PRÁCTICA 2

IMPLEMENTACIÓN DEL ALGORITMO ID3

Montserrat Sacie Alcázar

Asignatura: Ingeniería del Conocimiento

Ingeniería del Software

UCM, 2020

Índice

1. Introducción

El objetivo de la práctica 2 es implementar el Algoritmo ID3 que permita calcular un árbol de decisión dados como entrada un fichero con la lista de atributos y un fichero con la lista de ejemplos.

En la memoria se explican:

- En **el apartado 2,** los **detalles de la implementación del algoritmo**, así como de las ampliaciones implementadas y la **distribución de clases de código del proyecto** implementado en Eclipse.
- El **apartado 3** explica los **detalles de la interfaz gráfica para ejecutar la práctica**. En la entrega se añade un video explicativo de ejecución del programa para mayor claridad.
- Por último, en el **apartado 4** se explica la estructura necesaria que deben tener los ficheros de entrada de este programa y la solución obtenida tras la ejecución.

2. Detalles de Implementación del Algoritmo A*

El Algoritmo ID3 se ha implementado en el lenguaje de programación **Java** y usando el entorno de desarrollo **Eclipse**.

Las **funcionalidades básicas** implementadas han sido las explicadas en el enunciado de la práctica para poder ejecutar el algoritmo:

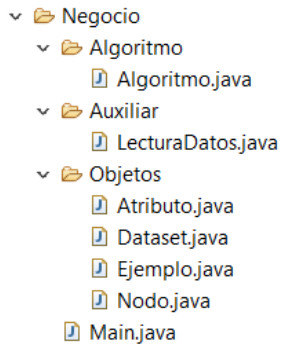
a) Lectura de ficheros (se seleccionan del directorio en el que los tengamos desde la interfaz gráfica).
b) Almacenamiento de los datos leídos en dos ArrayList, uno para los ejemplos y otro para los nombres de los atributos.
c) Cálculo para la primera iteración del atributo seleccionado o de menor mérito.
d) Reestructuración de los datos para la siguiente iteración.

Por otro lado, las ampliaciones **implementadas** son**:**

e) Implementación de todos los niveles de Recursividad.
f) Comprobar el correcto funcionamiento del algoritmo para los ejemplos de la tabla.
g) Comprobar el resultado esperado para un ejemplo (nuevo o incluido en la lista de ejemplos de entrada).
h) Obtención de las reglas a partir del árbol de decisión resultado de la ejecución del algoritmo.
i) Posibilidad de ejecutar este algoritmo con otra lista de atributos diferente y otros ejemplos que contengan valores para esos atributos, siempre que la variable de salida o resultado tome valores “si” y “no”.

Estructura del Proyecto

Negocio: clases con la implementación de la lógica del programa



- **Main.java**: clase principal a ejecutar por el usuario en Eclipse para lanzar el programa. Esta invoca a la ventana principal de la interfaz gráfica.

- **Atributo.java:** objeto Atributo que contiene:
  - El **nombre** del atributo
  - Un HashMap **valores** donde se guarda por cada valor distinto que puede tomar el atributo, el número de ejemplos de la lista que contienen ese valor.
  - Un HashMap **positivos** que almacena para cada valor distinto que toma el atributo, el número de ejemplos positivos (donde la variable resultado = "si")
  - Un int **numEjemplos** que guarda el número total de ejemplos de la lista de ejemplos de la subtabla
  - Un double **merito** para guardar su valor una vez calculado

  Es en la clase Atributo es donde se implementan los cálculos necesarios para obtener el mérito del atributo.

- **Ejemplo.java**: Objeto que contiene un HashMap con los pares (nombre de atributo, nombre del valor) que toma ese atributo. Tratamos los valores que puede tomar un atributo siempre como String.

- **Dataset.java:** Contiene un ArrayList de objetos Ejemplo, otro ArrayList con los nombres de los atributos (sin el nombre de la variable resultado) y un String nomResultado que sí guarda el nombre del atributo resultado o decisión.

- **Nodo.java**: objeto Nodo que corresponde a cada uno de los nodos del árbol de decisión. Los atributos que contiene son:
  - Un objeto de tipo Dataset **subtabla:** Este objeto guarda la "sub-tabla" de ejemplos para el cálculo del mérito de los atributos restantes tras haber seleccionado el atributo mejor en el nodo padre. (iteración anterior)

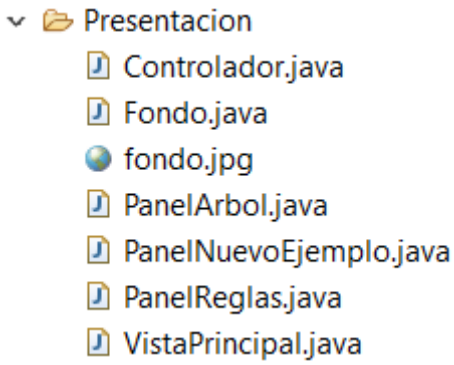
- Un ArrayList de **Atributos** inicializado dentro del método "inicializaAtributos", invocado dentro de la constructora de Nodo
- Un objeto Atributo **atributoMejor:** Inicialmente nulo**.** Como los méritos de cada atributo se inicializan dentro de la constructora de Nodo (invocación a inicializaAtributos), cuando se solicite su cálculo durante la ejecución del algoritmo (invocación a "CalculaAtributoMasInformativo") se recorre el ArrayList de atributos y se inicializa este campo con el atributo cuyo mérito es menor.
- Un arrayList de objetos Nodo **hijos.** Una vez obtenido el atributoMejor, se crea un nodo hijo por cada valor que pueda tomar ese atributo y selecciona los ejemplos que tienen cada uno de esos valores dando lugar a las subtablas.
- Un boolean **isHoja** que se será true si todos los ejemplos de la sub-tabla dan lugar al mismo resultado y por tanto no hay que seguir iterando sobre ese nodo. El resultado se guarda en el campo valor.

- **Algoritmo.java:** Los métodos implementados son:
  - **"execute":** método recursivo que incluye la implementación propiamente del Algoritmo ID3. Genera el árbol de decisión y devuelve el nodo raiz.
  - **"iniciarAlgoritmo":** método invocado por el Controlador, que recibe la lista de Ejemplos y de atributos. Este invoca a "execute".
  - **"calcularReglas":** Se recorre el árbol de decisión para generar un ArraList con las reglas (ArrayList de String).
  - **"recorridoEnProfundidad":** método invocado dentro de "calcularReglas" para recorrer el árbol durante el cálculo de reglas.
  - **"comprobarEjemplo":** Recorre el árbol de decisión comprobando qué rama contiene el valor para cada uno de los atributos del Nuevo ejemplo, hasta llegar a un resultado. En caso de no encontrar resultado devuelve un String vacío.

- **LecturaDatos.java:** Contiene dos métodos que reciben un String con la ruta de los ficheros .txt , los abren, los leen y devuelven un ArrayList de Ejemplos y un ArrayList de String para la lista de atributos respectivamente.

Presentación

Para la implementación de la interfaz gráfica se ha utilizado la herramienta *WindowBuilder* en Eclipse.

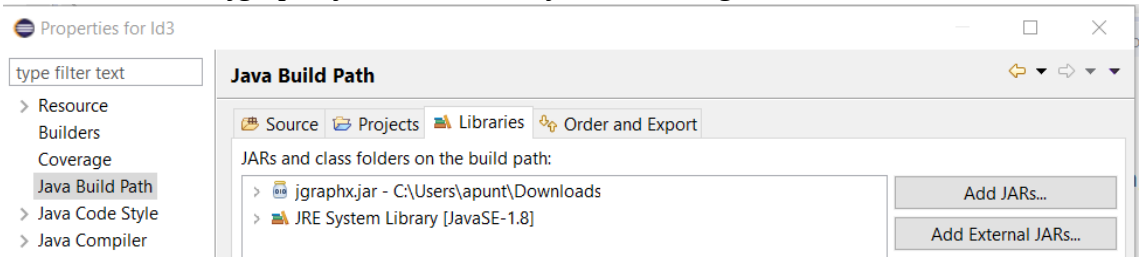


- **VistaPrincipal.java:** JFrame que contiene los paneles para la representación del árbol, el panel de Reglas y un panel para comprobar el resultado para un nuevo ejemplo.
- **PanelArbol.java:** Aquí se construye un grafo para representar el árbol de decisión obtenido por el algoritmo. Para su implementación se ha usado la librería "JGraphx" descargada e incluida en las librerías del proyecto (Comprobar que está ahí y si por lo que fuese no estuviera incluida; esta está disponible en el siguiente enlace: https://github.com/jgraph/jgraphx.)
- **PanelReglas.java:** Contiene un panel con un TextArea para escribir las reglas.
- **PanelNuevoEjemplo:** Contiene un textField en el que se puede escribir un ejemplo nuevo siguiendo el mismo formato que en los ficheros .txt proporcionados (los valores de atributos separados por coma). Aquí se invoca al Controlador cuando el usuario selecciona el botón comprobar, obteniendo el resultado para este ejemplo.
- **Controlador.java:** clase que comunica Negocio con Presentación.
- **Fondo.java**: JPanel cuya única utilidad es colocar una imagen de fondo. Sobre él se incluirán el resto de paneles anteriormente explicados.

3. Manual de Usuario

Importar el proyecto Id3 en Eclipse

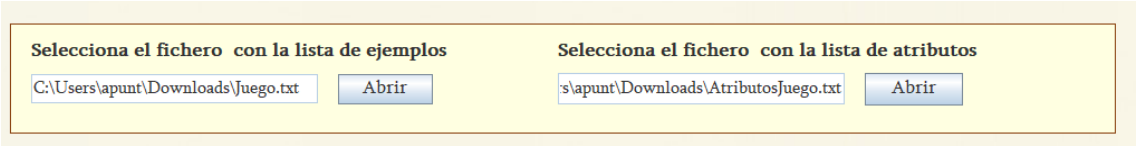
Incluir la librería jgraphx.jar. Esta se incluye en la entrega.

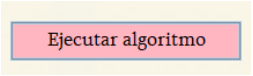


Arrancar la aplicación pulsando Run colocados sobre Main.java

Nos aparece:

Primero cargamos el fichero *juego.txt* que contiene la lista de ejemplos y *lista.atributos.txt.*

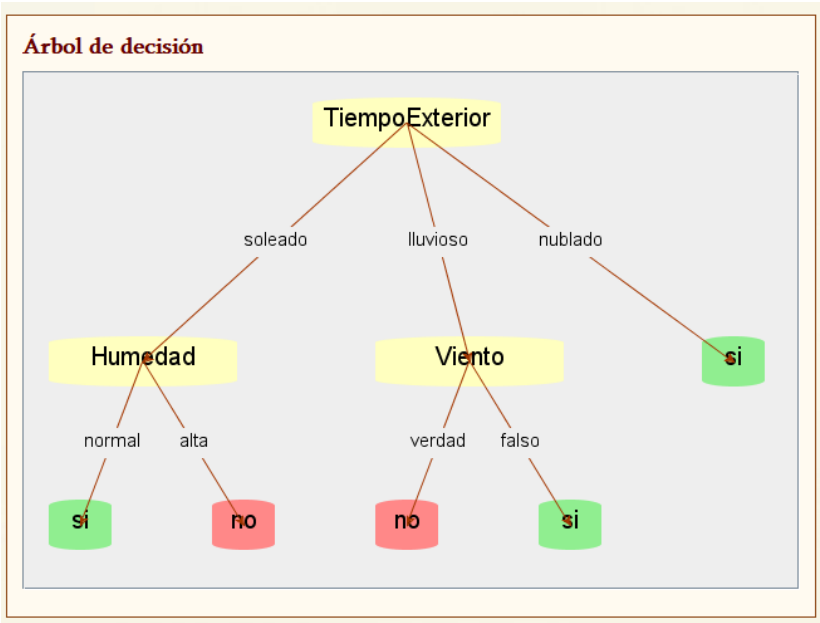


Pulsar 

Si los ficheros no contienen los ejemplos y atributos en un formato adecuado o no hemos abierto alguno de los ficheros o los dos; nos volverá a pedir que seleccionemos los ficheros.

Panel con el árbol de decisión obtenido:

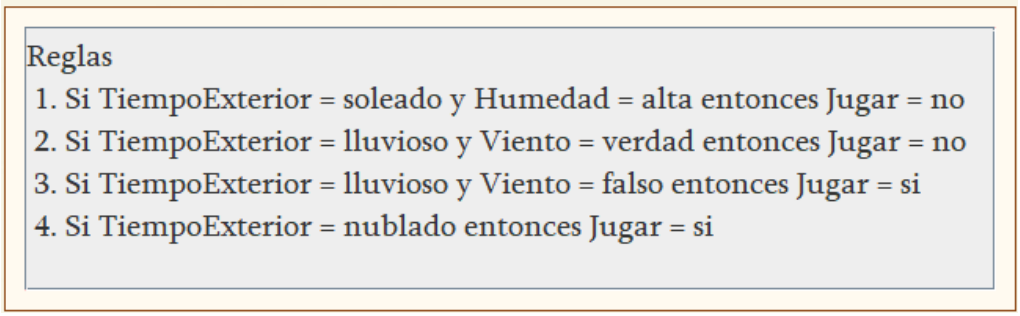
Justo bajo el panel de selección de ficheros aparece



En cada vértice se escribe el atributo escogido, en las ramas los valores distintos que puede tomar y en los nodos hoja el valor resultado alcanzado por cada camino.

Panel con las reglas

Bajo el panel que contiene el grafo aparece un JScrollPane con las reglas extraídas del árbol de decisión:

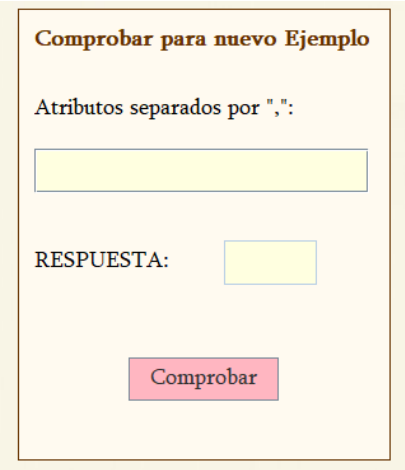


Si no se llegó a una solución, continuará vacío.

En consola se escriben los méritos calculados en cada nodo asociados a cada atributo. Cada iteración corresponde a un nivel del árbol.

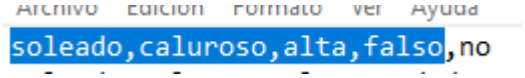
```
Iteración 0
----------------
TiempoExterior = 0.6935361388961918 (menor mérito)
Temperatura = 0.9110633930116763
Humedad = 0.7884504573082896
Viento = 0.8921589282623617

#################
Iteración 1
Rama ->TiempoExterior = soleado
---------------
Temperatura = 0.4
Humedad = 0.0 (menor mérito)
Viento = 0.9509775004326937

#################
Iteración 1
Rama ->TiempoExterior = lluvioso
---------------
Temperatura = 0.9509775004326937
Humedad = 0.9509775004326937
Viento = 0.0 (menor mérito)

#################
```

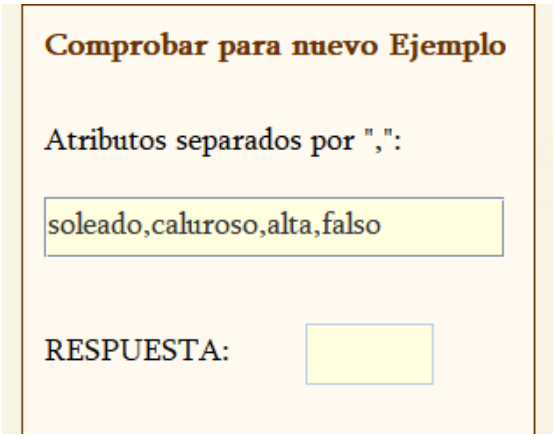
Panel para comprobar el resultado para un nuevo ejemplo o para los ejemplos de la lista:



Este panel se encuentra a la derecha del panel del Árbol.

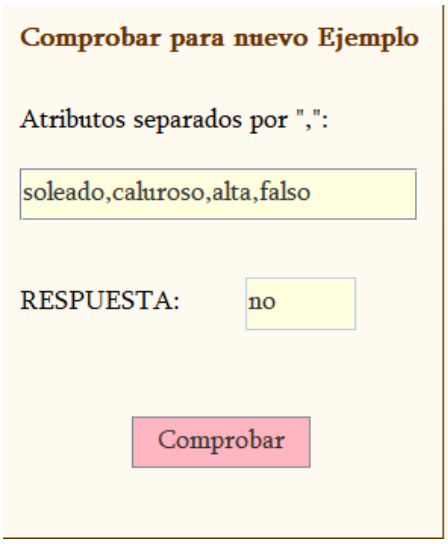
Introducimos uno de los ejemplos del fichero *juego.txt* sin el resultado, para comprobar que efectivamente nos da la respuesta que debe.





Pulsamos 

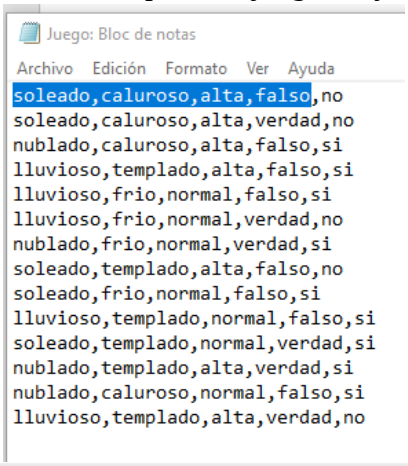
. En caso de no haber escrito nada en los atributos o en caso de intentar comprobar el resultado para un ejemplo, antes de ejecutar el algoritmo; nos saldrá un panel emergente informándonos de que no podemos hacer eso. Como no es el caso, nos aparece:

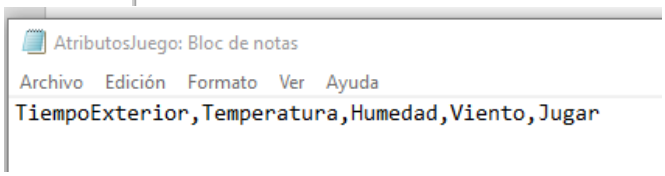


4. Ficheros de entrada para la ejecución

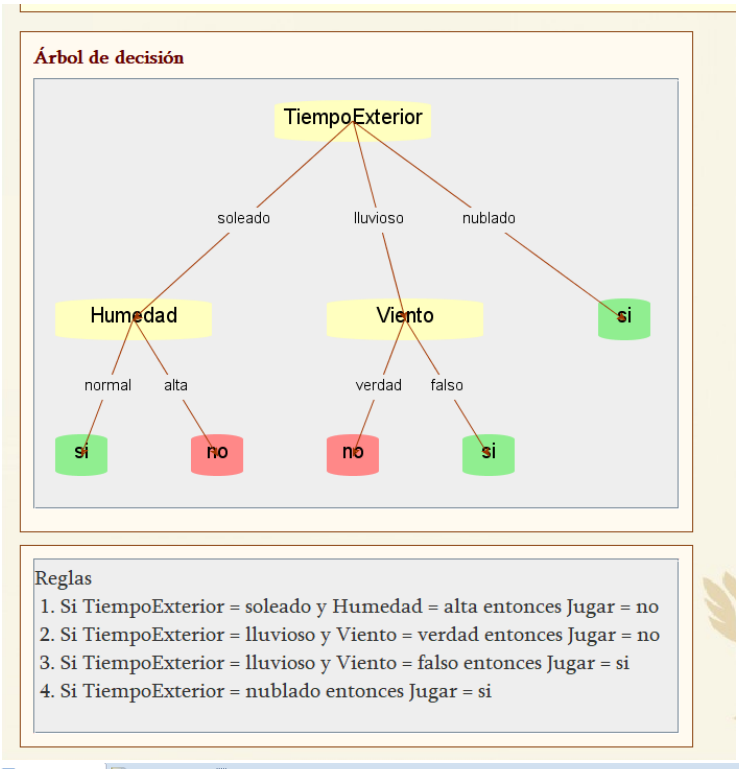
Se ha ejecutado el algoritmo dos veces

1. Con los ficheros entregados en esta práctica *juego.txt* y *lista.atributos.txt*





Resultado

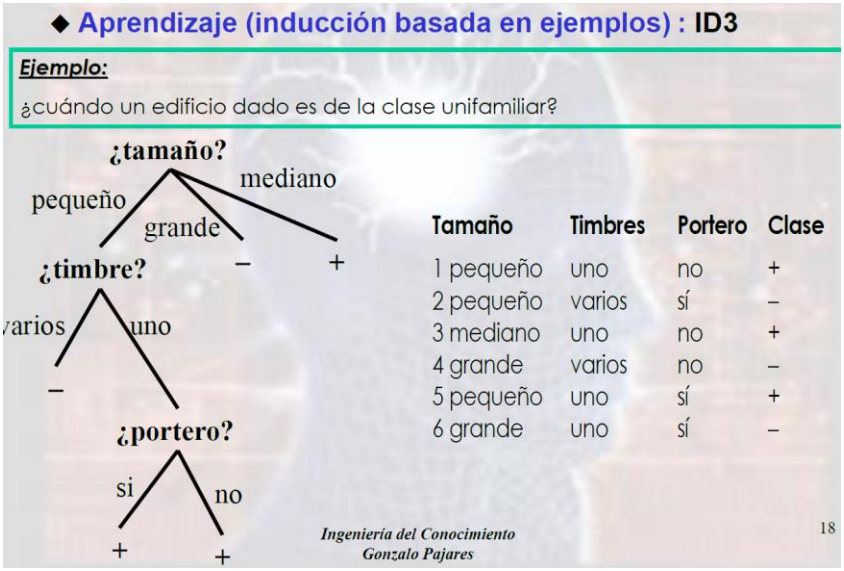


Console ⊠ Problems Debug Shell

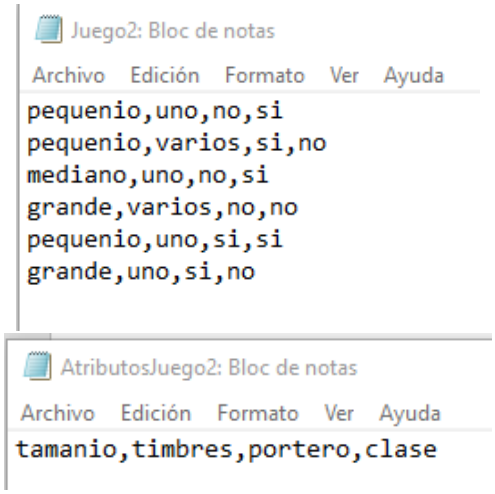
Main (3) [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (25 may. 2020 21:11:43)

```
Iteración 0
----------------
TiempoExterior = 0.6935361388961918 (menor mérito)
Temperatura = 0.9110633930116763
Humedad = 0.7884504573082896
Viento = 0.8921589282623617

################
Iteración 1
Rama ->TiempoExterior = soleado
----------------
Temperatura = 0.4
Humedad = 0.0 (menor mérito)
Viento = 0.9509775004326937

################
Iteración 1
Rama ->TiempoExterior = lluvioso
----------------
Temperatura = 0.9509775004326937
Humedad = 0.9509775004326937
Viento = 0.0 (menor mérito)

################
```

2. Se ha creado un fichero *juego2.txt* y *lista.atributos2.txt* que contiene los ejemplos de la diapositiva 18 del tema 4 II.

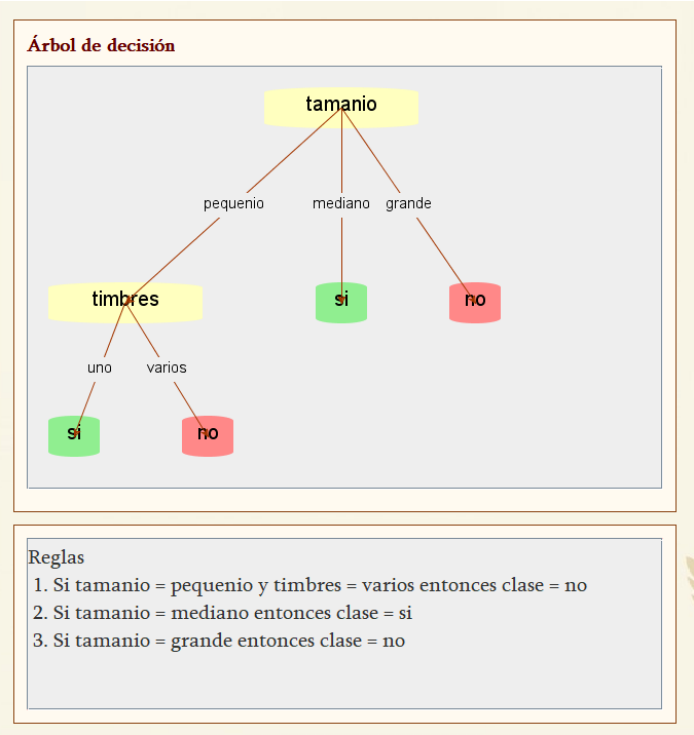


Quedándonos



Este algoritmo ha sido implementado para poder recibir cualquier lista de atributos que tomen una serie de valores cada uno con la única condición de que la variable de salida tome valor “si” o “no”.

Resultado:




```
Console  Problems  Debug Shell
Main (3) [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (25 may. 2020 21:11:43)
Iteración 0
----------------
tamanio = 0.4591479170272448 (menor mérito)
timbres = 0.5408520829727552
portero = 0.9182958340544896

#################
Iteración 1
Rama ->tamanio = pequenio
----------------
timbres = 0.0 (menor mérito)
portero = 0.6666666666666666

#################
```

Se incluye en la entrega un video que muestra la ejecución de estos ejemplos.