Esta carpeta contiene una implementación del algoritmo de K-Means clustering construido desde cero usando Python. El proyecto demuestra conceptos clave del aprendizaje no supervisado mediante la agrupación de un conjunto de datos sintéticos y la posterior predicción usando el algoritmo KNN.

Dentro de esta, se puede encontrar el archivo .py del código, así como un pdf con ejemplos del funcionamiento del código.

Para empezar con el código, se inicializa el dataset, que son puntos al azar hechos con la función de make_moons. El número de datos que se generará está definido por el usuario. Se inician definiendo las funciones para realizar K-Means:

- Centroides: Estos van a servir como punto de partida para hacer la clusterización, se generan de manera aleatoria.
- Asignar Clusters: Con base en la distancia de cada punto con los centroides, se elige la menor distancia, esto quiere decir que por cada punto se elige su cluster más cercano, lo cuál hace que se agrupen en los clusters. El número de clusters de igual manera está definida por el usuario.
- Actualizar Centroides: Una vez que todos los puntos están definidos es un solo cluster, se recalcula el centroide de cada uno de estos para que esté en el lugar medio dentro de todos los puntos asignados a su cluster.
- K-Means: Esta última función es donde todo lo anterior se junta y trabajan para hacer la clusterización. Esta función recibe los parámetros: X que representa el número de datos, y que es el número de clusters que ingresó el usuario, además del hiperparámetro de las iteraciones máximas, que serán de 100 para asegurar una buena clasificación en clusters. Dentro de la función hay una condición que verifica si los centroides nuevos y los anteriores son iguales, en caso de que sí, significa que el modelo llego a la solución antes de las 100 iteraciones y el algoritmo se detiene.

De igual manera, se agregan las funciones necesarias para realizar una predicción con KNN de un punto agregado por el usuario y determinar el cluster al que este punto pertenece.

- Distancia Euclidiana: Se ocupará para calcular la distancia entre 2 puntos.
- KNN: Se calcula la distancia entre el punto ingresado y los demás puntos del dataset. Se ordenan de mayor a menor y se seleccionan los más cercanos, en este caso, se hace un KNN con los 3 vecinos más cercanos. Dentro de estos 3 puntos más cercanos, se guardan sus etiquetas que representan su cluster y el que más se repita es el cluster al que se va a asignar este nuevo punto.

Finalizando con todas estas funciones se agrega código para la funcionalidad correcta del código:

- Se incializan colores para los clusters (el máximo de clusters son 10) y se pone el nombre de estos en español para que al final el resultado diga: "Cluster Azul: 122 puntos" con la finalidad de que se pueda identificar el número de puntos que hay en cada cluster con base en los colores.
- Se pregunta al usuario coordenadas de 'x' y 'y' para un nuevo punto y este se guarda en un array de numpy. Se predice el cluster en el que está y de igual manera se identifica con el color del cluster.
- Al final, se hace un scatter plot con los puntos del dataset, identificando los puntos con los colores diferentes en los clusters, con los centroides marcados con una X, y en una estrella se identifica el punto del usuario con el que se usó la predicción KNN.

Es importante destacar lo siguiente:

- En problemas supervisados, las métricas como precisión, recall, F1, y la matriz de confusión son fundamentales para evaluar la capacidad del modelo de clasificar correctamente las observaciones con base en etiquetas conocidas. Sin embargo, este es un problema no supervisado y no existen etiquetas para comparar, lo que hace inapropiado el uso de estas métricas. En cambio, la evaluación se enfoca más en cómo de coherentes y compactos son los clusters resultantes.
- En cuanto al KNN, al ser sólo un punto, no es necesario ninguna métrica de precisión ya que son solo ver la gráfica, o en dado caso los clusters de sus vecinos más cercanos, será suficiente para demostrar su efectividad.
- En problemas no supervisados, dividir el conjunto de datos en entrenamiento, validación, y prueba es menos relevante, ya que el objetivo no es aprender una función que generalice a partir de datos etiquetados, sino descubrir patrones en el conjunto de datos completo, recalcando que se debe de usar la totalidad de los datos para aprender de todos ellos.