



CASO PRÁCTICO 2 - INFORME

ALGORITMOS DE CLUSTERING

OBJETIVO

RESOLVER PROBLEMAS DE CLUSTERING

REALIZADO POR

💻 Andrade Peñafiel Mateo

💻 Huilca Revelo Victor German

💻 Espinoza Bone Jose Manuel

MATERIA

MÉTODOS ANALÍTICOS Y DE OPTIMIZACIÓN
APLICADOS A LA INTELIGENCIA ARTIFICIAL

Introducción

En este segundo caso práctico vamos a resolver problemas de clustering. Esta herramienta nos da la capacidad de resolver fácilmente problemas sencillos de un modo muy visual.

En este caso práctico se ha aplicado los siguientes algoritmos:

- DBSCAN
- K-Means
- HierarchicalClustering

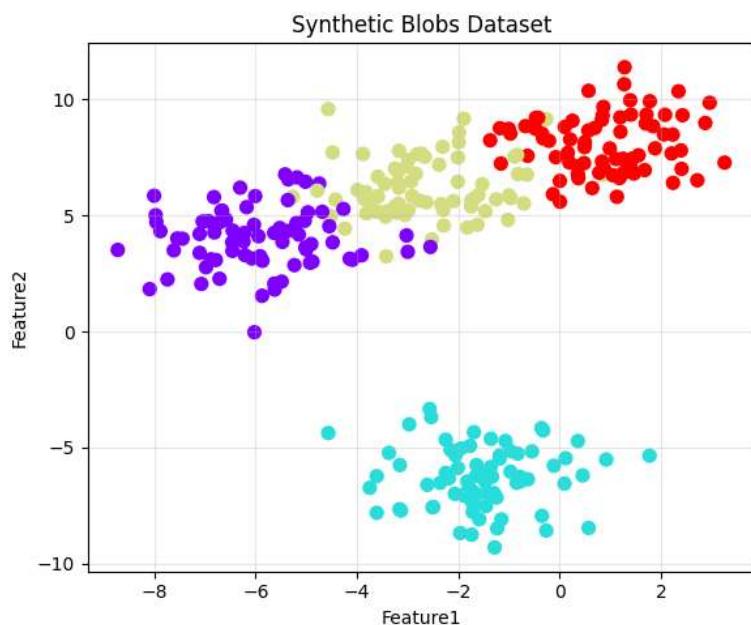
Son tres herramientas que nos ayudan a crear grupos o clústers. Cada una tiene sus hiper parámetros para ser aplicada a un dataset.

Se ha tenido tres fases las cuales se explican a continuación.

FASE I - Generación y representación de datos

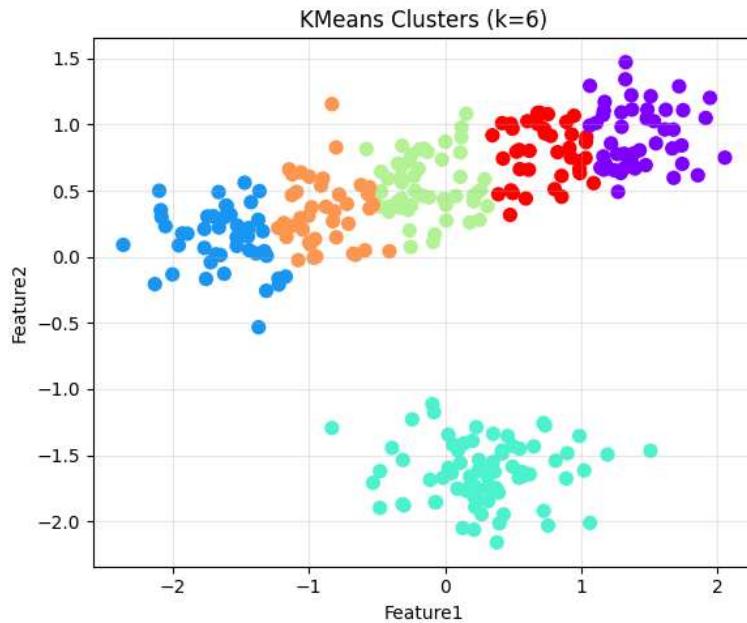
Siguiendo la misma estrategia del caso práctico 1 se ha creado un Python notebook en el cual primeramente se generan datos sintéticos para esta práctica usando la función [CreateSyntheticBlobsDataset\(\)](#), esta función hace uso de una función de sklearn para generar blobs o grupos de datos, se utiliza en esta función como semilla(random state) la suma de cada uno de los ordinales ascii de las iniciales de los nombres de los integrantes.

Se tiene adicionalmente la función [PlotSyntheticBlobsDataset\(\)](#) para graficar este dataset.



Como estos modelos son susceptibles a los valores muy grandes, procedemos a escalar los datos el scaler utilizado fue StandardScaler. Una vez escalados los datos tenemos todo listo para aplicar el primer modelo que es Kmeans.

Aplicamos Kmeans y establecemos el números de clústeres en 6 de manera directa y mostramos los resultados en un gráfico.



Como se observa en el gráfico Kmeans de manera bien visible y exacta nos muestra 6 grupos de datos en la gráfica. Visualmente se ven correctos pero podemos buscar el mejor valor para k.

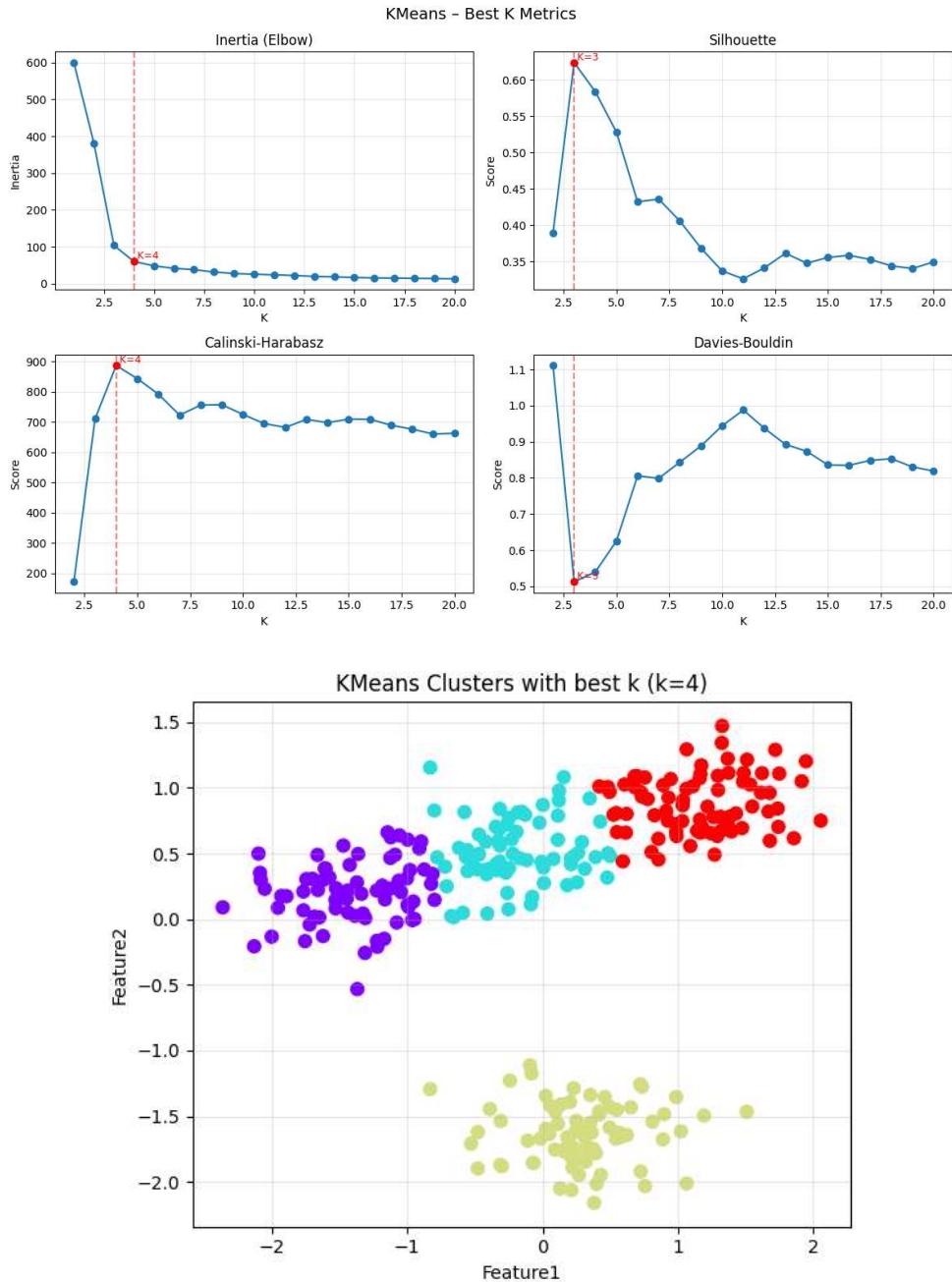
FASE II - Número óptimo de clusters con técnica del codo

Para calcular el valor óptimo para k utilizando la técnica del codo se utilizó la library Kneed. En la presente práctica se tiene 4 formas de obtener el valor de k utilizando diferentes métricas:

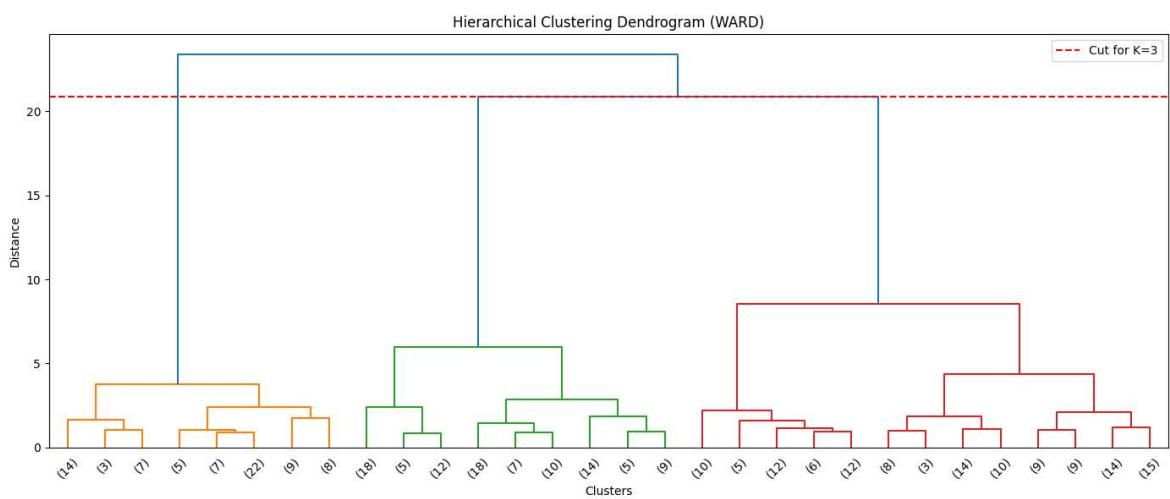
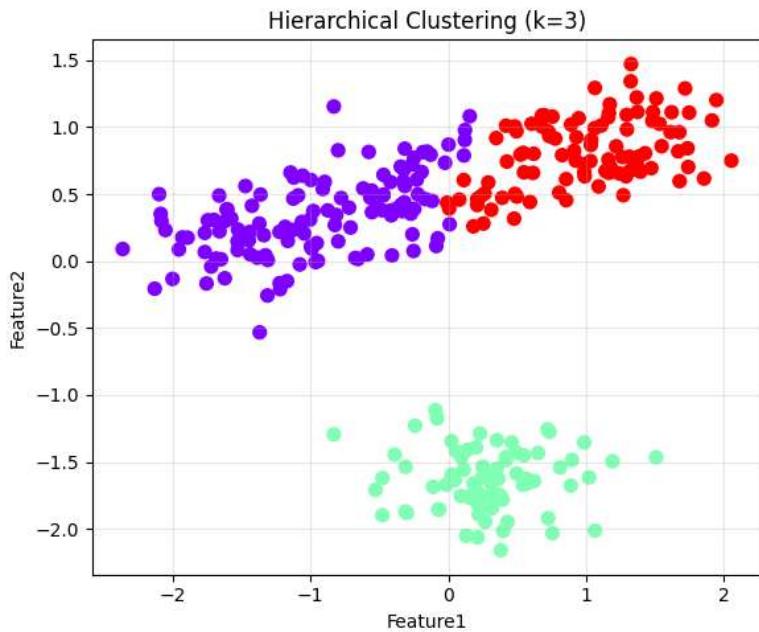
- Usando las inercias con la técnica del codo.
- Score de siluetas
- Score Davies-Bouldin.
- Score Calinski and Harabasz.

Todo lo anterior mediante la función que retorna el mejor k para cada score. Se retorna la mediana entre los valores válidos de siluetas, Calinski and Harabasz, Davies-Bouldin, método del codo y por último si son None todos se retorna 2 por default.

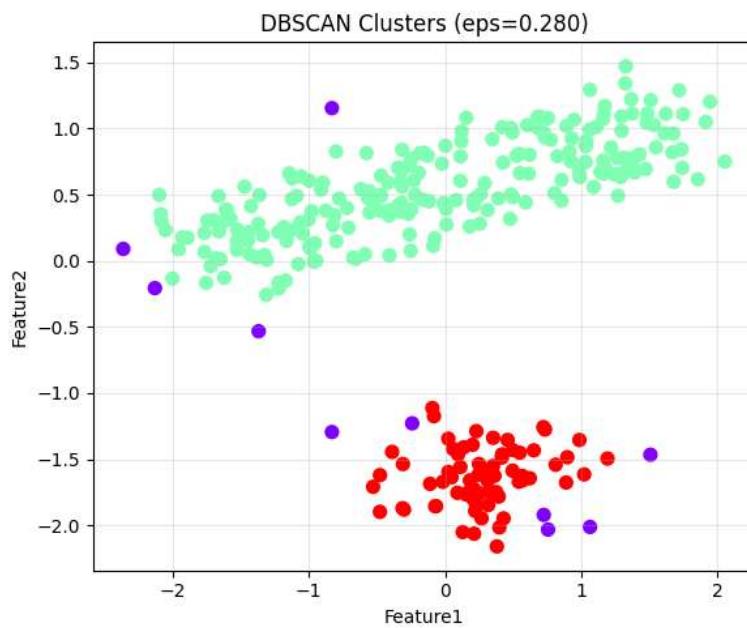
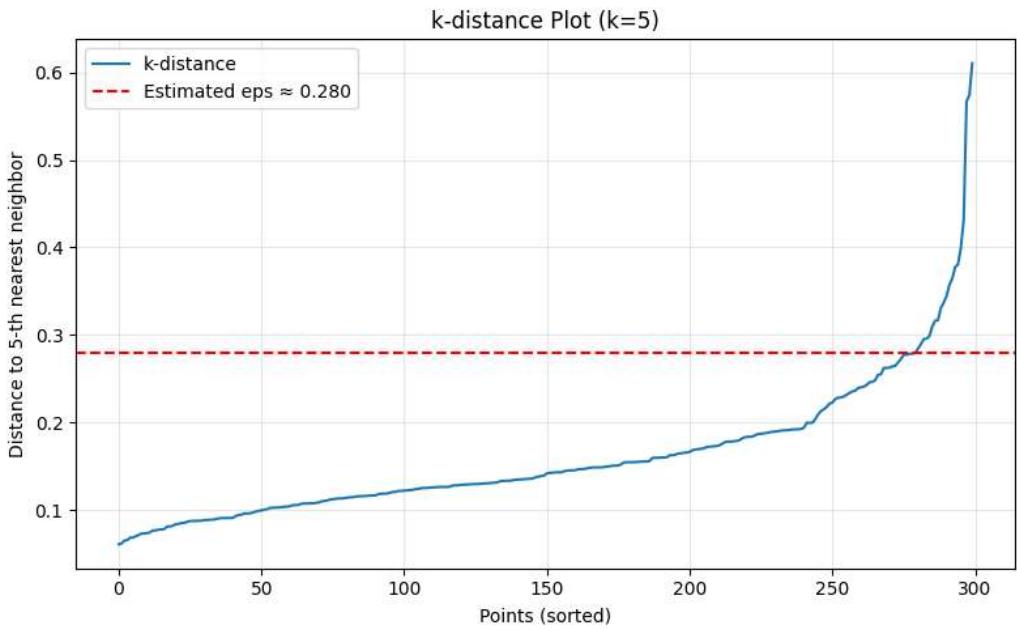
Adicional opcional se puede mostrar la gráfica correspondiente por cada score/métrica. También se puede utilizar a conveniencia del desarrollador/científico de datos el k de una métrica específica todo esto mediante la función [EvaluateKmeansBestKs\(\)](#).



Una función similar existe para Hierarchical clustering que es [EvaluateHierarchicalBestKs\(\)](#) la diferencia es que esta función no tiene el método del codo, ya que no aplica. Se tiene la opción de mostrar el dendrogram para ver los clusters.



Para el algoritmo DBSCAN es complejo obtener el valor de eps pero existe una forma aproximada usando la library Kneed y detectando el quiebre de todas las distancias al vecino más cercano último. La función tiene también la opción de mostrar el gráfico de distancias para validar visualmente el resultado la función es [EvaluateDBScanBestEps\(\)](#).



Para esta fase utilizando el método del codo el valor de k óptimo es 4, según el cálculo y viendo la gráfica. Similar a los otros scores 3 con siluetas, 4 con Calinski and Harabasz, 3 con Davies-Bouldin.

FASE III - Generación y representación de datos

Para esta fase se calculó los bestK para HC, Kmeans y bestEps para DBSCAN. Luego se aplicó cada uno de los modelos a los datos escalados con standard scaler. Y se graficó los 3 resultados adicional las gráficas de distancias para DBSCAN, los scores para KMeans y el dendrogram para Hierarchical clustering.

Para DBSCAN se obtuvo que el mejor EPS es 0.28 dando como resultado dos clusters y varios puntos como ruido(-1).

Para Kmeans utilizando siluetas se tiene un k de 4, dando como resultado 4 clusters.

Para HC usando siluetas y visualizando en el dendrogram se obtiene 3 clusters.

CONCLUSIONES

El algoritmo Kmeans es fácil de aplicar y muy efectivo. Los resultados son similares en las diferentes métricas para el valor de k.

El algoritmo DBSCAN es más complejo de aplicar ya que obtener eps no es tan sencillo y la propuesta de esta práctica es una aproximación. Se debe probar con otros parámetros para número mínimo de vecinos y eps.

El algoritmo HC mediante el dendrograma es sencillo visualizar los clusters. Se puede calcular el valor óptimo de k.

RECOMENDACIONES

Siempre visualizar el gráfico de distancias para DBSCAN y visualizar la gráfica para tener mejor certeza y corroborar el valor de eps obtenido en la función que lo calcula.

El método del codo es muy usado en Kmeans, si es recomendable utilizar las otras opciones de scores.

Siempre utilizar las gráficas antes de decidirse por el valor que retorna la función. El visualizar el gráfico con criterio fuerte, refuerza el valor obtenido en las diferentes gráficas para los diferentes modelos ya sea k o eps.

ANEXO 1. CÓDIGO FUENTE.

El código fuente se puede encontrar en el siguiente repo.

<https://github.com/UIDE-Tareas/2-Ciclo-Vida-Inteligencia-Artificial-Tarea2>

Revisar a detalle toda la implementación completa ya que contiene mucho código para este informe.