Métodos de Aprendizaje Automático
Práctico 2

Bruno Garate & Nicolás Izquierdo & Guillermo Siriani

1. Ejercicio 6 - ID3

1.1. Implementación

Se diseñó una solución que fuese flexible para experimentar. Por lo tanto, se separaron la obtención de los datos, la ejecución del algoritmo y la aplicación de las validaciones en diferente componentes. Se eligió una representación genérica del árbol y los atributos. Los principales elementos del diseño se describen en los siguientes párrafos.

Una entrada de datos se representa mediante un diccionario de atributos y sus valores. Los archivos CSV se representan como una lista de diccionarios. Para construir las entradas del algoritmo se registran los atributos de interés (sus nombres), junto a una función de mapeo. Esta función permite construir representaciones diferentes sobre los datos, por ejemplo, mapear el atributo *G3* a un atributo con los valores *Aprobado - Reprobado*.

A continuación, se le pasan los datos transformados, y el nombre del *atributo objetivo* al ejecutor del algoritmo *ID3*. La ejecución del algoritmo retorna un árbol de decisión anotado. Además de la información del atributo seleccionado en los nodos del árbol, la etiqueta del mismo en las aristas y el valor del *atributo objetivo* en las hojas, se incluyó información de interés para uso posterior en estadísticas.

Se incluyó un componente adicional que permite renderizar los árboles construidos para su visualización. Este componente requiere de *graphviz* instalado, además del paquete python del mismo nombre.

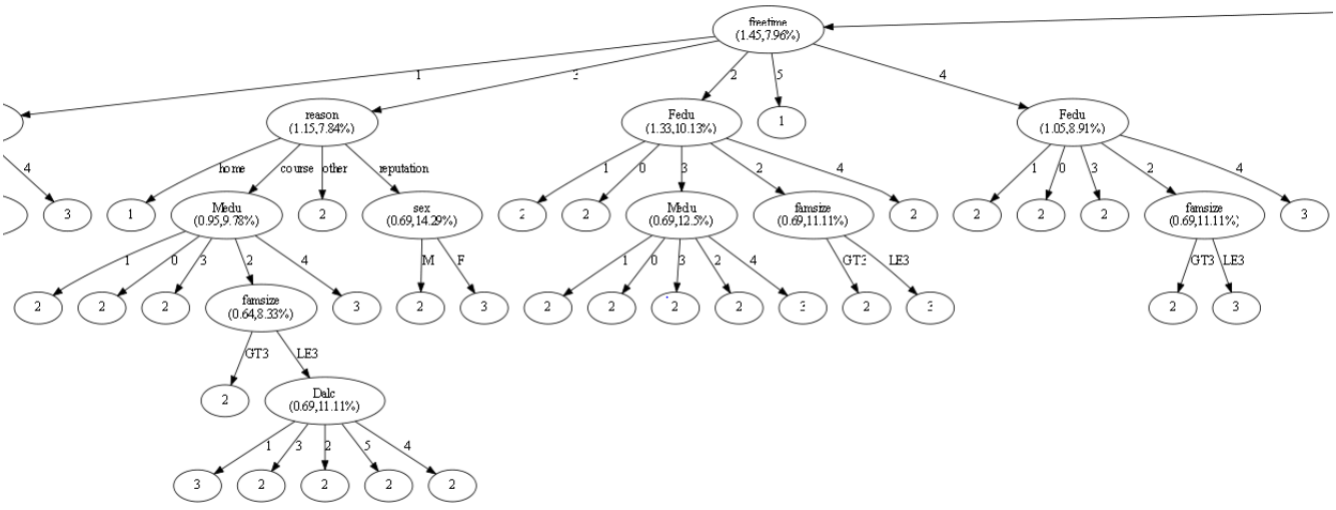

Figura 1: Fragmento de un árbol generado por el algoritmo implementado y renderizado en *graphviz*

Para poder validar la implementación del algoritmo se realizó una prueba contra un conjunto de datos de prueba conocido. Los datos utilizados parten del *dataset* del clima. Se corrió tanto el algoritmo original así como una versión modificada del algoritmo que minimiza la ganancia de información, y de esta forma poder comparar el comportamiento esperado.

1.2. Modelado del espacio

Se permitió la transformación de atributos para mejorar la adaptación del algoritmo. Por ejemplo, el *atributo objetivo G3* se representó como un entero en el rango [0,4]. Esto disminuye el

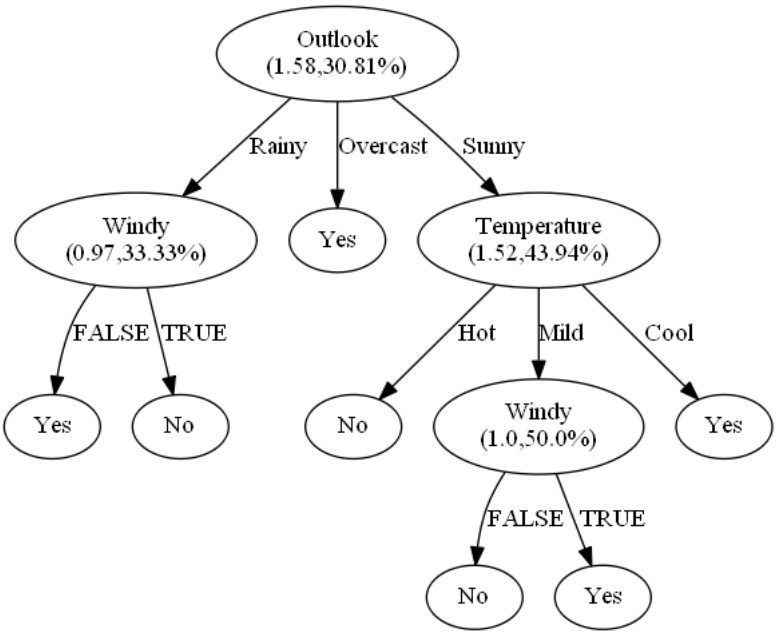

Figura 2: Árbol generado por el algoritmo contra el *dataset* de mock de datos clímaticos

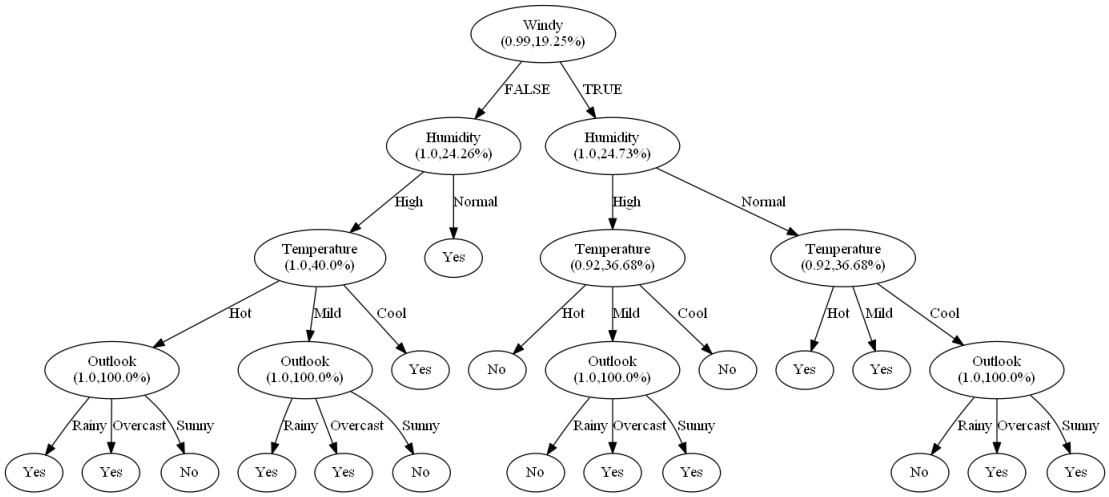

Figura 3: Árbol generado por una versión del algoritmo modificado contra el *dataset* de mock de datos clímaticos

espacio de búsqueda del algoritmo. Además, consideramos que es válido argumentar que lo que se busca es aproximar el comportamiento de los estudiantes en el curso (por ejemplo Deficiente, Regular, Bueno, Muy bueno) más que encontrar un valor especifico.

Además, también se convirtieron los valores de entrada, por ejemplo, particionando el atributo de relación familiar en tres. Es importante tomar en cuenta que la partición fue uniforme y no consideró la distribución de los valores. Sería posible, por ejemplo, si suponemos que G3 sigue una distribución gaussiana, tomar particiones que tomen los valores a diferentes intervalos de σ^2.

Es importante notar que a medida que los valores de resultado se particionan, se vuelven cada vez más probables. Por lo tanto, así como el árbol de decisión se hace más *exacto*, se hace menos *preciso*. Por eso durante nuestras pruebas comparamos las performance del árbol contra la selección 'al azar' de un resultado.

Ocurre un efecto similar en los atributos de entrada. La partición de estos atributos se debe

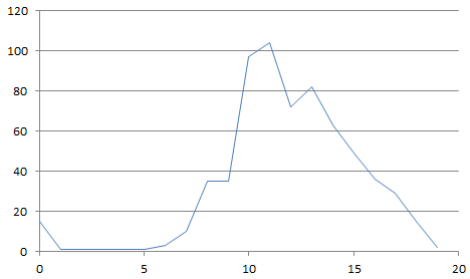

Figura 4: Densidad de los valores de G3, antes de la partición

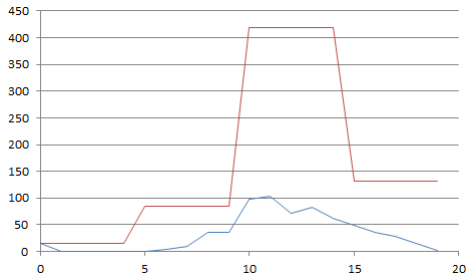

Figura 5: Densidad de los valores de G3 y densidad de la partición comparados

manejar cuidadosamente, ya que pese a mejorar la exploración del espacio de soluciones, puede disminuir la precisión y el aporte del atributo sobre la respuesta. También la selección de los límites de las particiones es importante, ya que las mejoras obtenidas en el particionamiento de las etiquetas de los atributos se pueden perder.

A la salida se le incorporó información estadística relevante para el estudio como las distancia máximas y mínimas entre la raíz y las hojas y la relación entre hojas y ejemplos para diferentes tipos de hojas.

1.3. Casos y poda

Se calcularon los valores posibles de un atributo a partir del total de ejemplos de forma de considerar estos valores aún cuando no apareciesen en los ejemplos locales durante la construcción del árbol.

Cuando un *valor posible* de un atributo no se encuentra en los ejemplos, se considera el valor del *atributo objetivo* más común en los ejemplos del árbol al que se le están construyendo los hijos. De esta forma se asume que a falta de mayor información sobre la probabilidad condicional, se toma la probabilidad local (condicionada al sub árbol en que se encuentra) de la ocurrencia del evento.

Por eso se incorporó una *poda verde* del árbol, dónde se remplazan los nodos con hojas que poseen el mismo valor para el atributo objetivo, por hojas con este valor, procediéndose con el remplazo de forma *bottom-up*.

Se ve en la figura como la poda puede ayudar a reducir la profundidad del árbol, sin modificar su comportamiento. Los ejemplos presentaban múltiples valores para el *atributo objetivo*, dado el mismo vector de atributos de entrada, por lo que se generó una cadena que consumió todos los atributos antes de producir una hoja mediante uno de los casos base (el de mayoría específicamente).

Al construirse las hojas se los anotó con una propiedad que toma uno de cuatro posibles valores: *único*, *mayoría*, *else* o posteriormente *poda*. Único refiere al caso base del algoritmo ID3 al alcanzar una lista de ejemplos con un mismo valor en el atributo objetivo. Mayoría se emplea en otro de los

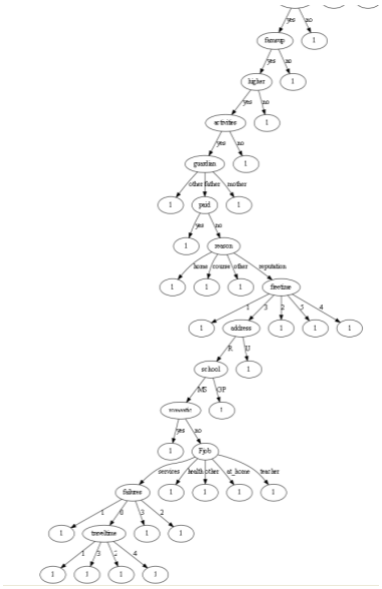

Figura 6: Ejemplo de una rama a la que se la aplicaría la poda

casos base, dónde existen múltiples valores posibles del atributo objetivo, pero no se encuentran más atributos disponibles (o se alcanzó un criterio de parada del algoritmo) y se etiquete a la hoja del árbol con el valor más común. Else responde a los hijos que *saturan* el árbol de forma que quede completo al recorrer, como fue comentado en párrafos anteriores. El valor *poda* se emplea para identificar los nodos que se obtienen de realizar la poda *bottom-up* descrita en el punto anterior.

1.4. Validación cruzada

Se selecciona cuatro quintos del conjunto de datos *D* y se procede a aplicar una validación cruzada, dividiendo la partición seleccionada en 10 conjuntos disjuntos e iterando sobre ellos de tal forma que en cada iteración el conjunto actual *Ti* se conserva como caso de test mientras que los restantes 9 conjuntos *(D - Ti)* se utilizan para la generación de un árbol de decisión. Luego se valida el árbol resultante con *Ti* y se calcula el error obtenido.

Finalmente se promedian los errores luego de la iteración y se obtiene un *error estimado*. Si bien el método de validación cruzada puede ser fácilmente usado de forma errónea, en el caso de estudio se argumenta que los conjuntos de prueba son extraídos de una misma población, brindando así mayor seguridad sobre la veracidad del resultado.

1.5. Aplicación del algoritmo

Las pruebas se realizaron sobre el conjunto de datos de estudiantes de portugués (por ser el mayor de los dos) con atributo objetivo “G3”. Los valores posibles de atributo objetivo se transformaron a 4 valores posibles. Para la construcción del árbol se tuvieron en cuenta todos los atributos a excepción de “G1”, “G2” y “age”. Para los siguientes atributos (originalmente numéricos de 1 a 5) particionamos sus valores en 3 valores posibles:

- "freetime"
- "goout"
- "Dalc"
- "Walc"
- "health"

Los posibles valores del atributo “absences” se pariticionaron en 4 partes iguales (entre 0 y 40). Para el resto de los atributos se mantuvieron sus valores originales.

Para tener una buena estimación de la eficiencia del algoritmo realizamos varias pruebas con diferentes combinaciones de datos de entrenamiento y de prueba. Además de la validación cruzada, validamos contra el 1/5 de casos de prueba tomando distintos valores máximos de profundidad del árbol. Los porcentaje de acierto del árbol para las distintas pruebas fueron los siguientes:

- Validacion Cruzada: % 53.0980392157
- Sin límite de profundidad: % 52.7441860465
- Límite de profundidad 1: % 64.2170542636
- Límite de profundidad 2: % 64.8062015504
- Límite de profundidad 3: % 62.8837209302
- Límite de profundidad 4: % 60.4031007752
- Límite de profundidad 5: % 54.1085271318
- Límite de profundidad 6: % 52.7751937984
- Límite de profundidad 7: % 52.7441860465

1.6. Observaciones

A modo anecdótico podemos afirmar que al cambiar el signo de la entropía y realizar las mismas pruebas, los porcentajes de acierto se incrementaron levemente. La causa de este fenómeno nos es incierta quitando el hecho de que pudiera tratarse de un fallo en el código. Es interesante de todas formas notar que los árboles generados por el algoritmo modificado que emplea los atributos con la menor ganancia de información son mucho más profundos y frondosos, como cabe esperar a partir de la minimización de la ganancia de la información.

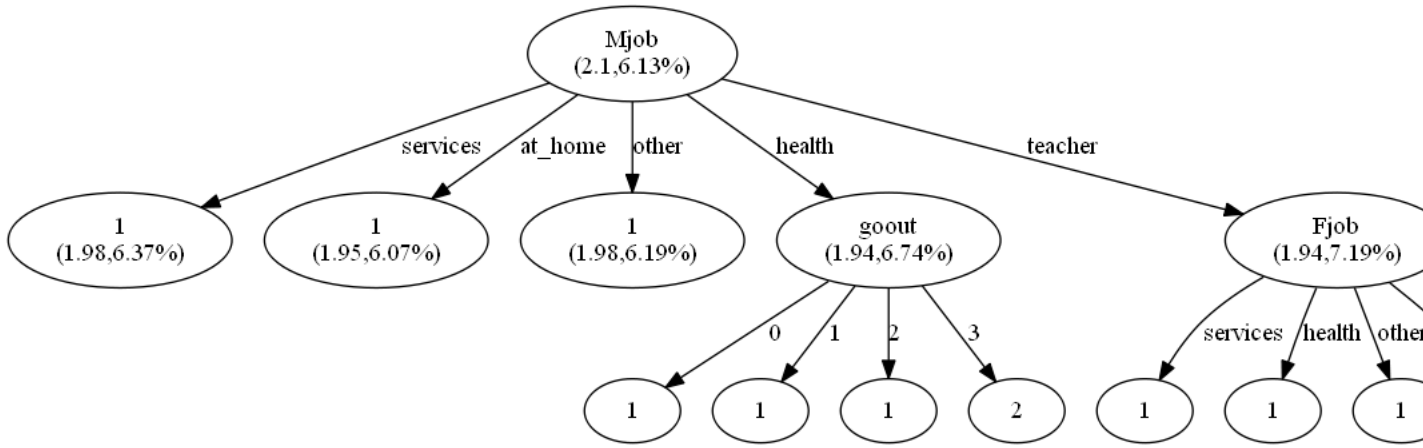

Figura 7: Árbol generado con profundidad máxima 2

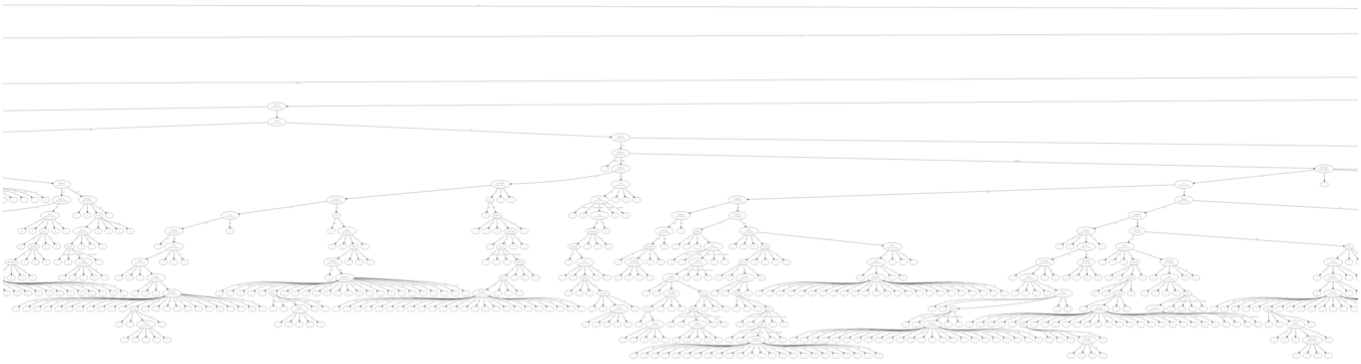

Figura 8: Fragmento del árbol resultante de la minimización de la ganancia de la información

Validacion Cruzada: 61.5529411765
Evaluacion sin poda: 60.2170542636
Evaluacion profundidad 1: 65.4573643411
Evaluacion profundidad 2: 65.4573643411
Evaluacion profundidad 3: 65.4573643411
Evaluacion profundidad 4: 65.3023255814
Evaluacion profundidad 5: 65.1162790698
Evaluacion profundidad 6: 64.5891472868
Evaluacion profundidad 7: 64.4031007752
Evaluacion profundidad 8: 61.3333333333
Evaluacion profundidad 9: 63.503875969

Es importante notar que la ecuación de ganancia de información posee un *bias* hacia los atributos con muchos atributos. Hay literatura que sugiere el uso de la siguiente ecuación, llamada *Ratio de Ganancia*, que compensa los atributos con muchos valores.

$$GainRatio(T, X) = \frac{Ganancia(T, X)}{Split(T, X)}$$

$$Split(T, X) = -\sum_{c \in A} P(c) log P(c)$$

1.7. Conclusiones

Los resultados obtenidos si bien han sido favorables, es decir, los porcentajes de acierto de parte del árbol de decisión superaron el 50 %, no han cumplido con nuestras expectativas. Probablemente este resultado se deba a la poca cantidad de ejemplos disponibles tomando en cuenta la cantidad de combinaciones de atributos posibles de las muestras. En este caso, 1 de las 4 particiones que tomamos para los posibles valores del atributo objetivo abarca la amplia mayoría de los ejemplos, lo cual creemos que permite que los árboles con poca profundidad adivinen fácilmente a que partición pertenecen los otros ejemplos. Luego, parece haber un sobre-entrenamiento sobre un conjunto de datos que no es lo suficientemente grande para ser representativo de todas las combinaciones posibles.

Referencias

[1] Data Warehousing and Data Mining (DWDM), *Lesson 5*, http://www.inf.unibz.it/dis/teaching/DWDM/slides2011/lesson5-Classification-2.pdf

[2] Maschinelles Lernen: Symbolische Ansätze, *Decision-Tree Learning*, http://www.ke.tu-darmstadt.de/lehre/archiv/ws0809/mldm/dt.pdf

[3] Earl Harris Jr., *Information Gain Versus Gain Ratio: A Study of Split Method Biases*, http://www.mitre.org/sites/default/files/pdf/harris_biases.pdf