Memoria Práctica 2. ID3

Integrantes:

-Francisco José Fernández Ferreiro

-Alejandro Moreno Murillo

-Sergio Sánchez Chamizo

Desarrollo de la práctica:

El código está implementado en Python, utilizando algunas librerías como pandas, numpy y igraph.

Durante el desarrollo de la práctica, al ser un algoritmo que es utilizado de manera común, pudimos buscar diferentes ejemplos sobre este algoritmo, que es usado en otro tipo de proyectos y aplicándolo a nuestra práctica pudimos conseguir las funciones que explicaremos más adelante.

Lo primero que hicimos como ampliación fue conseguir todos los niveles de recursividad del árbol que resultó sencillo ya que encontramos ejemplos de gran utilidad y los pusimos en práctica. Como segunda mejora nos propusimos mostrar visualmente el árbol, y debido a que no encontrábamos pistas sobre ello, nos llevó bastante tiempo, pero finalmente lo conseguimos buscando más a fondo y probando todo tipo de cosas. Cuando creamos el árbol con el algoritmo ID3 éste se nos crea como un diccionario de python, el cual manualmente hemos pintado con la librería “networkx”.

Procedimiento seguido para su implementación:

El código está dividido en varias funciones que realizan tareas específicas en el proceso de construcción del árbol de decisión. La primera función, "calc_total_entropy", calcula la entropía total del conjunto de datos a partir de la columna que contiene el atributo objetivo. La función toma como entrada los datos, el nombre de la columna de el atributo objetivo y la lista de clases de el atributo objetivo. La función itera sobre cada clase de el atributo objetivo y calcula la entropía para cada una de ellas, sumándose para obtener la entropía total.

La siguiente función, "calc_entropy", calcula la entropía de los datos que corresponden a un valor específico de una variable. La función toma como entrada los datos correspondientes a ese valor, el nombre de la columna de el atributo objetivo y la lista de clases de el atributo objetivo. La función

itera sobre cada clase de el atributo objetivo y calcula la entropía para cada una de ellas, sumándose para obtener la entropía total.

La tercera función, "calc_info_gain", calcula la ganancia de información de una variable. La función toma como entrada el nombre del atributo, los datos, el nombre de la columna de el atributo objetivo y la lista de clases de el atributo objetivo. La función calcula la entropía total del conjunto de datos y luego calcula la entropía para cada valor único del atributo, ponderándola por la probabilidad de que un ejemplo tenga ese valor. La ganancia de información se calcula restando la entropía total de la suma ponderada de las entropías para cada valor único.

La cuarta función, "find_most_informative_feature", encuentra el atributo con la mayor ganancia de información. La función toma como entrada los datos, el nombre de la columna de el atributo objetivo y la lista de clases de el atributo objetivo. La función itera sobre cada variable y llama a la función "calc_info_gain" para cada una de ellas. Devuelve el nombre del atributo con la mayor ganancia de información. Este atributo será la raiz de nuestro árbol de decisión

La quinta función, "generate_sub_tree", genera un subárbol para un valor específico de una variable. La función toma como entrada el nombre del atributo, los datos, el nombre de la columna de el atributo objetivo y la lista de clases de el atributo objetivo. La función cuenta cuántas veces aparece cada valor único del atributo y crea un diccionario para almacenar el subárbol. Para cada valor único de el atributo, la función filtra los datos para incluir solo aquellos que tienen ese valor, y luego verifica si es una clase pura o no. Si es una clase pura, agrega un nodo al diccionario con el valor de la clase. Si no es una clase pura, agrega un nodo al diccionario con un signo de interrogación y llama a la función "make_tree" para construir un subárbol más profundo.

La función "make_tree" construye el árbol de decisión.

La función “predict” toma como entrada un modelo de árbol de decisión generado por la función “make_tree”, así como una fila de datos. La función utiliza el modelo para predecir la clase de la fila de datos proporcionada. La función comienza en la raíz del árbol de decisión y avanza hacia abajo por el árbol de decisión hasta llegar a una hoja. Cada nodo del árbol de decisión tiene una etiqueta que corresponde a una columna del conjunto de datos, y cada rama tiene una etiqueta que corresponde a un valor posible de esa columna. La función compara el valor de la columna en la fila de datos proporcionada con el valor de la etiqueta de la rama actual. Si los valores coinciden, la función sigue por esa rama del árbol de decisión. Si los valores no coinciden, la función sigue por otra rama del árbol de decisión. La función continúa este proceso hasta llegar a una hoja del árbol de decisión, que contiene la predicción de la clase.

La función “evaluate” toma como entrada un modelo de árbol de decisión generado por la función “make_tree” y un conjunto de datos de prueba. La función utiliza el modelo para hacer predicciones para cada fila en el conjunto de datos de prueba y luego calcula la precisión de las predicciones en comparación con las verdaderas clases de la columna de decisión del conjunto de datos de prueba.

La precisión se define como el número de predicciones correctas dividido por el número total de predicciones. La función devuelve la precisión de las predicciones como un número entre 0 y 1.

Manual de Usuario:

Para comprobar la ejecución de esta práctica necesitaremos tener a mano un navegador y los ficheros de datos adjuntados en el entregable de la práctica.

Para empezar, abriremos la carpeta de drive y el video para repetir los pasos.

Video manual de usuario: Tutorial

Enlace al drive con el cuaderno de Python usando Google Colab de la práctica: Drive