Caso Práctico 2

Objetivo y alcance del trabajo

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.

Recuerda:

- Se debe entregar tanto el fichero **.ipynb** como un informe **.pdf**.
- Al puntuar, se valora positivamente contenido extra que aporte valor a la práctica, y es necesario para obtener la máxima nota posible.
- En todos los apartados lo principal es la interpretación que hace el alumno de los resultados, por ejemplo: en caso de que se pida calcular un valor, no limitarse a calcularlo, sino explicar qué significa y qué implica en el contexto de la práctica. Si no se razonan los resultados se valorará de manera reducida.
- El trabajo es grupal, todos los integrantes deben participar para obtener nota. En caso contrario, se debe contactar con el tutor a través de mensaje privado por la plataforma para resolver el problema.

Fase I - Generación y representación de datos

Siguiendo la orientación dada en la primera práctica, genera un cuadernillo en blanco y verifica que funcione correctamente. Utilizando el siguiente código, se pueden generar datos sintéticos:

```
from matplotlib import pyplot as plt
from matplotlib import numpy as np
from sklearn.datasets import make_blobs

estado = ord('A') # cada grupo debe cambiarlo por su inicial
# generar datos aleatorios y representarlos
X, y = make_blobs(n_samples = 300, centers = 4, random_state = estado,
cluster_std = 1.2)
plt.scatter(X[: , 0], X[: , 1], s = 50);
```

Utilizar el código anterior, pero cambiando la letra acorde a la inicial del nombre de un integrante del grupo para generar datos y entrenar un modelo de K-Means usando el set de datos obtenido especificando K = 6, representarlo y comentar los resultados.

```
numero_clusters = 6
from sklearn.cluster import KMeans
est = KMeans(numero_clusters, n_init=10)
est.fit(X)
y_kmeans = est.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='rainbow');
```

Fase II - Número óptimo de clusters con técnica del codo

La suma de distancias de un cluster con respecto a su centroide (inercia) se puede calcular con la siguiente función (donde *est* se corresponde con el modelo anteriormente entrenado) Podemos calcular la inercia de los clusters usando *est.inertia_*

Recuerda que la técnica del codo calcula la inercia para un número de clusters diferentes, y el punto en que la gráfica hace un codo se considera el número óptimo de clusters, ya que incrementar más el número produce una reducción poco significativa de la inercia.

Utilizar toda esta información para:

- Obtener una gráfica sobre la que aplicar la técnica del codo
- Interpretar esa gráfica y elegir un K óptimo
- Agrupar los datos con K-Means utilizando el K obtenido y representar los clusters
- Comparar los resultados obtenidos frente a K=6 obtenidos en la fase I

Fase III - Comparativa con otros modelos

Cada grupo debe elegir un algoritmo de entre los vistos en clase (**Clustering Jerárquico** *[1]* **o DBSCAN***[2]*). Se debe utilizar el algoritmo elegido para agrupar los elementos de la fase anterior y comparar los resultados obtenidos entre ambos algoritmos.

> [1] Referencia para Clustering Jerárquico:
> https://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering
> [2] Referencia para DBSCAN:
> https://scikit-learn.org/stable/modules/clustering.html#dbscan