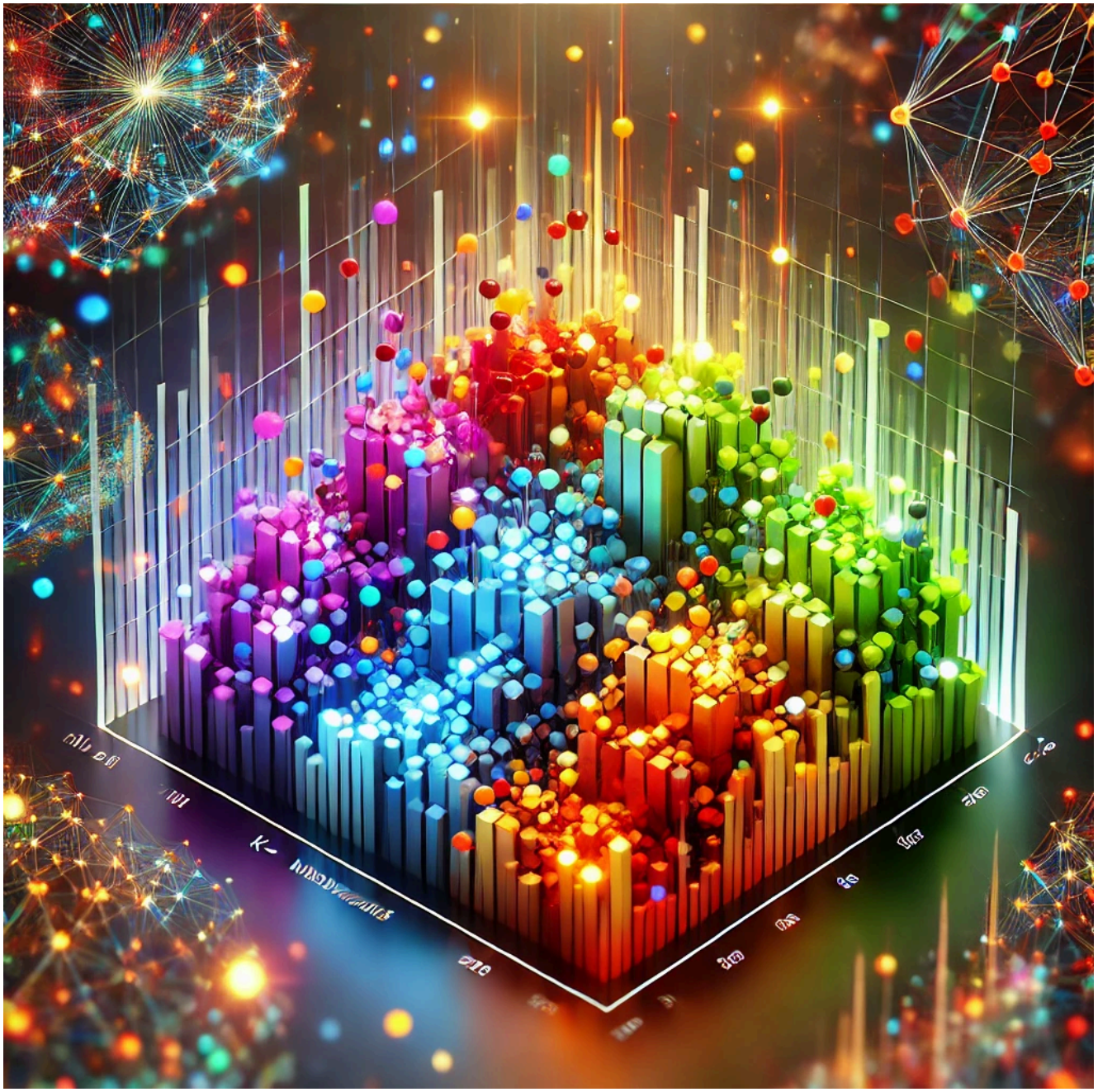
Clustering con K-Means

Tomás Rial Costa / Github: https://github.com/tomasrial46
Adán González Rodríguez / Github: https://github.com/AdanAgr



Implementación del Algoritmo K-Means

Funciones Principales

`Kmeans(RandomPoints=False, iteracion_max=50, k=4)`

Esta función implementa el algoritmo de K-Means.

Parámetros:

- *`RandomPoints`* (bool): Si es *`True`*, los centroides iniciales se seleccionan aleatoriamente en el rango de los datos. Si es *`False`*, se seleccionan aleatoriamente de los puntos existentes.
- *`iteracion_max`* (int): Número máximo de iteraciones.
- *`k`* (int): Número de clusters.

Proceso:

1. Se inicializan los centroides según la opción elegida.
2. Se asigna cada punto al cluster más cercano según la distancia euclidiana.
3. Se recalculan los centroides como el promedio de los puntos en cada cluster.
4. Si los centroides no cambian entre iteraciones, el proceso termina.
5. Se grafican los clusters en cada iteración.

`DistEuclidiana(a, b, num_centroides)`

Calcula la distancia euclidiana entre un punto `a` y cada centroide en `b`.

Parámetros:

- `a` (list): Coordenadas del punto.
- `b` (list): Lista de centroides.
- *`num_centroides`* (list): Contador de puntos asignados a cada cluster.

Retorno:

- `int`: Índice del centroide más cercano.

`Graf(k, clusters, centroides)`

Grafica la distribución de puntos en los clusters con colores diferentes y los centroides con una "X" negra.

Parámetros:

- `k` (int): Número de clusters.
- `clusters` (list): Lista de listas con los puntos de cada cluster.
- `centroides` (list): Lista de centroides actuales.

Problemas encontrados:

Al terminar el código y hacer pruebas, nos dimos cuenta que había ocasiones en las que el código erraba ya que no se asignaban puntos a un cluster específico, por lo que tuvimos que añadir una condición más que añade el centroide del cluster vacío aleatoriamente para que no salte el error.