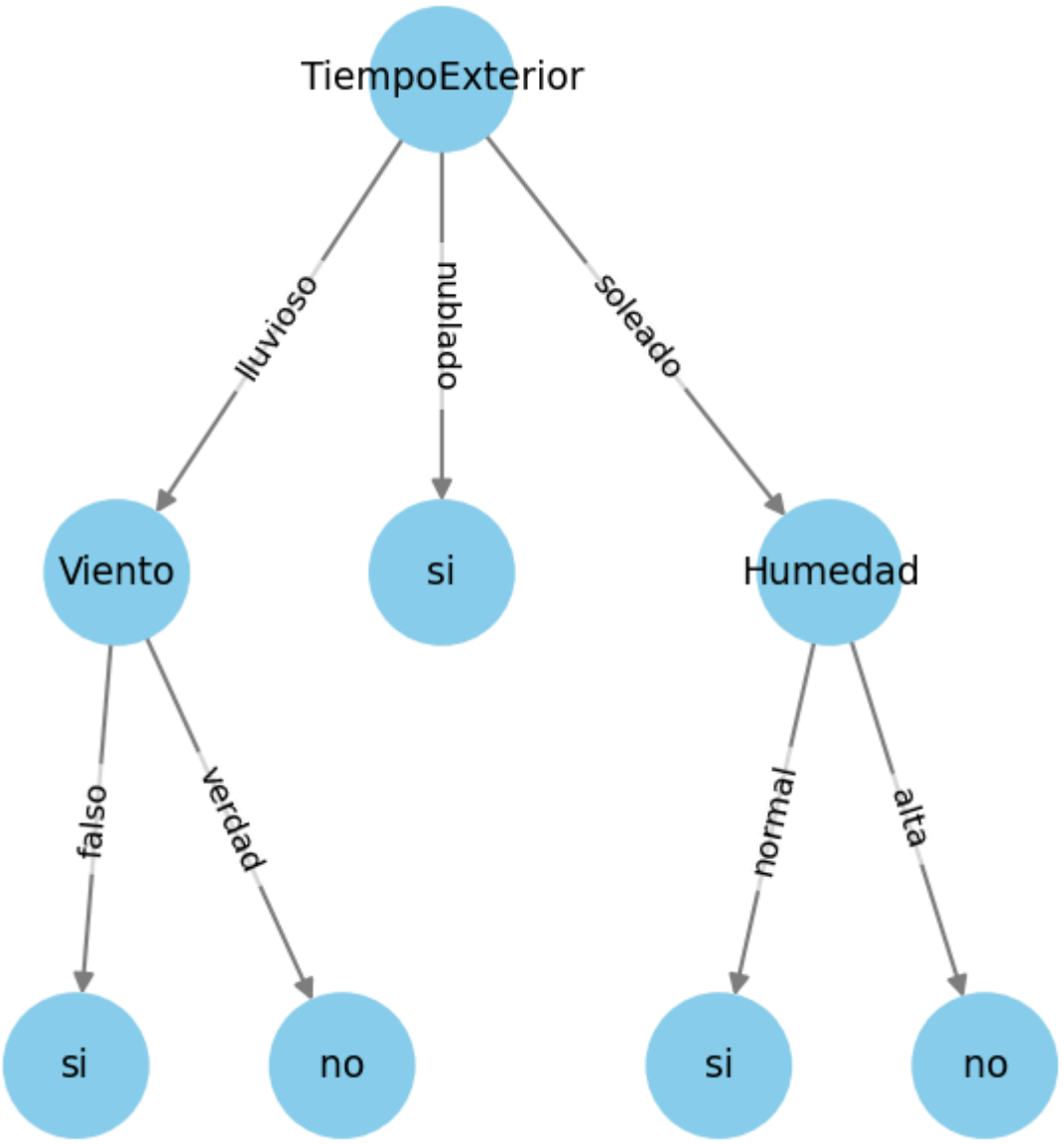
Algoritmo ID3 en Python



Jaime Alonso Fernández 2024/2025

Ingeniería del Conocimiento:Grado en Ingeniería Informática de la Universidad Complutense de Madrid. Plan académico 2019

1. Descripción del algoritmo base

La implementación del algoritmo base, se limita al mismo algoritmo que realizamos en papel en clase. El objetivo será construir un árbol de decisión que clasifique un conjunto de ejemplos según sus atributos.

La estructura del árbol se decide en función de una heurística (por ejemplo, entropía o ganancia de información), escogiendo en cada paso el mejor atributo para dividir los datos.

Durante la ejecución, el árbol se construye recursivamente y se utilizan nodos con información sobre los atributos y los valores de estos. Las decisiones tomadas en cada nodo se representan mediante aristas etiquetadas con el valor del atributo correspondiente.

1.1. Implementación del algoritmo base

Para entender el funcionamiento de la implementación, primero debemos entender el funcionamiento de cada uno de los métodos auxiliares usados. Entendida esta parte fundamental, podemos pasar a ver la implementación final.

Métodos auxiliares:

Métodos de Lectura:

Aquí están todos los métodos que tienen como funcionalidad el leer los archivos donde se recoge la información que permitirá construir el árbol. Contamos con el siguiente:

- *read_file*

Métodos de ayuda:

Estos son los métodos auxiliares que abstraen los cálculos de manera que la lectura de la implementación final sea más accesible y fácil de leer. Contamos con los siguientes:

- *get_majority_label*
- *get_all_same_sign*
- *get_attributes_probabilities*

También incluyo aquí el método de predicción ya que este se basa en la estructura del árbol para funcionar. Este método se encuentra en el código bajo el seudónimo de ***predict***.

Métodos de interacción:

Finalmente aqui recogemos los métodos que para nuestra implementación inicial sirven de comunicación con el usuario, bien sea para hacer la predicción o para ver que está pasando. Tenemos los siguientes:

- *ask_if_prediction_and_predict*

- *get_possible_answers*
- *show_tree*

Método *__main__*:

Este es el método principal y solo se ejecutará en caso de que ejecutemos el archivo y no si otro file accede a él. Aquí recogemos la lógica para nuestra interacción desde la terminal. En caso de querer probarse con más archivos hay que cambiar el valor *'hard-coded'* dentro de este método.

Cálculo de heurística (mérito):

Este cálculo se da dentro del método *get_attributes_probabilities*. Este itera por todas las opciones que existen dentro del dataset y para cada una saca los valores de **n**, **p**, **r** y con ello su mérito. Este mérito será finalmente guardado en un diccionario (que después devuelve) de manera que exista fácil acceso a toda la información en el futuro. Véase el código en la imagen 1.

```
result = {}
# Por cada uno de los atributos de la lista creamos una entrada en el diccionario
for i, attribute in enumerate(attributes):
    result[attribute] = {'options': list(set(row[i] for row in examples))}
    merit = 0
    # Por cada entrada del atributo seleccionado
    for j, option in enumerate(result[attribute]['options']):
        # Esto sería 'n'
        total_element_count = sum(1 for row in examples if row[i] == option)
        # Contamos los valores positivos
        p_count = sum(1 for row in examples if row[i] == option and row[key_position] == key_positive)
        # Sacamos la fracción (p valor)
        p_value = p_count / total_element_count if total_element_count else 0
        # La inversa será entonces el n valor
        n_value = (total_element_count - p_count) / total_element_count if total_element_count else 0
        # Sacamos r
        r_value = total_element_count / len(examples)
        # Hacemos el cálculo del mérito
        merit += r_value * (-p_value * math.log2(p_value) if p_value else 0 - n_value * math.log2(n_value) if n_value else 0)
    # Guardamos el mérito en el diccionario
    result[attribute]['merit'] = merit
# Devolvemos el diccionario
return result
```

Imagen 1

Desarrollo del algoritmo:

Con toda la información mencionada, podemos pasar a implementar el algoritmo ID3 en nuestro código. Para ello primero comenzamos verificando que la información exista y sea correcta y no haya trivialidades. Esto se ve entre las líneas 55 y 78 de la implementación, donde detenemos la ejecución si:

a) No existen ejemplos.
b) Los ejemplos sí existen pero sólo contiene el atributo a predecir.

En estos dos casos se considera que estamos en un estado incorrecto y se frena la ejecución en ese punto. Por otra parte en ese lapso de código también verificamos que la columna del

atributo a predecir no contenga siempre el mismo valor, ya que en caso de tener una unidad de respuestas, se tomará ese valor como nodo final de esa rama, evitando cálculos innecesarios.

En cualquier otro caso de los no mencionados, se procederá a sacar los méritos de los atributos existentes. De entre esos atributos buscaremos aquel con un mérito menor y le asignaremos un nombre de manera que no haya repeticiones. Esto se hace así por como funciona la librería de construcción del árbol (networkX) ya que esta procesa los nodos por nombre y si tuviéramos dos nodos con el mismo nombre el programa lo entendería como el mismo.

Ahora, añadimos los nodos al árbol y le asignamos nombre a las aristas (edges), para finalmente pasar a crear un dataset nuevo el cual contenga los atributos y la información de todas las columnas, menos la seleccionada (la de menor mérito) y en cuanto a la información tomaremos sólo aquella que tenga la opción acorde a esa rama. (En caso de que el atributo con menor mérito sea tiempo, haremos 3 ramas, una para soleado, otra para lluvioso y otra para nublado. Para la rama de soleado tomaremos el mismo dataset pero solo donde las filas de Tiempo sea ‘soleado’ y luego eliminamos la columna ‘Tiempo’)

Finalmente haremos una llamada recursiva por cada rama nueva con la nueva información de manera que se calcule todo el algoritmo.

1.2. Expansión del algoritmo: Visualización del árbol

Acabada ya la base de la práctica, me propuse implementar todas las “expansiones” que pudiera. La primera de todas sería mostrar gráficamente el árbol generado.

Para ello, haremos uso del paquete **matplotlib** ya que la había utilizado anteriormente para mostrar gŕaficos y sabía manejarme con él. Comenzamos creando un “figure” el cual es el recuadro donde mostraremos la gráfica, luego calcularemos la posición de los nodos usando un método de **networkx** el cual te devuelve la posición para el formato que yo quería (uno arbóreo), realizamos el dibujo de los mismos, repetimos para las aristas (edges) y finalmente mostramos la figura. (Imagen 2).

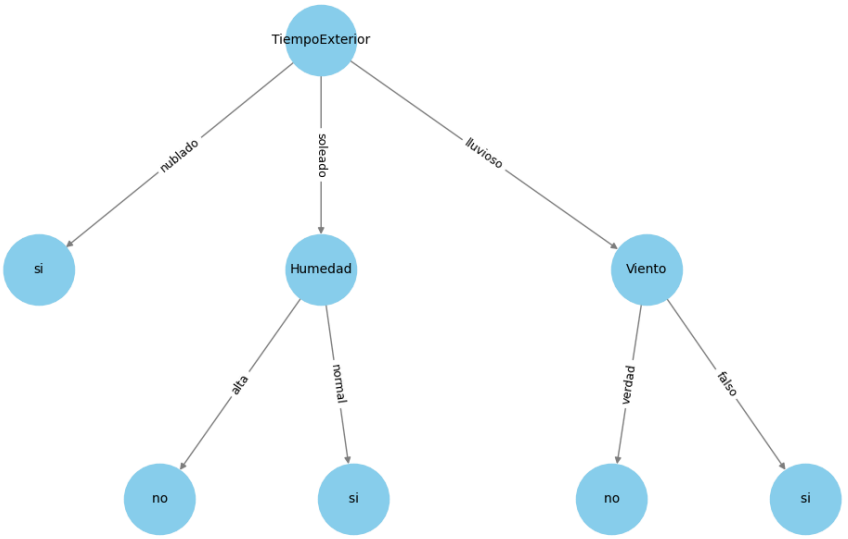


Imagen 2

1.3. Expansión del algoritmo: Input para la predicción

Está es la última expansión que se ha llevado a cabo de cara a la lógica.

De cara a lo que sería la lógica de la misma, no tiene una complicación demasiado excesiva: con el árbol ya construido, la librería **networkx** nos permite recorrer sus ramas fácilmente y es esto justo lo que necesitamos para poder sacar la predicción. Vamos iterando por los atributos a predecir y vemos si los nombres de las aristas (edge) son iguales al nombre del atributo y de ser así, tomamos ese recorrido hasta llegar al final.

Para esta expansión para mí lo más importante era que los usuarios no sólo vieran los campos a rellenar si no que siempre tuvieran presente las opciones para cada uno de los inputs. Por ello he creado un método externo el cual recoge todas las repuestas únicas en un set y se las muestra al usuario. Es precisamente este “abuso” del set (ya que lo utilizo porque no puede tener claves repetidas), el que hace que los elementos no aparezcan siempre en el mismo orden.

Finalmente el usuario es informado en ambas versiones (No GUI y GUI) de la predicción del modelo.

1.4. Expansión del proyecto: GUI

A lo largo de toda esta práctica, me he centrado en desarrollar las entrañas del proyecto. En esta práctica he tratado de usar python para, por una parte, seguir aprendiendo a usarlo correctamente, y por otra por la facilidad que ofrece mediante el uso de librerías.

Gracias a esto he sido capaz de acabar la lógica con bastante tiempo y por ello me puse con TKinter a preparar una GUI la cual facilite en cierta medida el uso de la aplicación. Para ello he creado una ventana dinámica que se va rellenando según los datos introducidos. Por una parte tenemos la pantalla en estado inicial la cual pide el path relativo de los archivos a leer (Imagen 3), una vez introducidos y pulsado el botón correspondiente, se generará y mostrará el árbol de decisión para el dataset (Imagen 4). Es en este punto también que aparecen los campos a la derecha donde podremos realizar una predicción de los valores, seleccionandolos mediante una combo box y tras pulsar el botón de predecir, se mostrará en un mensaje el resultado.

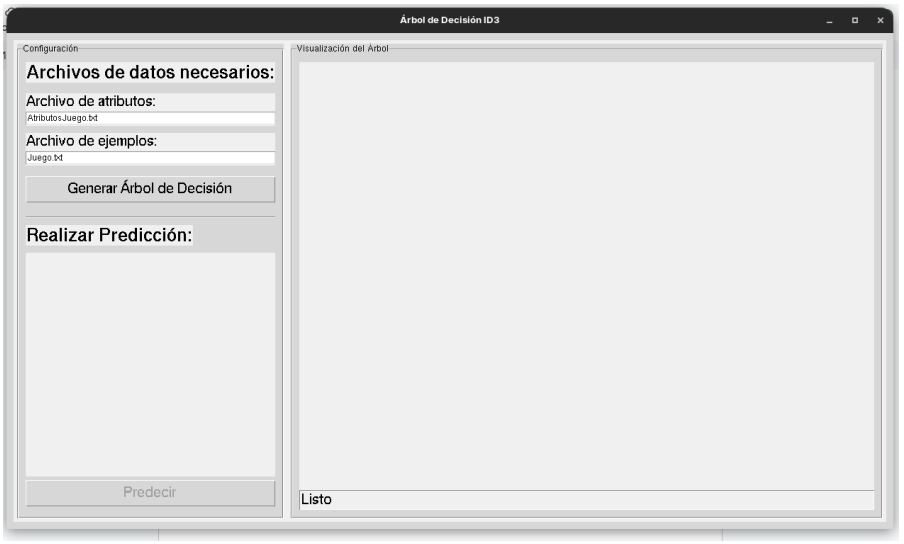


Imagen 3

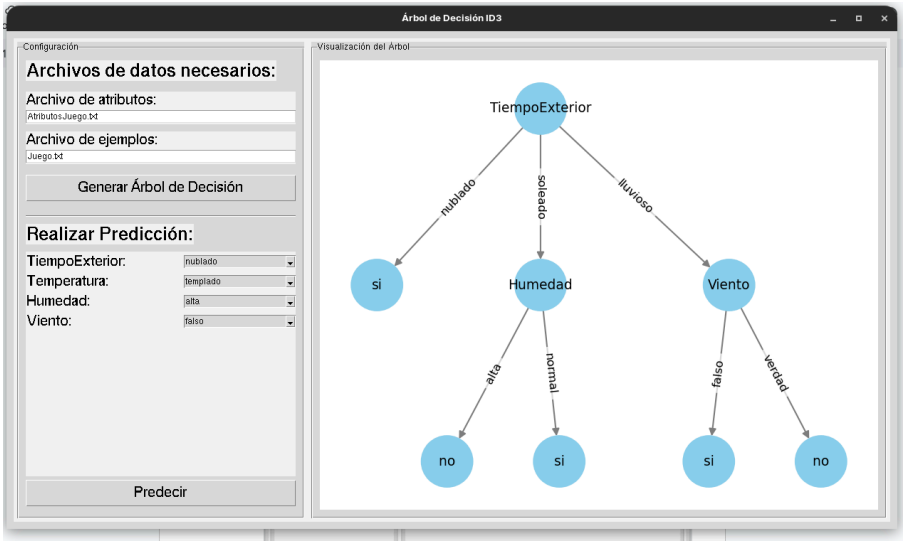


Imagen 4

2. Manual de uso:

Requisitos:

- Esta entrega no cuenta con un archivo ejecutable por lo que es imprescindible python 3.12 en adelante instalado a través de una fuente oficial (junto a pip) o desde conda.
- Las librerías externas empleadas son las siguientes:

Nombre Librería	Instalación mediante pip	Instalación mediante conda
matplotlib	pip install matplotlib	conda install matplotlib
networkx	pip install networkx	conda install networkx
pygraphviz	pip install pygraphviz	conda install pygraphviz

- Respecto al uso de pygraphviz: Es necesario junto con el (preferiblemente antes de instalarlo) instalar en tu dispositivo Graphviz. Cualquier problema en su instalación o ejecución podrá deberse al no contar con este paquete en su instalación.

Ejecución:

La ejecución del proyecto puede llevarse a cabo por diferentes medios, desde IDE 's como Pycharm o Visual Studio Code, o a través de una terminal. En ambos casos podremos usar tanto la versión de terminal como la versión con GUI.

Para ejecutar la versión de terminal tendremos que ejecutar el archivo **ID3DecisionTree.py**, ya que este no solo implementa la lógica si no que cuenta con un apartado que solo se ejecuta en situación de llamada específica y muestra una versión del proyecto (salvo para el árbol que lo muestra en un figure).

Para ejecutar la versión con GUI (recomendado) tendremos que ejecutar el archivo **GUI_Main.py**. Este contiene una clase interfaz haciendo uso de TKinter y referencia al archivo **ID3DecisionTree.py** del cual recoge toda la lógica, por lo que el comportamiento en ambos casos es el mismo.

La ejecución sin IDE se realizará abriendo una terminal en la carpeta del proyecto y ejecutando las siguientes líneas de comando:

Versión	Comando para python normal	Comando para python3
No GUI	python ID3DecisionTree.py	python3 ID3DecisionTree.py
GUI	python GUI_Main.py	python3 GUI_Main.py

3. Ejemplos de uso

Durante este apartado vamos a mostrar los 2 mismos casos de uso en las versiones GUI y NoGUI. Para ello usaremos el dataset proporcionado por los profesores (Juego.txt y AtributosJuego.txt) y otros que muestran una reducción a 3 atributos (Juego3.txt, AtributosJuego3.txt).

Ejemplo con el dataset original (NoGUI):

Primero hard-codeamos la información en el código de main (Imagen 5)

```
# Leemos atributos y los ejemplos proporcionados (material de entreno)
attributes_names = read_file( filepath: "AtributosJuego.txt",  separator: ",")
example_names = read_file( filepath: "Juego.txt",  separator: ",")
```

Imagen 5

Y procedemos a ejecutar el archivo **ID3DecisionTree.py**.

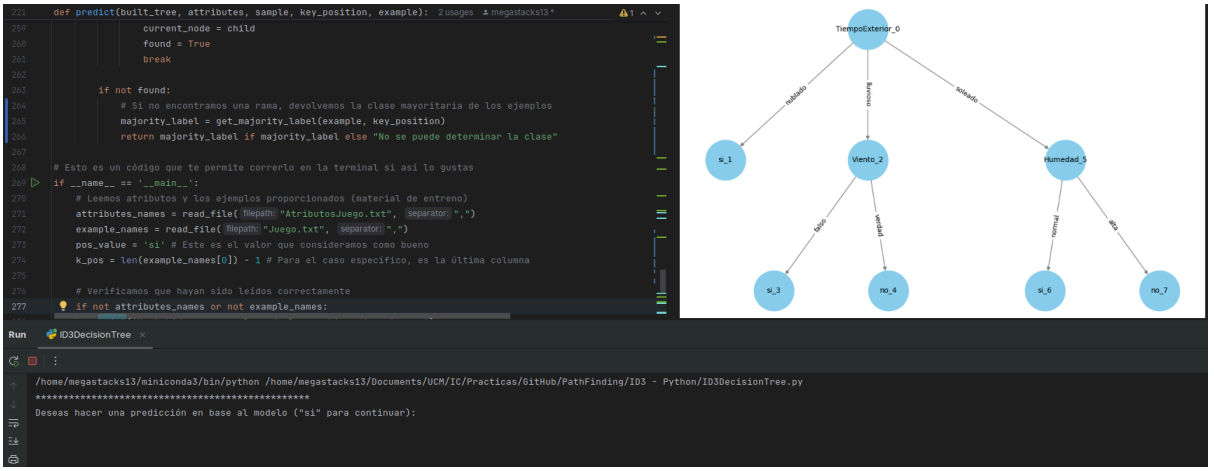


Imagen 6

Como vemos en la imágen 6 se nos ha construido el árbol a la derecha, donde podemos ver toda la información. Por otra parte en la terminal se nos pide escribir ‘si’ para realizar una predicción. Vamos a hacer una:

```
**********************************************
Deseas hacer una predicción en base al modelo ("si" para continuar): si
Comienzo de proceso de añadido manual de datos a la muestra:
    Valor para la variable TiempoExterior (['nublado', 'lluvioso', 'soleado']): lluvioso
    Valor para la variable Temperatura (['frio', 'templado', 'caluroso']): caluroso
    Valor para la variable Humedad (['normal', 'alta']): normal
    Valor para la variable Viento (['falso', 'verdad']): verdad

El modelo ha predicho: no
```

Imagen 7

Si ahora comparamos el resultado de la imagen 7 con el árbol, vemos que el resultado es correcto.

Ejemplo con el dataset original (GUI):

Esta vez comenzaremos ejecutando el archivo **GUI_Main.py**. Una vez ejecutado se nos abrirá una ventana como la vista en la imagen 8.

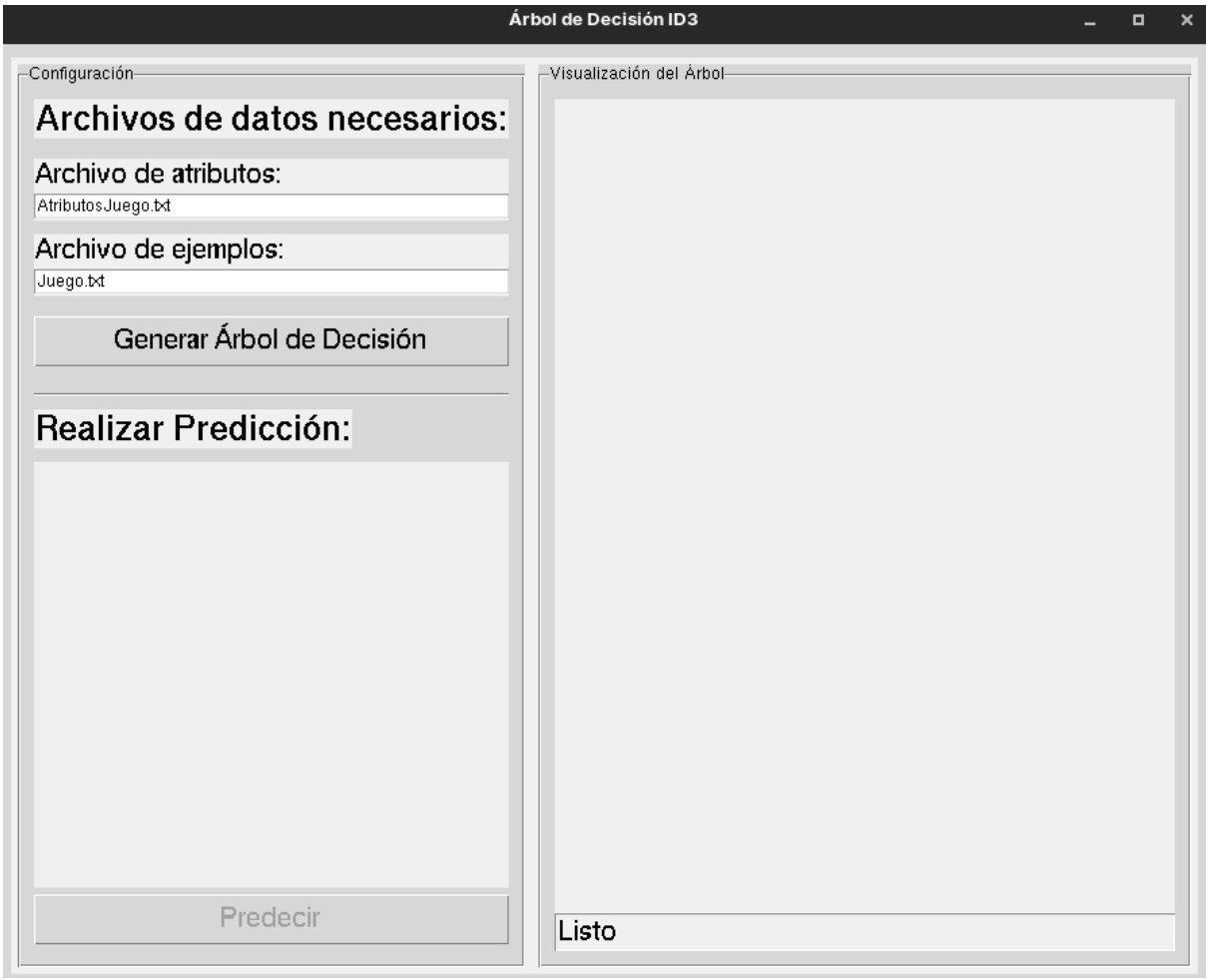


Imagen 8

En este caso dejaremos los **path** por defecto y le daremos a “Generar Árbol de Decisión”. Con esto se nos construirá un árbol en la ventana de la derecha y se abrirá la posibilidad de realizar predicción (Imagen 9).

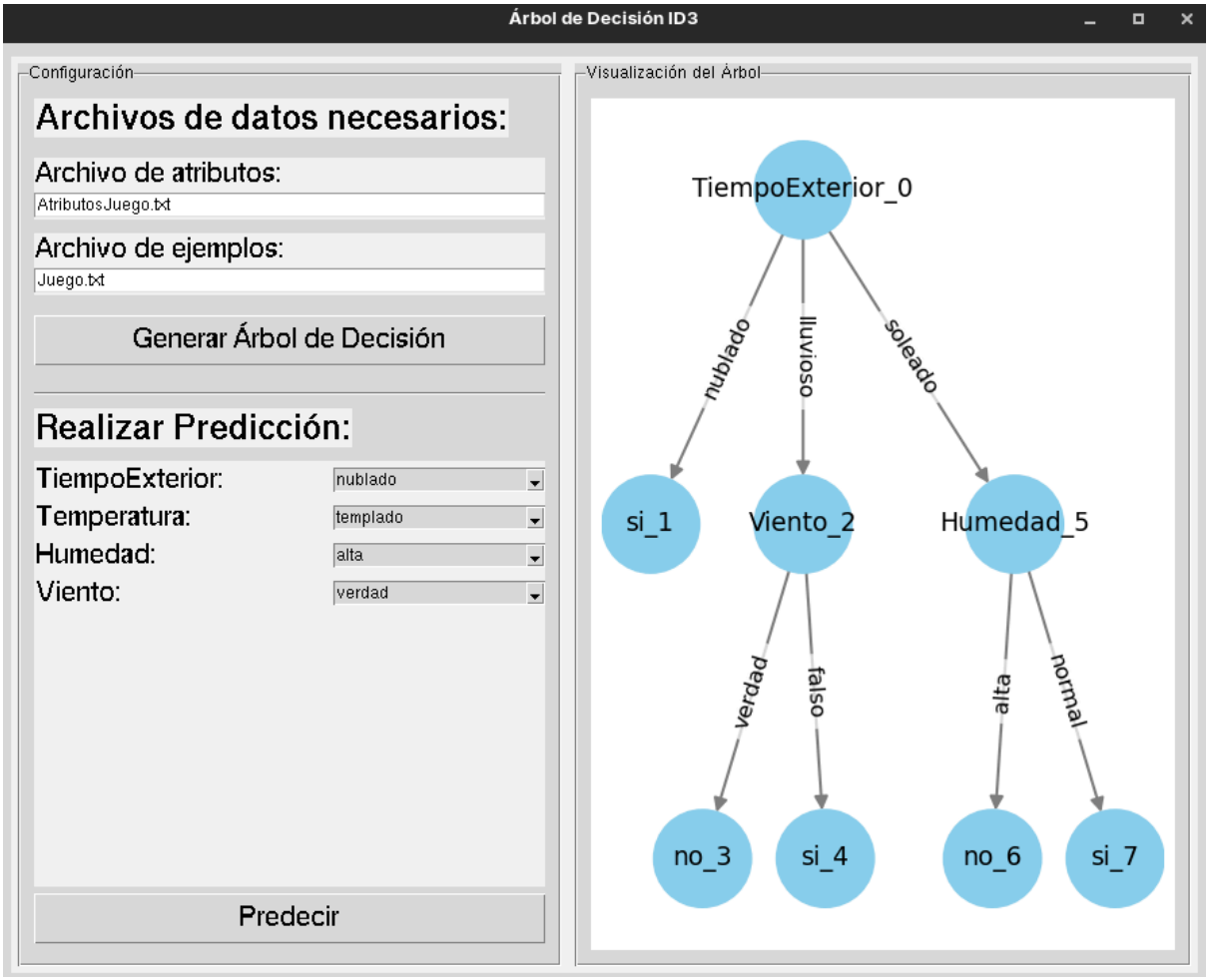


Imagen 9

Seleccionamos los mismos atributos que para la predicción anterior y ejecutamos pulsando el botón “Predecir” (Imagen 10).

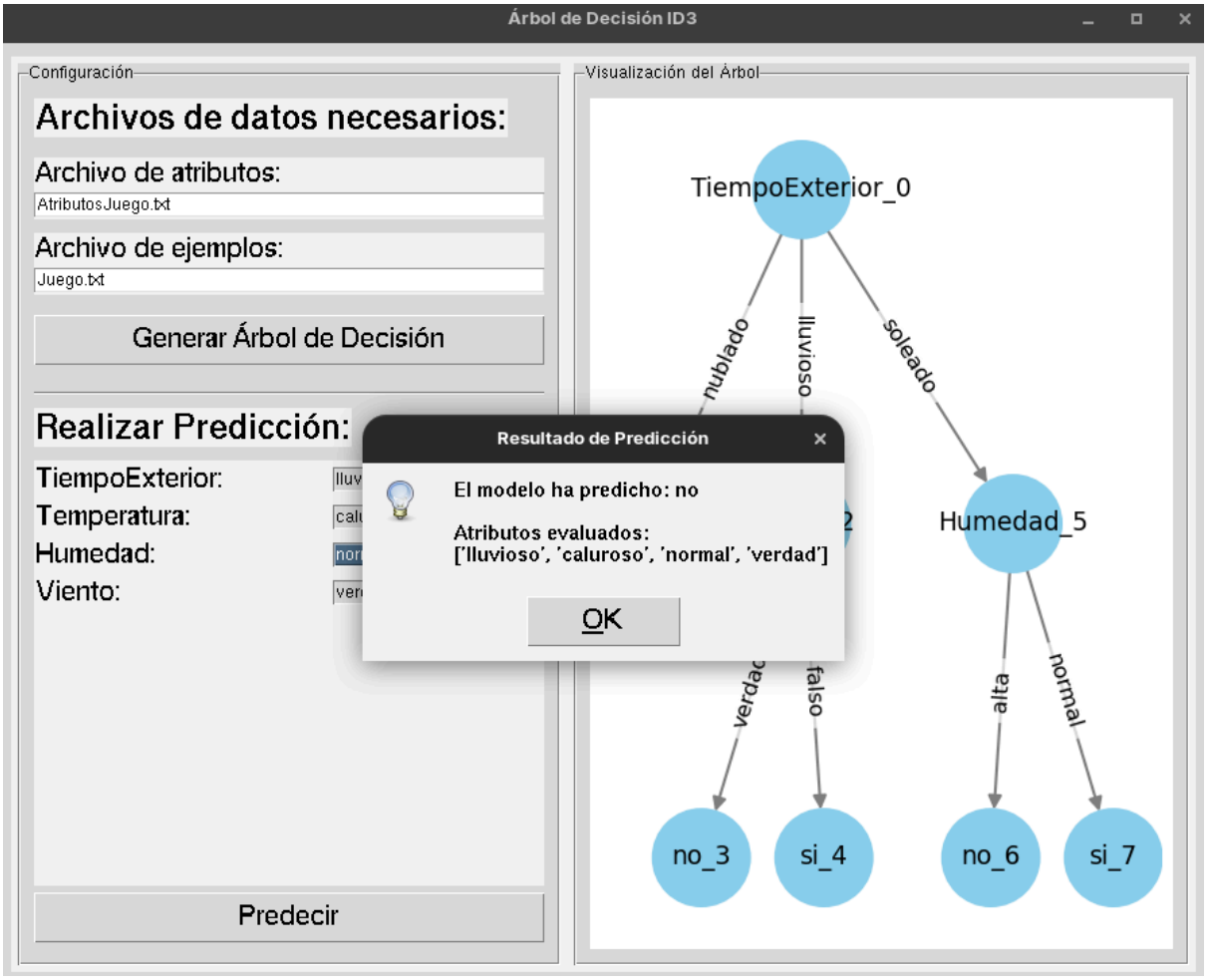


Imagen 10

Y como vemos en la predicción de la Imagen 10, esta concuerda con el árbol y con la predicción NoGUI.

Ejemplo con el dataset recortado (NoGUI):

Primero hard-codeamos la información adecuada (Imagen 11) en el código de main.

```
# Leemos atributos y los ejemplos proporcionados (material de entreno)
attributes_names = read_file( filepath: "AtributosJuego3.txt",  separator: ",")
example_names = read_file( filepath: "Juego3.txt",  separator: ",")
```

Imagen 11

Ahora procedemos a ejecutar el archivo **ID3DecisionTree.py**.

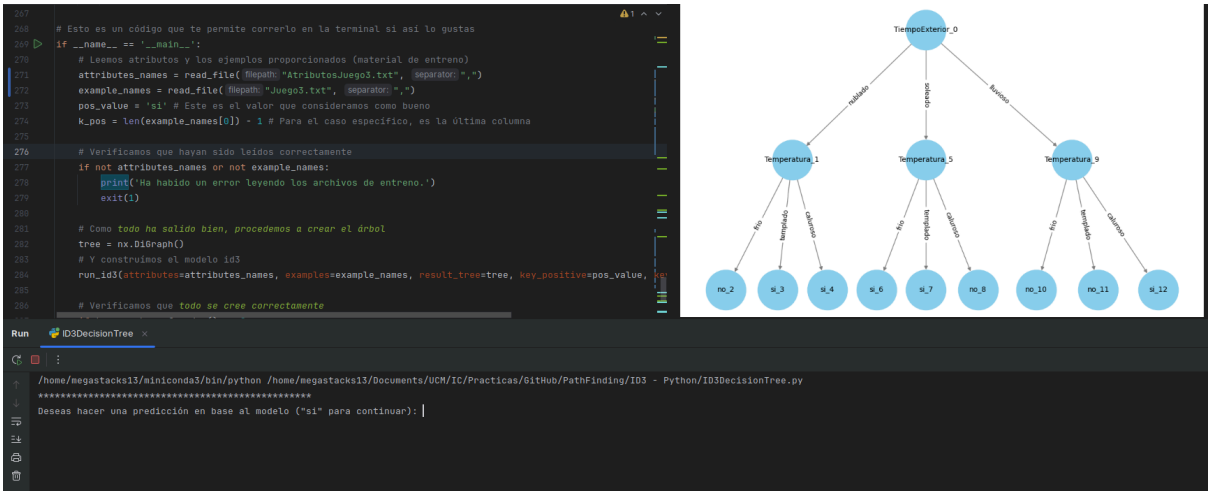


Imagen 12

Como vemos en la imagen 12, se nos ha construido el árbol a la derecha, donde podemos ver toda la información. Por otra parte en la terminal se nos pide escribir 'si' para realizar una predicción. Un ejemplo de predicción se puede ver en la imagen 13, donde si nos fijamos en el árbol y lo predicho, vemos que el resultado es acorde.

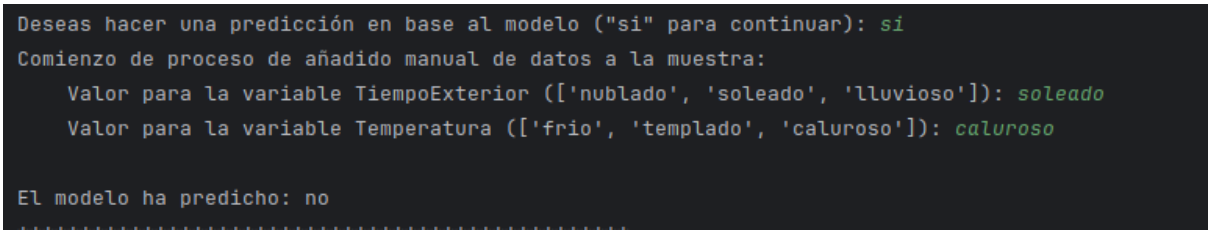


Imagen 13

Ejemplo con el dataset recortado (GUI):

Esta vez comenzaremos ejecutando el archivo **GUI_Main.py**. Una vez ejecutado se nos abrirá una ventana como la de la imagen 8.

En este caso modificaremos los **path** por defecto en pos de poner el de los archivos nuevos y le daremos a "Generar Árbol de Decisión". Con esto se nos construirá un árbol en la ventana de la derecha y se abrirá la posibilidad de realizar predicción (Imagen 14)

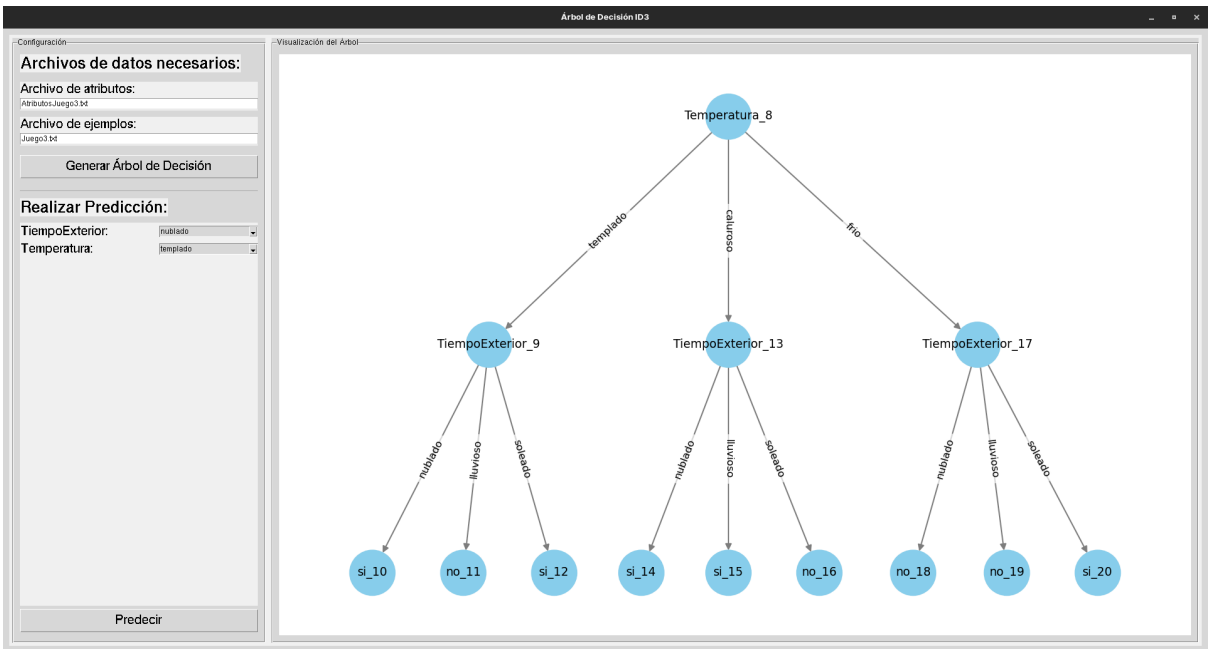


Imagen 14

Seleccionamos los mismos atributos que para la predicción anterior y ejecutamos pulsando el botón “Predecir”. Y como vemos en la predicción de la Imagen 15, esta concuerda con el árbol y con la predicción NoGUI.

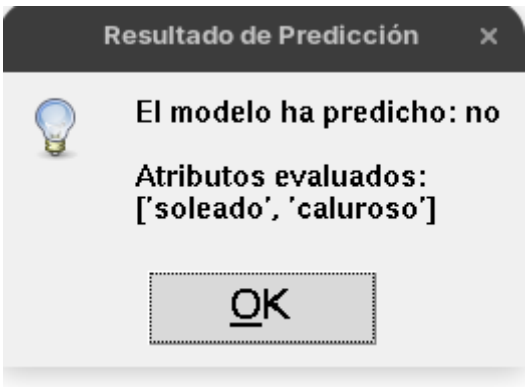


Imagen 15