número de iteraciones determinado. Cabe objetar que aunque Mini-Batch K-means reduce el coste computacional, también reduce la calidad del resultado. [6]

E. Mean Shift

El objetivo del algoritmo Mean Shift es encontrar regiones de alta densidad en el espacio de características de los datos. Seguidamente, se detalla el define el proceso detalladamente: [7]

1) *Selección del parámetro de ancho de banda (bandwidth):* El ancho de banda es un parámetro crítico en Mean Shift y determina la sensibilidad del algoritmo para agrupar los datos. Seleccionar un valor adecuado para el ancho de banda es fundamental para obtener resultados precisos. Un ancho de banda pequeño conduce a una segmentación fina de los datos, mientras que uno grande puede agrupar datos más distantes en un mismo cluster. En nuestro caso, estimaremos el bandwidth estableciendo dos hiperparámetros: *quantile* (Q) y *n samples* (N). El parámetro Q se refiere a un valor entre 0 y 1 que determina el tamaño relativo del ancho de banda. Un valor más pequeño de Q resultará en un ancho de banda más estrecho, lo que implica una segmentación más fina de los datos. Por el contrario, un valor más alto de Q producirá un ancho de banda más amplio y agrupará más puntos juntos. El parámetro N representa el número de muestras utilizadas para estimar el ancho de banda. Aumentar el valor de N puede proporcionar una estimación más precisa del ancho de banda, ya que se consideran más puntos de datos. Sin embargo, también puede aumentar la complejidad computacional, especialmente en conjuntos de datos grandes.
2) *Estimación de la densidad de los datos:* Se calcula una función de densidad suavizada usando los datos y el ancho de banda seleccionado. Los puntos con mayor densidad serán considerados como centroides potenciales.
3) *Desplazamiento de los puntos hacia los centroides:* Para cada punto de datos, se calcula la dirección hacia el centroide más cercano. Se actualiza la posición del punto siguiendo esta dirección hasta que converge a un centroide. Este proceso se repite para todos los puntos hasta que se alcanza la convergencia.
4) *Asignación de etiquetas a los grupos:* Después de la convergencia, los puntos se asignan a los grupos correspondientes según su centroide más cercano.

El algoritmo de clustering Mean Shift es una técnica eficaz y versátil para la identificación de grupos en conjuntos de datos. A diferencia de otros algoritmos de clustering, Mean Shift no requiere la especificación previa del número de grupos y es capaz de descubrir automáticamente la estructura de los datos. Esto lo convierte en una herramienta valiosa en aplicaciones donde no se tiene conocimiento previo sobre la distribución de los grupos.

F. OPTICS

En muchas bases de datos del mundo real, la estructura intrínseca de los clústeres no puede ser caracterizada por parámetros de densidad globales, y se pueden necesitar densidades locales muy diferentes para revelar clústeres en diferentes regiones del espacio de datos.

En principio, se podría aplicar un algoritmo de clustering basado en densidad con diferentes configuraciones de parámetros, pero existen un número infinito de posibles valores de parámetros. La idea básica del algoritmo OPTICS para abordar este desafío es crear no un clustering explícito, sino producir un nuevo ordenamiento de clústeres de los puntos de la base de datos con respecto a su estructura de clustering basado en densidad, que contiene información sobre cada nivel de clustering del conjunto de datos hasta una distancia generadora Eps. Este ordenamiento se visualiza gráficamente para apoyar el análisis interactivo de la estructura de clústeres.

Para un valor constante de MinPts, los clústeres basados en densidad con respecto a una densidad más alta (es decir, un valor más bajo para Eps) están completamente contenidos en clústeres con respecto a una densidad más baja (es decir, un valor más alto para Eps).

En consecuencia, el algoritmo DBSCAN podría extenderse para clusterizar simultáneamente una base de datos para varios valores de Eps. Sin embargo, los puntos que son densidad-alcanzables con respecto al valor más bajo de Eps siempre deben procesarse primero para garantizar que los clústeres con respecto a una densidad más alta se finalicen primero. OPTICS funciona en principio como un algoritmo DBSCAN extendido para un número infinito de parámetros de distancia Eps_i que son menores que una distancia generadora Eps. La única diferencia es que no asigna membresías de clúster, sino que almacena el orden en el que los puntos son procesados (el ordenamiento de clustering) y la siguiente información que sería utilizada por un algoritmo DBSCAN extendido para asignar membresías de clúster [8].

G. Spectral Clustering

A diferencia de otros algoritmos de clustering, Spectral Clustering se basa en propiedades espectrales de una matriz

de afinidad construida a partir de los datos.

Dado un conjunto de datos de entrada, se construye una matriz de afinidad. Esta matriz captura las relaciones de similitud o proximidad entre los puntos de datos. A partir de la matriz de afinidad, se construye una matriz Laplaciana. La matriz Laplaciana refleja la estructura de conectividad de los datos.

Tras ello, se calculan los autovalores y autovectores de la matriz Laplaciana. Estos autovalores y autovectores representan las características espectrales de la estructura de conectividad de los datos. Los autovalores se ordenan de menor a mayor, y los autovectores correspondientes se organizan en una matriz.

Después, los autovectores correspondientes a los k autovalores más pequeños. Estos autovectores forman una matriz de representación de los datos en un espacio de menor dimensión. La elección de k es un hiperparámetro del algoritmo y determina el número de clusters deseados.

Finalmente, se aplica un algoritmo de agrupamiento, como k-means, a la matriz de representación de menor dimensión para obtener los clusters finales. Los puntos de datos se asignan a los clusters en función de su proximidad en el espacio de menor dimensión.

En resumen, el algoritmo de Spectral Clustering utiliza técnicas de álgebra lineal y propiedades espectrales para agrupar datos en clusters, ofreciendo una alternativa poderosa a los algoritmos de agrupamiento tradicionales. [9]

H. Mixture of Gaussians (GMM)

El Modelo de Mezclas de Gaussianas se basa en la idea de que los datos en un conjunto de datos pueden ser generados por una combinación de varias distribuciones gaussianas. En lugar de asignar puntos de datos a clústeres específicos de manera determinista, GMM asigna probabilidades a cada punto de datos para pertenecer a cada una de las distribuciones gaussianas en el modelo. El objetivo del algoritmo GMM es encontrar la mejor configuración de distribuciones gaussianas y sus respectivas probabilidades para representar la estructura subyacente del conjunto de datos [10].

Además, GMM utiliza el algoritmo EM (*Expectation-Maximization*) para estimar los parámetros óptimos en el modelo de mezclas gaussianas. Este algoritmo alterna entre dos pasos principales: el paso de expectativa, donde se calculan las probabilidades posteriores de las variables latentes dados los parámetros actuales, y el paso de maximización, donde se actualizan los parámetros del modelo utilizando estas probabilidades posteriores. Este proceso iterativo continúa hasta alcanzar la convergencia, mejorando iterativamente los parámetros del modelo y maximizando la verosimilitud de los datos observados.

I. Partition around medoids (PAM)

La principal característica del algoritmo PAM es el uso de medoides. Un medoide es aquel objeto que minimiza la distancia promedio a todos los demás objetos dentro del mismo cluster. Se diferencia del centroide en que el medoide es un punto real de los datos originales, mientras que el centroide es un punto calculado como la media de todos los puntos de un clúster.

El algoritmo PAM comienza seleccionando k objetos aleatorios del conjunto de datos como medoides iniciales, donde k es el número de clusters deseado. Luego, asigna cada objeto restante al medoide más cercano, formando así los clusters iniciales. A continuación, realiza iteraciones para mejorar la calidad de los clusters, intercambiando cada medoide con cada objeto no medoide en su cluster correspondiente y calculando la suma de las distancias entre el nuevo medoide y los objetos de su cluster. Si el intercambio reduce la suma de las distancias, se actualiza el medoide.

Este proceso iterativo se repite hasta que no se puedan encontrar más intercambios beneficiosos, lo que indica que los medoides seleccionados son los más representativos y estables para cada cluster. En este punto, el algoritmo PAM finaliza y devuelve los clusters resultantes.

Resumidamente, el algoritmo PAM es una técnica de clustering basada en medoides que asigna objetos a los medoides más cercanos y busca mejorar iterativamente los medoides para formar clusters estables y representativos. [11]

J. UPGMA

El algoritmo UPGMA (Unweighted Pair Group Method with Arithmetic Mean) es un método utilizado en el clustering jerárquico para agrupar objetos construyendo árboles filogenéticos.

En primer lugar, se calcula una matriz de distancias entre todos los objetos del conjunto de datos. Esta matriz de distancias representa la similitud entre los objetos, donde los valores más bajos indican una mayor similitud.

A continuación, se seleccionan los dos objetos más similares y se agrupan en un nuevo clúster. La distancia entre este nuevo clúster y los demás objetos se calcula utilizando un promedio ponderado de las distancias de los objetos originales.

Este proceso se repite iterativamente, agrupando los clústeres existentes hasta que todos los objetos estén agrupados en un único clúster. Durante cada iteración, se actualiza la matriz de distancias y se recalculan las distancias entre los nuevos clústeres y los objetos restantes.

Finalmente, dependiendo del número de clusters que se quieran representar (hiperparámetro k), se cortará el árbol por una profundidad que genere el número de clusters solicitado. Si no se consigue, se devolverá el más cercano a k.

El algoritmo UPGMA es un enfoque sencillo y eficiente para el clustering jerárquico, ya que solo requiere de la matriz de distancias inicial y no depende de parámetros adicionales.

Por último, se detalla el funcionamiento de las tres métricas empleadas para evaluar la calidad de los grupos obtenidos tras aplicar un algoritmo de clustering:

K. Silhouette Coefficient

El coeficiente de Silhouette se calcula utilizando la distancia media intracluster (a) y la distancia media al cluster más

cercano (b) de cada muestra. El coeficiente de silueta de una muestra es (b - a) / max(a, b). Para mayor claridad, b es la distancia entre una muestra y el conglomerado más cercano del que no forma parte la muestra. El mejor valor es 1 y el peor -1. Los valores cercanos a 0 indican conglomerados superpuestos. Los valores negativos generalmente indican que una muestra ha sido asignada al cluster equivocado, ya que un cluster diferente es más similar [12].

L. Calinski-Harabasz Index

El índice Calinski-Harabasz se basa en la comparación de la relación ponderada entre la suma de los cuadrados (la medida de la separación del clúster) y la suma de los cuadrados dentro del clúster (la medida de cómo se empaquetan estrechamente los puntos dentro de un clúster). Idealmente, los clústeres deben estar bien separados, por lo que la suma entre el valor de los cuadrados debe ser grande, pero los puntos dentro de un clúster deben estar lo más cerca posible el uno del otro, dando como resultado valores más pequeños de la suma dentro del clúster de medida de cuadrados. Dado que el índice Calinski-Harabasz es una relación, con la suma de los cuadrados entre el numerador y la suma de cuadrados dentro del denominador, las soluciones de clúster con valores más grandes del índice corresponden a soluciones ”mejores” que las soluciones de clúster con valores más pequeños [13].

M. Davies-Bouldin Index

El índice de Davies-Bouldin se calcula utilizando la distancia media entre los centroides de los grupos y la similitud media dentro de cada grupo. Un valor del índice de más bajo indica una mejor calidad del clustering, donde los grupos están más separados y más cohesivos. Por lo tanto, se busca minimizar este índice al seleccionar el número óptimo de grupos o al comparar diferentes algoritmos de clustering [14].

III. Implementación

A. Metodología de trabajo

Previamente a la realización del notebook en el que se ha llevado a cabo el estudio, se predefinió una metodología de trabajo con el objetivo de facilitar el desarrollo de éste. Se estableció el uso de GitHub para la gestión del proyecto, una política de ramificación basada en entorno, una política de codificación, una política de commits y una política de pull requests. Además, para la realización del artículo se acordó el uso de la herramienta *Overleaf*.

Posteriormente, se hizo un reparto de tareas equitativo a través de tableros Scrum de GitHub para las tareas de código, de forma que éstas pasaban primero por una fase de espera para ser realizadas, una fase de realización y por último una fase de revisión por parte del otro desarrollador antes de dar la tarea por finalizada [15].

B. Implementación en Python

En esta sección, describiremos cómo se han implementado diferentes algoritmos de clustering y métricas de calidad en el lenguaje de programación Python.

Tanto las métricas como los primeros ocho algoritmos han sido implementados mediante el uso de la biblioteca *scikit-learn*, que es una popular librería de aprendizaje automático que proporciona implementaciones eficientes de una amplia gama de algoritmos de clustering y de métricas de calidad.

Sin embargo, los algoritmos Partition Around Medoids (PAM) y UPGMA han sido implementados por nosotros desde cero. Para ello, nos hemos apoyado principalmente en la biblioteca de Python *numpy*, que proporciona estructuras de datos eficientes y funciones para manipular matrices y vectores, lo que la convierte en una herramienta muy útil dentro del campo del aprendizaje automático.

1) PAM: Para la implementación de PAM se ha creado una función que recibe como parámetros el dataset y el número de clusters deseado. Opcionalmente, se puede indicar el número máximo de iteraciones, situado por defecto en 100. La función realiza el proceso anteriormente especificado del algoritmo PAM, siguiendo una serie de pasos: primero se inicializan aleatoriamente los medoides, y luego se realizan iteraciones en las que se asigna cada punto al medoide más cercano y se van actualizando los mededides hasta que se llegue al número máximo de iteraciones o hasta que los medoides de dos iteraciones consecutivas sean idénticos. Finalmente, se devuelve una lista con las etiquetas asigndas a cada uno de los datos.

2) UPGMA: Inicialmente la implementación supuso un reto de mayor complejidad al tratarse de un algoritmo en el que hay que actualizar la matriz de distancias en cada iteración. En nuestro caso, hemos hecho uso de la distancia euclídea para la distancia entre puntos y de la media para calcular la distancia de un nuevo grupo. Teniendo en cuenta lo anterior, nuestra función se compone de tres bloques:

- Primero, se inicializa una lista de listas llamada clusters, donde cada elemento representa un clúster inicial que contiene un solo objeto. A medida que el algoritmo avanza, los clústeres se fusionan y se actualiza la matriz de distancias.
- En cada iteración del bucle while, se encuentra la distancia mínima en la matriz de distancias y se identifican los índices de los clústeres asociados a esa distancia. Los clústeres se fusionan en un nuevo clúster y se actualizan los índices y la matriz de distancias.
- El bucle continúa hasta que el número de clústeres alcanza el hiperparámetro k. Luego, se asignan etiquetas a cada objeto basándose en los clústeres resultantes y se devuelven las etiquetas ordenadas.

C. Selección de hiperparámetros

En este apartado se justificarán los distintos rangos de hiperparámetros que han sido escogidos para testear cada algoritmo de clustering y realizar la comparativa.

1) Hiperparámetros de BIRCH: El hiperparámetro Branching Factor (B) lo hemos variado entre tres valores diferentes: 10, 50, y 100, consiguiendo de esta manera obtener resultados con un mayor número de clusters a medida que aumenta el valor de B. Por otro lado, al parámetro Threshold (T) le hemos

dado otros tres valores: 0.2, 0.5 y 0.8, de forma que probamos el algoritmo con diferentes tipos de umbrales de más a menos restrictivo, respectivamente. Con mayores valores de T hemos obtenido clusters más grandes.

2) Hiperparámetros de DBSCAN: El hiperparámetro *Eps* lo hemos variado entre cuatro valores diferentes: 0.15, 0.2, 0.25 y 0.6, consiguiendo de esta manera obtener clusters con vértices con distinta distancia de separación. Por otro lado, al parámetro *MinPts* le hemos dado tres valores: 10, 20 y 100, de forma que probamos el algoritmo con distintas densidades de la vecindad. Con mayores valores de *Eps* y *MinPts* hemos obtenido clusters con mayor tamaño.

3) Hiperparámetros de Mean Shift: Al hiperparámetro quantile (Q) le hemos otorgado los valores 0.1, 0.2 y 0.3, de manera que se obtienen diferentes anchos de banda, que son más amplios a medida que aumenta Q. Con el fin de probar la influencia del parámetro n samples (N), le hemos asignado los valores 20, 500 y 1000. N indica el número de ejemplos que se toman para estimar el ancho de banda, y a vista de los resultados hemos percibido que en una amplia mayoría de casos los agrupamientos con una N más alta han sido mejores, tal y como era esperado.

4) Hiperparámetros de OPTICS: En este caso la selección de hiperparámetros ha sido escogida buscando valores interesantes en las métricas, además de buscar agrupaciones lógicas. El hiperparámetro *Eps* lo hemos variado entre 0.15 y 0.8 y el parámetro *MinPts* valores entre 5 y 100.

5) Hiperparámetros de k-means, Mini-Batch K-means, Spectral Clustering, Mixture of Gaussians, PAM y UPGMA: Todos estos algoritmos tienen un factor común: su único hiperparámetro es el número de clusters deseado (K). Al estar realizando un estudio comparativo de estos algoritmos, hemos considerado que lo más apropiado era utilizar el mismo rango de valores de K para todos estos algoritmos. El rango ha sido establecido entre 2 y 10, ya que son valores que nos permiten realizar pruebas lo suficientemente variadas y obtener conclusiones claras.

IV. Pruebas y experimentación

En esta sección se detallarán tanto los experimentos realizados como los resultados conseguidos:

A. Métodos de evaluación de algoritmos

En este apartado se explicarán los métodos implementados para la realización de las pruebas y de experimentos.

- *Get-best-score* El método *get-best-score* implementa un enfoque para encontrar el algoritmo y los hiperparámetros que mejor se ajustan a un conjunto de datos según una métrica de calidad especificada. El método utiliza varios métodos auxiliares para realizar esta tarea:
  - En primer lugar, el método *algorithms-hyperparameters* define una estructura de datos que contiene los algoritmos de clustering disponibles y los posibles valores de hiperparámetros asociados a cada algoritmo. Estos valores son predefinidos y agrupados en una lista o en una combinación de listas, dependiendo del algoritmo.
  - El método *run-clustering-algorithm* se utiliza para ejecutar un algoritmo de clustering específico con los hiperparámetros proporcionados. Dependiendo del algoritmo, se utilizan diferentes argumentos para la función *clustering*, que es responsable de aplicar el algoritmo. También se calcula una métrica de calidad utilizando la función *get-quality-metric* y se puede visualizar el resultado si se establece el argumento *visualize* en *True*.
  - El método *select-solution* se utiliza para seleccionar la mejor solución según la métrica de calidad especificada. Se compara la puntuación de cada solución en la lista *solution* y se selecciona la mejor según el criterio especificado. La mejor solución se imprime en la consola.

  Finalmente, el método *get-best-score* itera a través de los algoritmos y los hiperparámetros definidos en *algorithms-hyperparameters*. Para cada combinación de algoritmo e hiperparámetros, se ejecuta el método *run-clustering-algorithm* y se guarda el resultado en un diccionario llamado *solutions*. Luego, se llama al método *select-solution* para obtener la mejor solución de clustering según la métrica de calidad especificada. Al final, se imprime en la consola el algoritmo, la puntuación y los hiperparámetros utilizados en la mejor solución.

- *Get-total-scores:* En resumen, el método *get-total-scores* automatiza el proceso de obtención de los valores normalizados de las métricas de calidad para cada algoritmo de clustering y sus hiperparámetros asociados. Utiliza métodos auxiliares para calcular las puntuaciones de las métricas, encontrar los valores mínimos y máximos, y realizar la normalización de las puntuaciones. La normalización de las métricas de calidad se ha realizado para buscar el algoritmo que tenga la mejor puntuación teniendo en cuenta todas las métricas. Una vez se obtienen las métricas normalizadas, con valores entre 0 y 1, estas se suman, obteniendo el *total score* de un determinado algoritmo con unos hiperparámetros concretos. El valor del *total score* se encontrará entre 0 y 3 siendo mejor cuanto más elevado sea. Adicionalmente, se ha creado un método llamado *get-optimal-algorithm*, que recibiendo el resultado de la función *get-total-scores* obtiene el algoritmo e hiperparámetros con mayor *total score*, es decir, el más óptimo teniendo en cuenta todas las métricas.

B. Experimentos realizados

(Indicando razonadamente la configuración empleada, qué se quiere determinar, y como se ha medido)

1) Mejor algoritmo e hiperparámetros para una métrica: Haciendo uso del método *get-best-score* se ha realizado un experimento en el que se mide el mejor hiperparámetro para un algoritmo de los citados anteriormente, medidos por la métrica dada por parámetro. Como hemos explicado a lo largo del

artículo, las métricas utilizadas son el coeficiente de Silhouette, el índice de Calinski-Harabasz y el índice de Davies-Bouldin.

2) Mejores hiperparámetros para un algoritmo: Se ha realizado un experimento a través del método *get-total-scores* en el que se obtienen los *total scores* para cada combinación de hiperparámetros y algoritmo. Esto puede ser interesante para comparar estos resultados con los del experimento anterior, ya que, además de comparar la calidad de clasificación de los distintos algoritmos para una métrica concreta, se estudiará si los algoritmos que mejor puntúan teniendo en cuenta las tres métricas son los mismos o no.

C. Resultados obtenidos

En este apartado se va mostrar el resultado obtenido en los experimentos mencionados anteriormente, que van a ser analizados en el siguiente apartado.

- Mejor algoritmo e hiperparámetros para una métrica: en esta sección se mencionarán otros resultados aparte del algoritmo e hiperparámetros con mejor puntuación, para ampliar el análisis posteriormente.
  - Silhouette Coefficient: En la tabla I aparecen los resultados proporcionados por el método *get-best-score*. El algoritmo con mejor puntuación es el algoritmo UPGMA con 2 clusters y el algoritmo OPTICS con 0.8 de *Eps* y 5 de *MinPts*.

TABLA I
SILHOUETTE COEFFICIENT

Algoritmo	Hiperparámetros	Puntuación
Birch	10, 0.8	0.399
DBSCAN	0.6, 100	0.249
K-means	3	0.489
Mini-batch K-means	3	0.487
Mean Shift	0.2, 1000	0.487
OPTICS	0.8, 5	0.534
Spectral Clustering	2	0.489
GMM	2	0.42
PAM	2	0.485
UPGMA	2	0.534

  - Calinski-Harabasz Index: En el caso de Calinski-Harabasz, el algoritmo con mejor puntuación es el algoritmo K-means con 5 clusters. Además, la tabla II muestra el resto de resultados proporcionados.

TABLA II
CALINSKI-HARABASZ INDEX

Algoritmo	Hiperparámetros	Puntuación
Birch	100, 0.2	1409
DBSCAN	0.25, 20	340
K-means	5	1665
Mini-batch K-means	5	1634
Mean Shift	0.1, 20	1105
OPTICS	0.31, 40	386
Spectral Clustering	3	1593
GMM	8	1497
PAM	5	1537
UPGMA	4	976

  - Davies-Bouldin Index: Por último, en la tabla III se muestran los resultados usando como parámetro ésta métrica, siendo los algoritmos con mejor puntuación el OPTICS con 0.8 de *Eps* y 5 de *MinPts* y el UPGMA con 2 clusters.

TABLA III
DAVIES-BOULDIN INDEX

Algoritmo	Hiperparámetros	Puntuación
Birch	10, 0.8	0.708
DBSCAN	0.15, 10	3.296
K-means	3	0.666
Mini-batch K-means	3	0.666
Mean Shift	0.3, 1000	0.595
OPTICS	0.8, 5	0.347
Spectral Clustering	3	0.665
GMM	10	0.842
PAM	3	0.678
UPGMA	2	0.347

- Mejores hiperparámetros para un algortimo: para exponer los resultados de este experimento se indican los mejores hiperparámetros obtenidos para cada uno de los algoritmos. Para obtener estos resultados se ha hecho uso de la función *get-total-scores*, cuyo funcionamiento ha sido descrito anteriormente, y de la función *plot-optimal*, que realiza una representación gráfica de los *total scores* obtenidos y nos indica cuál es el mejor.
  - BIRCH: el mayor *total score* obtenido ha sido 2.6233, utilizando los hiperparámetros branching factor con valor 100 y threshold con valor 0.8.
  - DBSCAN: el mayor *total score* obtenido ha sido 1.2667, utilizando los hiperparámetros Eps con valor 0.6 y MinPts con valor 100.
  - K-means: el mayor *total score* obtenido ha sido 2.8611, utilizando el hiperparámetro k con valor 3.
  - Mini-batch K-means: el mayor *total score* obtenido ha sido 2.8462, utilizando el hiperparámetro k con valor 3.
  - Mean Shift: el mayor *total score* obtenido ha sido 2.4041, utilizando los hiperparámetros quantile con valor 0.1 y n samples con valor 200.
  - OPTICS: el mayor *total score* obtenido ha sido 2.0, utilizando los hiperparámetros Eps con valor 0.8 y MinPts con valor 10.
  - Spectral Clustering: el mayor *total score* obtenido ha sido 2.8588, utilizando el hiperparámetro k con valor 3.
  - Mixture of Gaussians: el mayor *total score* obtenido ha sido 2.6437, utilizando el hiperparámetro k con valor 9.
  - Partition around medoids: el mayor *total score* obtenido ha sido 2.7919, utilizando el hiperparámetro k con valor 3.
  - UPGMA: el mayor *total score* obtenido ha sido 2.4792, utilizando el hiperparámetro k con valor 4.

A vista de los resultados expuestos, puede observarse que el algoritmo más óptimo para nuestro conjunto de datos es k-means con k=3, ya que es el que ha sumado el *total score* más alto.

D. Análisis de los resultados

De forma previa a realizar el análisis se presenta gráficamente el conjunto de datos que ha sido empleado para llevar a cabo este estudio. Se trata de un dataset que contiene 1000 elementos, cada uno de ellos con tres características. Esto nos permite representarlos mediante el siguiente gráfico en tres dimensiones:

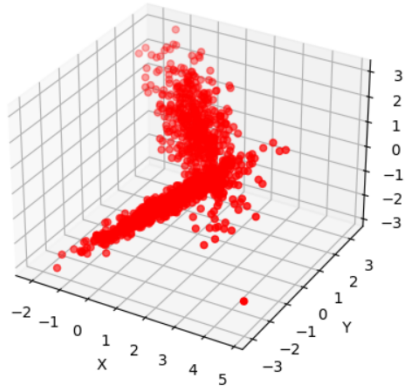

Fig. 1. Gráfico 3D con los elementos del dataset.

Se puede observar que el conjunto de datos, que ha sido generado aleatoriamente, resulta bastante apropiado para el caso de estudio que estamos planteando en este artículo, ya que representa un estado intermedio entre lo que sería una única nube de puntos (la cual difícilmente podría dividirse en clusters de forma eficiente), y una serie de pequeñas nubes (que serían fácilmente divisibles). Es importante tener en cuenta la forma del conjunto de datos, ya que las conclusiones obtenidas dependerán de ella en gran medida.

En primer lugar, cabe destacar que, a pesar de que los valores mostrados en la tabla III sean las mejores puntuaciones generadas de cada algoritmo, hay algunas puntuaciones que reflejan que el algortimo tiene una mala puntuación para todos los hiperparámetros utilizados. Es el caso de DBSCAN, que devuelve como mejor valoración 3.296, representando esto una clasificación más dispersa de los clusters por parte de DBSCAN (clusters con poca distancia entre sí). Como podemos apreciar en las tablas generadas en el apartado anterior, DBSCAN es el algoritmo que peores resultados presenta, debido a tener dificultades para encontrar clusters de densidades variables. Otro tipo de algoritmos, como OPTICS, al ser un algoritmo que es más flexible en términos de formas y jerarquías de clusters, resulta más interesante para nuestro conjunto de datos.

Por otro lado, uno de los casos que más llama la atención vistos los resultados es el del algoritmo OPTICS con los hiperparámetros 0.8 y 5. Dicho algoritmo es, junto con UPGMA (k=2), el que obtiene un mejor coeficiente de silueta y un mejor índicie de David-Bouldin. Esto nos podría hacer pensar que se trata de la configuración idónea para analizar nuestro conjunto de datos. Sin embargo, su *total score* tiene un valor de 2.0, lo que lo hace estar lejos de los mejores algoritmos en cuanto a la puntuación total. Esto se debe a que en la métrica restante, el índice de Calinski-Harabasz, obtiene una puntuación muy baja. Esto es un ejemplo claro de cómo el método diseñado en el que se tienen en cuenta las tres métricas nos ha permitido detectar que un algoritmo que parecía idóneo realmente no lo era tanto.

Además, cada métrica tiene un objetivo de medición distinto y por lo tanto obtener una puntuación mejor va a tener distintos significados en caso de la métrica utilizada. Por ello, el algoritmo e hiperparámetros que ha obtenido mejor puntuación en una métrica en concreto también es el que más se ajusta al objetivo de dicha métrica. En el caso de la métrica Calinski-Harabasz, el algoritmo e hiperparámetros con mejor puntuación es el K-means con 5 clusters. Esto no nos asegura que sea el mejor algoritmo y el mejor hiperparámetro para la tarea de clasificación del conjunto de datos, sino que es la clasificación en la que los clusters están más separados entre sí y en la que los puntos del cluster están lo más cerca posible entre ellos.

En relación con el apartado anterior, en la figura 2 se muestra a la izquierda el resultado del algoritmo mencionado anteriormente y a la derecha la solución del modelo de mezcla gaussiana. Ambos algoritmos han sido puntuados positivamente por distintas métricas; sin embargo, el GMM para 2 clusters muestra una clasificación más adecuada ya que, el conjunto de datos puede interpretarse como la unión de dos gaussianas.

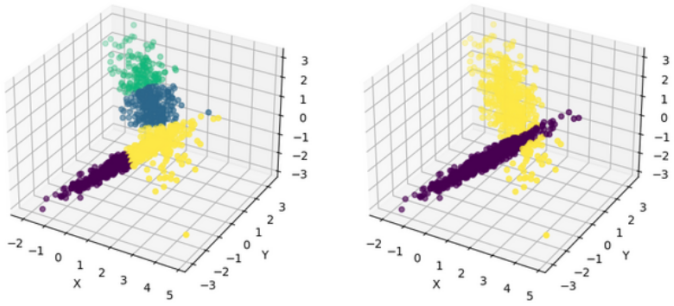

Fig. 2. Comparación de los gráficos de K-means con k=5 y GMM con k=2

El método de evaluación del *total score* puede no ser apropiado si tenemos un objetivo claramente definido, o si ya sabemos qué nos interesa más medir. Esto se debe a que si, por ejemplo, tenemos claro que en un contexto determinado nos va a interesar tener en cuenta únicamente el coeficiente de silueta, no tendríamos por qué evaluar simultáneamente el resto de métricas, sino que lo correcto sería evaluar únicamente el coeficiente de silueta. En otras palabras, el método de evaluación del *total score* está pensado para un caso general en el qué no se tiene un objetivo concreto y se quiere comprobar

qué combinación de algoritmo e hiperparámetros tiene más posibilidades de ser apropiada.

Como conclusión del análisis, según nuestro método de búsqueda del algoritmo e hiperparámetros con puntuaciones de métricas normalizadas, el algoritmo que cumple mejor con todas las métricas de calidad es el K-means con 3 clusters, mostrado en la figura 3. Por último, consideramos que el coeficiente de silueta ha resultado una métrica muy interesante para nuestro conjunto de datos ya que, es una métrica que evalúa la calidad de clasificación del algoritmo, de forma que aquellos algoritmos que han realizado una agrupación por gaussianas o en una segmentación de dichas gaussianas, han obtenido una mejor puntuación.

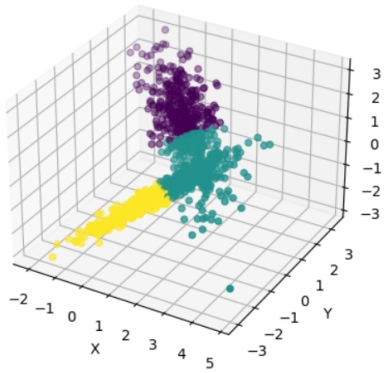

Fig. 3. Gráfico 3D K-means con k=3

V. Conclusiones

En este artículo, se ha investigado y evaluado el desempeño de diferentes combinaciones de algoritmos de clustering e hiperparámetros, empleando una serie de métricas de calidad. Inicialmente, se ha tratado de proporcionar un contexto apropiado con el objetivo de establecer las bases teóricas sobre las cuales se sostiene el estudio. Tras ello, se ha explicado todo el proceso de implementación que ha sido llevado a cabo para preparar la fase de experimentación. Una vez hecho esto, ya se ha podido comenzar a hablar más concretamente de las pruebas y experimentos realizados, analizando los resultados obtenidos y llegando a conclusiones que se sustentan en todo lo anterior.

A nivel de conclusiones, podemos destacar que si queremos evaluar la capacidad global de realizar un buen clustering de un algoritmo no podemos centrarnos únicamente en el resultado que pueda obtener dicho algoritmo en una métrica, ya que cada una de las métricas se centra en medir aspectos diferentes. Por este motivo, consideramos que el método de evaluación propuesto es una buena aproximación a averiguar cuál es la combinación de algoritmo e hiperparámetros óptima, ya que tiene en cuenta las diferentes métricas. Sin embargo, como ya se ha analizado previamente, es un método que no es apropiado en el caso en el que sepamos qué métricas concretas nos interesa tener en cuenta.

Para finalizar, en cuantos a posibles aspetos de mejora, se han identificado varias áreas que podrían beneficiarse de investigaciones adicionales. La primera de ellas está relacionada con la eficiencia computacional, ya que al tener que comparar un elevado número de algoritmos con diversos valores de hiperparámetros se han tenido que realizar muchas ejecuciones de estos algoritmos, lo que lo convierte en un método ineficiente en caso de que se requiera obtener resultados de forma rápida. Por otro lado, se lograría aumentar la fiabilidad del método de evaluación que tiene en cuenta las tres métricas de calidad si se añadieran otros tipos de métricas relacionadas con el clustering, como el índice de Rand o el índice de Fowlkes–Mallows. Además, un aspecto que consideramos importante de este artículo es la realización de un estudio de los hiperparámetros antes del uso de las métricas, y creemos que se podría hacer más hincapié más en este asunto. En este mismo sentido, también sería interesante incluir pruebas con nuevos valores de hiperparámetros o añadir otros algoritmos de clustering, con el fin de realizar una comparación más completa.

Referencias

[1] D. Pascual, F. Pla and S. Sánchez, 2007. “Algoritmos de agrupamiento”. https://www.academia.edu/8412241/Algoritmos$_de_agrupamiento$

[2] Miguel Garre, Juan José Cuadrado, Miguel A. Sicilia, Daniel Rodríguez, Ricardo Rejas. ETS Ingeniería Informática, Universidad de Alcalá de Henares, 2007, “Comparación de diferentes algoritmos de clustering en la estimación de coste en el desarrollo de software ”. https://www.redalyc.org/pdf/922/92230103.pdf

[3] Maklin, C. (1 de julio de 2019). BIRCH Clustering Algorithm Example In Python. Medium. https://towardsdatascience.com/machine-learning-birch-clustering-algorithm-clearly-explained-fb9838cbeed9

[4] Charu C. Aggarwal and Chandan K. Reddy,”Data Clustering: Algorithms and Applications,” Chapman and Hall/CRC, pp. 113–114, August 2013. https://people.cs.vt.edu/ reddy/papers/DCBOOK.pdf

[5] Ramírez, L. (5 de enero de 2023). Algoritmo k-means: ¿Qué es y cómo funciona?. IEBS School. https://www.iebschool.com/blog/algoritmo-k-means-que-es-y-como-funciona-big-data/

[6] Aman Kharwal. (10 de septiembre de 2021). Mini-batch K-means Clustering in Machine Learning. https://thecleverprogrammer.com/2021/09/10/mini-batch-k-means-clustering-in-machine-learning

[7] Yufeng. (22 de febrero de 2022). Understanding Mean Shift Clustering and Implementation with Python. Medium. https://towardsdatascience.com/understanding-mean-shift-clustering-and-implementation-with-python-6d5809a2ac40

[8] Charu C. Aggarwal and Chandan K. Reddy,”Data Clustering: Algorithms and Applications,” Chapman and Hall/CRC, pp. 116–117, August 2013. https://people.cs.vt.edu/ reddy/papers/DCBOOK.pdf

[9] Chatterjee, M. (24 de octubre de 2022). Introduction to Spectral Clustering. Great Learning. https://www.mygreatlearning.com/blog/introduction-to-spectral-clustering/

[10] Charu C. Aggarwal and Chandan K. Reddy,”Data Clustering: Algorithms and Applications,” Chapman and Hall/CRC, pp. 64–67, August 2013. https://people.cs.vt.edu/ reddy/papers/DCBOOK.pdf

[11] Helm, M. (20 de agosto de 2021). A deep dive into partitioning around medoids. Medium. https://towardsdatas¡cience.com/a-deep-dive-into-partitioning-around-medoids-a77d9b888881

[12] Scikit-learn documentation: Silhouette coefficient. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette$_score.html$

[13] Documentación Herramienta de diagnóstico de K-centroides. https://help.alteryx.com/2020.2/es/K-Centroids$_Diagnostics$

[14] Scikit-learn documentation: Davies-Bouldin score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.davies$_bouldin_score.html$

[15] Álvaro Bernal and Álvaro González, GitHub Repository, Clustering algorithms. https://github.com/alvgonfri/clustering$_algorithms$