Punto 1: Sistema de Recomendación y filtrado

Problema:

Justificar y graficar cual seria la mejor estructura jerárquica para un sistema de recomendación que usa árboles para filtrar y recomendar productos o servicios basados en preferencias y un conjunto de datos de al menos 200 elementos.

Solución:

Para un sistema de recomendación basado en árboles, una estructura jerárquica eficiente podría ser el uso de un KD-Tree (K-dimensional tree).

¿Por qué un KD-Tree?

Un KD-Tree es una estructura de datos que divide el espacio en regiones basadas en múltiples dimensiones. En el caso de una tienda de ropa, por ejemplo, estas dimensiones podrían ser características como el tipo de prenda, el precio, el material, el público para el cual fue confeccionada la prenda, entre otros. Los KD-Trees son especialmente útiles para sistemas de recomendación porque permiten realizar búsquedas eficientes en espacios multidimensionales, lo que es ideal para encontrar prendas que coincidan con las preferencias específicas de un cliente.

Si bien es cierto que hay otros tipos de estructuras de datos que podrían funcionar para este tipo de problemas incluyendo knn (búsqueda de vecinos cercanos), es cierto que dada la relativa simplicidad y su gran eficiencia, un KD-Tree es una excelente opción para este tipo de tareas siempre que se busque eficiencia, simplicidad y precisión para las recomendaciones. Además, tienen una gran solvencia manejando grandes conjuntos de datos (del orden de millones) en espacios de baja dimensionalidad (2-20 dimensiones), lo que los convierte en una opción más que viable en este tipo de situaciones.

Estructura Propuesta

1. Nodos: Cada nodo del árbol representa un producto, conteniendo sus características como dimensiones (tipo de prenda, precio, el material, el público para el cual fue confeccionada la prenda, etc).
2. Divisiones: El árbol se divide basándose en una dimensión en cada nivel, alternando entre las dimensiones disponibles.

Sara Lorena Duque Ramírez
Natalia Andrea García Ríos

3. Búsqueda: Para recomendar prendas de ropa, se busca en el árbol prendas de ropa similares a las que le gustaron al usuario, utilizando sus preferencias como punto de referencia.

Implementación en Python

Utilizando la biblioteca scikit-learn en Python, hemos desarrollado un sistema de recomendación para una tienda de ropa. Este sistema se basa en la implementación de un KD-Tree, una estructura de datos eficiente para búsqueda espacial, que nos permite determinar qué prendas de ropa son más recomendables a partir de un conjunto de datos previamente proporcionado.

En este proyecto, consideramos una serie de características clave, como el tipo de prenda, el precio, el material y el público al que está dirigida cada prenda. Para lograr esto, seguimos una serie de pasos específicos en el desarrollo del sistema:

1. Recolección de datos interactiva: Al inicio se buscó una forma interactiva de recopilar datos para el sistema de recomendación. Para ello, se utilizó la librería *IPyWidgets* en los Jupyter Notebooks. Se implementó un código que presentaba listas desplegables (dropdowns) interactivas, donde los usuarios podían seleccionar el tipo de prenda, el material, el público objetivo y si consideraban la prenda como recomendada o no. Además, se incluyó un control deslizante (float slider) que permitía definir el precio de la prenda. Finalmente, se agregó un botón para añadir la prenda al conjunto de datos (dataset) que se utilizará más adelante para entrenar el modelo de recomendación.

   Sin embargo, se identificó un problema durante la recolección de datos: la cantidad limitada de datos generaba resultados sesgados. Así que, para solucionar esto, se desarrolló un fragmento de código que permitía generar filas con datos aleatorios que cumplían con las características requeridas.

2. Preparación de datos utilizando One Hot-Encoding: Para adaptar las características de las prendas de ropa a un formato numérico adecuado para el KD-Tree, se aplicó el método de codificación One Hot Encoding. Este método convierte las características categóricas en vectores binarios, asignando un valor binario único a cada categoría presente en una característica. Por ejemplo, si había diferentes tipos de tela, cada tipo se convertiría en una columna binaria con valores de 0 o 1, indicando la presencia o ausencia de ese tipo de tela en una prenda en particular.

3. Implementación del KD-Tree con *scikit-learn*: Utilizando la biblioteca *scikit-learn*, se construyó un KD-Tree basado en las características transformadas de las prendas. La línea de código `knn = KNeighborsClassifier(n_neighbors=3)` inicializa el clasificador de vecinos más cercanos (KNeighborsClassifier) con un valor de 3 para el número de vecinos. Este clasificador se utiliza en la construcción del KD-Tree para identificar los vecinos más cercanos a un punto dado en el espacio de características.

4. Entrenamiento del modelo: La instrucción `knn.fit(X, y)` se empleó para entrenar el modelo KD-Tree. Aquí, `X` representa las características de las prendas transformadas mediante One Hot-Encoding, mientras que `y` corresponde a las etiquetas o la información de si una prenda es recomendada o no, según la interacción de los usuarios.

5. Prueba del modelo a través de una interfaz interactiva: Se creó una nueva interfaz para que los usuarios interactuaran con los widgets mencionados anteriormente. Sin embargo, esta vez, se eliminó el botón de agregado al dataset. Y, adicional, cada vez que se modificaba uno de los widgets, el sistema evaluaba las nuevas características y determinaba si el producto era recomendado o no, imprimiendo este resultado para la retroalimentación del usuario. Esta fase permitió verificar la efectividad del modelo en tiempo real y su capacidad para proporcionar recomendaciones basadas en las preferencias seleccionadas por el usuario.

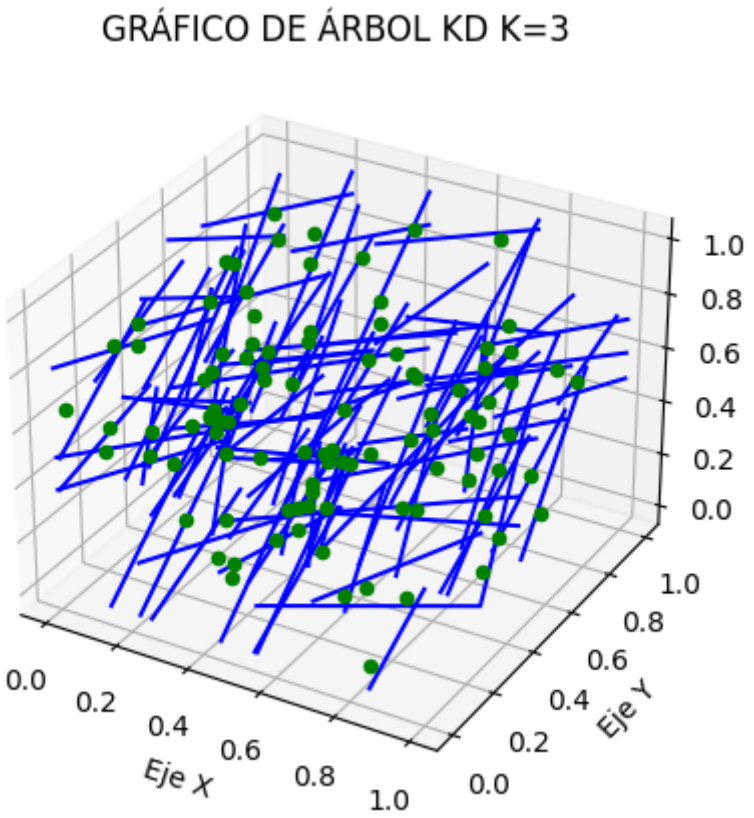
Pueden observar el notebook de Jupyter realizado, aquí.

Visualización

Para visualizar la estructura del KD-Tree, se puede utilizar una representación gráfica, pero dado que estamos trabajando con múltiples dimensiones, es complicado representar directamente el resultado. Una aproximación sería visualizar proyecciones en 2D o 3D de las divisiones del árbol.

La visualización difiere entre el ordenador y el usuario debido a que este tipo de máquinas no tienen la capacidad de visualizar internamente el KD-Tree como un espacio de múltiples dimensiones sino como una colección de puntos almacenados en posiciones de memoria (a menudo arreglos en memoria contigua para mayor eficiencia) conectados unos con otros. Para nosotros puede ser más fácil visualizarlos como una colección de puntos en el espacio, aunque es cierto que no es posible verlo de un modo sencillo para un espacio de más de tres dimensiones.

Sin embargo, graficando un KD-Tree de tres dimensiones con datos aleatorios podríamos obtener lo siguiente:



En donde:

- Los puntos verdes representan los puntos de datos en el espacio 3D.
- Las líneas azules representan las divisiones del árbol KD. Cada división representa un nodo en el árbol KD, y divide el espacio en dos según el valor del punto en el nodo a lo largo del eje correspondiente.
- Los ejes están etiquetados como “Eje x”, “Eje y” y “Eje z”, que representan las tres dimensiones del espacio.
- Este tipo de gráficos se utilizan a menudo para visualizar cómo un árbol KD organiza los puntos en el espacio, y pueden ser útiles para entender conceptos como la búsqueda de vecinos más cercanos y la eficiencia de las consultas de rango en un árbol KD.

Conclusión

Un KD-Tree es una excelente opción para un sistema de recomendación debido a su eficiencia en espacios multidimensionales y su capacidad para realizar búsquedas rápidas

basadas en ciertas características propias de los datos. En muchos casos es una gran opción debido a su relativa simplicidad de implementación y eficiencia en la búsqueda de vecinos cercanos comparado a, por ejemplo, los quadtrees en espacios de baja dimensionalidad.

Con todo esto en mente, ha sido una muy buena decisión la decisión de implementar dadas las ventajas ampliamente mencionadas a lo largo de este escrito y que el problema de maldición de dimensionalidad, su mayor inconveniente, no aplica para esta implementación específica.