Implementación del Modelo de Clasificación con KD-Tree

10 de septiembre de 2023

1. Introducción

El KD-Tree (K-Dimensional Tree) es un árbol binario de búsqueda utilizado para organizar puntos en un espacio multidimensional. Su principal ventaja radica en su capacidad para acelerar operaciones de búsqueda, especialmente la búsqueda de los k vecinos más cercanos a un punto dado. Este algoritmo encuentra aplicaciones en una variedad de áreas, incluida la clasificación de patrones y la búsqueda de similitud en grandes conjuntos de datos.

En este informe, se presentará una implementación de un modelo de clasificación basado en KD-Tree. Se explorará cómo funciona este algoritmo en el contexto de la clasificación. Además, se examinaá la estructura de datos KD-Tree en detalle y cómo se utiliza para acelerar la búsqueda de vecinos cercanos en espacios de alta dimensionalidad.

2. Algoritmo KD-Tree

Un KD-Tree, que significa "K-Dimensional Tree.º "Árbol K-Dimensional", es una estructura de datos utilizada principalmente en la búsqueda y organización de puntos en espacios K-dimensionales, donde K representa el número de dimensiones en el espacio. Los KD-Trees son particularmente útiles para acelerar operaciones de búsqueda en espacios multidimensionales, como la búsqueda de vecinos más cercanos en geometría computacional y problemas de procesamiento de imágenes.

el algoritmo que se utilizó tiene la siguiente estructura:

2.1. Clase KDTree

- **__init__ (self , points , dim, dist_sq_func =None):** El constructor de la clase KDTree. Recibe una lista de puntos, la dimensión de los puntos y una función opcional para calcular la distancia cuadrada entre dos puntos. Si no se proporciona, se utiliza una implementación por defecto.

- **make(points, i=0):** Una función interna para construir el KD-Tree recursivamente. Ordena los puntos en función de la dimensión actual, divide el conjunto en dos partes y construye subárboles para cada parte.
- **add_point(node, point, i=0):** Agrega un punto al árbol KD-Tree. Recibe un nodo y un punto, y decide en qué rama del árbol debe insertarse el punto.
- **get_knn(node, point, k, return_dist_sq, heap, i=0, tiebreaker=1):** Encuentra los k vecinos más cercanos al punto dado. Utiliza un enfoque eficiente para buscar los vecinos más cercanos.
- **walk(node):** Recorre el árbol KD-Tree en orden y devuelve todos los puntos contenidos en él.
- **__iter__ (self):** Permite iterar sobre todos los puntos del KD-Tree.
- **add_point(self, point):** Agrega un punto al KD-Tree.
- **get_knn(self, point, k, return_dist_sq=True):** Devuelve una lista de los k vecinos más cercanos al punto dado. Puede devolver las distancias cuadradas si return_dist_sq es True.
- **get_nearest(self, point, return_dist_sq=True):** Devuelve el punto más cercano al punto dado. Puede devolver la distancia cuadrada si return_dist_sq es True.

2.2. Clase KDTreeClassifier

- **__init__ (self , points , dim, classes , dist_sq_func =None):** Constructor de la clase KDTreeClassifier. Recibe una lista de puntos, la dimensión de los puntos, una lista de clases correspondientes a los puntos y una función opcional para calcular la distancia cuadrada entre dos puntos.
- **build_kd_tree(self, dist_sq_func=None):** Construye el KD-Tree utilizando los puntos proporcionados y la función de distancia cuadrada opcional.
- **classify(self, point, k):** Clasifica un punto dado encontrando sus k vecinos más cercanos y votando por la clase más común entre esos vecinos.
- **point_label_dict:** Un diccionario que asigna cada punto a su etiqueta de clase correspondiente. Esto se utiliza en la función classify para contar las clases de los vecinos más cercanos.

3. Implementación

En esta implementación, se desarrolló un modelo de clasificación utilizando un KD-Tree (Árbol K-dimensional) para realizar predicciones en un conjunto de datos relacionado con la diabetes. El objetivo principal es clasificar a las personas como diabéticas o no diabéticas en función de varias características médicas y personales.

3.1. Carga y Preprocesamiento de Datos

El proceso comienza importando las bibliotecas esenciales, como Pandas, NumPy y Scikit-Learn, para manipular y evaluar los datos. Además, se crearon dos diccionarios (codigo_mapa y codigo_mapa_2) para codificar las variables categóricas presentes en el conjunto de datos original. Luego, se cargó el conjunto de datos desde un archivo CSV llamado 'diabetes_prediction_dataset.csv' utilizando la biblioteca Pandas.

A continuación, se realizaron las siguientes acciones de preprocesamiento de datos:

- Se crearon dos nuevas columnas en el DataFrame data llamadas smoking_history_num y gender_num, que representan las codificaciones numéricas de las columnas originales 'smoking_history' y 'gender'.
- Las columnas originales 'smoking_history' y 'gender' se eliminaron del DataFrame para mejorar la eficiencia del modelo.
- Los datos se dividieron en conjuntos de entrenamiento (train_data) y prueba (test_data) utilizando la función train_test_split de Scikit-Learn.

3.2. Construcción del Modelo KD-Tree

Una vez preparados los datos, se procedió a construir el modelo de KD-Tree para realizar clasificaciones. Se creó una instancia de la clase KDTreeClassifier, denominada arbolknn, que se inicializó con las características de entrenamiento (X), la dimensión del espacio de características y las etiquetas de entrenamiento (C).

3.3. Clasificación y Predicciones

Para realizar predicciones en el conjunto de prueba, se estableció el valor de k en 5, que representa el número de vecinos más cercanos considerados para la clasificación KNN. Luego, se iteró a través de las características de prueba (Y) y se utilizó el método classify de arbolknn para predecir la etiqueta de cada punto. Estas predicciones se almacenaron en una lista llamada predictions.

3.4. Evaluación de Métricas de Clasificación

Para evaluar el rendimiento del modelo, se calcularon varias métricas de clasificación, que incluyen True Negatives (TN), False Positives (FP), False Negatives (FN), True Positives (TP), Recall, Precision, Specificity, Accuracy y F1-Score. Estas métricas proporcionan información sobre la capacidad del modelo para predecir de manera precisa las etiquetas de diabetes.

Finalmente, se imprimieron estas métricas para evaluar el rendimiento global del modelo KD-Tree en la clasificación de personas como diabéticas o no diabéticas.

4. Resultados

Los resultados obtenidos a partir de la implementación del modelo KD-Tree para la clasificación de personas con diabetes son altamente prometedores y sugieren que el modelo tiene un buen rendimiento en la tarea de predicción. A continuación, se desglosan las métricas de evaluación clave:

Cuadro 1: Métricas de evaluación

Recall	**Precision**	**Specificity**	**Accuracy**	**F1-Score**
0.5211	0.8599	0.9921	0.9519	0.6489

Recall (Recuperación): El valor de Recall es de aproximadamente 0.5211. Esto significa que el modelo es capaz de identificar correctamente alrededor del 52.11 % de todas las personas que realmente tienen diabetes. En otras palabras, el modelo tiene una tasa razonable de detección de casos positivos reales.

Precision (Precisión): La precisión alcanza un valor de 0.8599. Esto indica que el 85.99 % de las predicciones positivas realizadas por el modelo son correctas en relación con el total de predicciones positivas. En otras palabras, el modelo tiende a realizar predicciones positivas con una alta precisión.

Specificity (Especificidad): La especificidad es de aproximadamente 0.9921. Esto significa que el modelo es muy bueno para identificar a las personas que no tienen diabetes (casos negativos reales). La alta especificidad indica que el modelo tiene una baja tasa de falsos positivos.

Accuracy (Exactitud): La precisión global del modelo, representada por la Accuracy, alcanza un valor de 0.9519. Esto indica que el 95.19 % de las predicciones totales realizadas por el modelo son correctas en relación con el conjunto de datos de prueba. El modelo tiene un alto nivel de exactitud general.

F1-Score: El valor de F1-Score es de aproximadamente 0.6489. El F1-Score es una medida que combina tanto la precisión como el recall, lo que significa que tiene en cuenta tanto los falsos positivos como los falsos negativos. Un F1-Score de 0.6489 sugiere un equilibrio entre la precisión y el recall.

En resumen, el modelo KD-Tree ha demostrado ser eficaz en la clasificación de personas con diabetes. Si bien presenta un recall moderado, lo que indica que podría mejorarse en la detección de todos los casos positivos, la alta precisión

y especificidad junto con una excelente exactitud global hacen que este modelo sea prometedor para aplicaciones médicas de diagnóstico. Además, el F1-Score equilibrado subraya la capacidad general del modelo para realizar predicciones precisas y confiables.

5. Conclusiones

Eficiencia en la Clasificación: El modelo KD-Tree se ha revelado como una herramienta eficiente para la clasificación de datos . A pesar de su simplicidad, ha logrado un alto nivel de precisión, especificidad y exactitud general en la tarea de clasificar.

Balance entre Precisión y Recall: El modelo logra un equilibrio adecuado entre la precisión y el recall, como se refleja en el valor del F1-Score. Esta característica es fundamental en aplicaciones médicas, donde tanto los falsos positivos como los falsos negativos pueden tener consecuencias significativas.

Optimización Futura: A pesar de los resultados prometedores, existen oportunidades para mejorar el modelo, especialmente en el aumento del recall para detectar más casos positivos reales. Esto podría lograrse a través de la exploración de diferentes algoritmos de clasificación o la optimización de los parámetros del modelo KD-Tree.

Continuación de la Investigación: Los resultados obtenidos en esta implementación respaldan la continuación de la investigación en el uso de estructuras de datos espaciales como KD-Tree en aplicaciones médicas y de diagnóstico.