Documentación: Implementación de K-Means desde Cero

Introducción

Este proyecto implementa el algoritmo de K-Means desde cero sin usar *sklearn.cluster.KMeans.* Se utiliza *make_blobs* para generar un dataset de prueba.

El algoritmo agrupa los datos en clusters y usa el método del codo para encontrar el mejor número de clusters (k).

Requisitos
Para ejecutar el código, instala las librerías necesarias con:

pip install numpy matplotlib

Cómo Funciona el Algoritmo

Generación del dataset:
Se crean puntos de datos con make_blobs().

Cálculo de distancias:
Se usa la distancia euclídea para asignar cada punto al cluster más cercano.

Inicialización de centroides:
Se puede elegir entre:

Aleatoria: Se generan valores dentro del rango de los datos.
Desde puntos existentes: Se seleccionan puntos del dataset como centroides iniciales.

Asignación de clusters:
Cada punto se asigna al centroide más cercano.
Si hay empate, se asigna al cluster con menos puntos.

Actualización de centroides:
Se recalculan los centroides como el promedio de los puntos asignados a cada cluster.

Criterio de parada:
Se repiten los pasos hasta que los centroides no cambian o se alcanza un número máximo de iteraciones (max_iter).

Método del codo:
Se prueba con diferentes valores de k y se mide la Suma de Errores Cuadráticos (SSE) para determinar el mejor número de clusters.

Visualización
Se grafican los clusters con diferentes colores.
Los centroides finales se marcan con una X roja.

Conclusión
Este código implementa K-Means desde cero con:
Distancia euclídea.
Diferentes métodos de inicialización.
Equilibrio de tamaño en caso de empate.
Método del codo para seleccionar k.
Visualización de los clusters.

Es una solución simple y funcional para aplicar K-Means sin librerías avanzadas.