IMPLEMENTACIÓN DE ÁRBOLES DE DECISIÓN EN LA PREDICCIÓN DEL ÉXITO ACADÉMICO

Stiven Yepes Vanegas
Universidad Eafit
Colombia
esyepesv@eafit.edu.co

Sara Rodríguez V
Universidad Eafit
Colombia
srodriguev@eafit.edu.co

Mauricio Toro
Universidad Eafit
Colombia
mtorobe@eafit.edu.co

RESUMEN

A lo largo de nuestras vidas nos sometemos a distintas pruebas en ámbitos como el educativo o académico. Las pruebas de estado saber 11 y saber pro son un gran ejemplo de estas. El éxito en tales pruebas es un factor muy importante para los estudiantes y las instituciones educativas, pues estas dan una perspectiva del nivel académico que se tiene tanto a nivel personal como grupal, por lo tanto, trataremos de predecir este éxito a través de árboles de decisión.

La solución que se plantea es la construcción de un árbol de decisión para ayudar a predecir el éxito académico de los estudiantes de Eafit para las pruebas saber Pro.

Se llega a la conclusión de que el principal factor de predicción para los resultados del saber pro son los resultados de la prueba saber 11, especialmente en el área de inglés.

Palabras clave

- Arboles de decisión.
- Algoritmos.
- Éxito académico.
- Predicción.

Palabras clave de la clasificación de la ACM

- Theory of computation → Design and analysis of algorithms → Data structures design and analysis → Pattern matching

1. INTRODUCCIÓN

En la actualidad se cuenta con grandes bases de datos en las instituciones, como lo es la base de datos de la universidad EAFIT, sin embargo, éstos son poco útiles si solo se almacenan y no se les relaciona entre sí para encaminarse a responder interrogantes más apremiantes, como lo es la probabilidad de éxito académico, problemática que se planea abordar a través de la creación de árboles de decisión.

2. PROBLEMA

El problema del éxito académico cuenta con muchas variables sociales, económicas, políticas, entre otras, pero a través de la tecnología, y en especial del aprendizaje supervisado aplicado en árboles de decisión, se puede buscar de manera más directa una relación entre los datos recogidos por el ICFES y la probabilidad de éxito académico de un estudiante de la Universidad EAFIT para hacer predicciones útiles que permitan crear medidas preventivas y de acompañamiento.

3. TRABAJOS RELACIONADOS

3.1 ID3

Existen diferentes métodos para construir un árbol de decisión, entre ellos está el algoritmo ID3, desarrollado por J. Ross Quinlan en 1983, cuyo nombre hace referencia a *Induction Decision Tree*. Hace parte de los algoritmos TDIDT *(Top-Down Induction of Decision Trees*).

Este algoritmo se basa principalmente en la entropía o probabilidad de ocurre ncia de un evento, la cual es utilizada para dividir un conjunto de datos en subgrupos más pequeños de forma recursiva. Para calcular la entropía utilizamos la siguiente formula:

$$\text{Entropia}(S) = \sum_{i=1}^{n} -p_i \log_2 p_i$$

Donde S es una colección de objetos, Pi es la probabilidad de los posibles valores e i las posibles respuestas de los objetos. Este algoritmo también nos introduce al concepto de ganancia de información, que es la que indica el siguiente atributo que dividirá el conjunto de datos. Su fórmula es la siguiente:

$$\text{Gan Inf}(S, A) = \text{Entropia}(S) - \sum_{v \in V(A)} \frac{|Sv|}{|S|} \text{Entropia}(Sv)$$

Donde S es una colección de objetos, A son los atributos de los objetos y V(A) el conjunto de valores que A puede tomar. Los principales componentes de este árbol son los nodos, que identifican los atributos; las ramas, que son los posibles valores con relación al nodo y las hojas, que son los conjuntos ya clasificados.

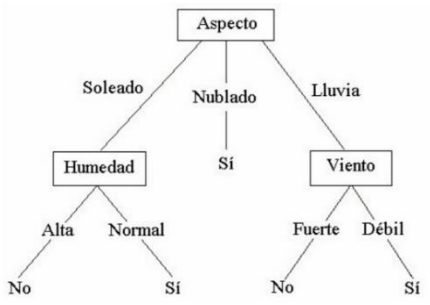


Figura 1: Ejemplo de un árbol con ID3.

Ver referencia 1.

En el ejemplo anterior vemos se subdividen los atributos de la mejor manera, esto lo hace siguiendo la siguiente secuencia: primero calcula la entropía de cada atributo, después divide el conjunto original en subconjuntos con menor entropía y crea nodos de estos, hace esto varias veces recursivamente y el resultado final son las hojas con las etiquetas o clasificaciones del problema en cuestión.

Algunos inconvenientes que tiene este algoritmo son el favoritismo por aquellos atributos que no necesariamente son los más útiles, la generación de grandes árboles y que solo es aplicable a problemas de clasificación y diagnóstico.

3.2 C4.5

El algoritmo C4.5 es una mejora hecha por J. Ross Quinlan al algoritmo ID3 en 1993, y como este hace parte de la familia de los algoritmos TDIDT. Al igual que su antecesor, Este algoritmo tiene los mismos componentes del ID3 y hace uso de la entropía y la ganancia de información, pero ahora acompañada de la estrategia de profundidad-primero (*depth-first*). La gran similitud que hay entre estos dos algoritmos se puede ver en el siguiente gráfico:

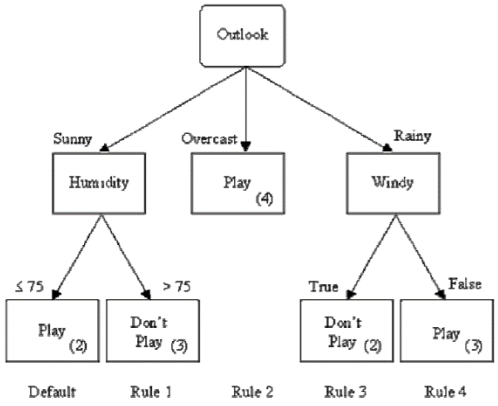


Figura 2: Ejemplo de un árbol con C4.5.

Ver referencia 2

Aquí se puede apreciar como esta versión mejorada del algoritmo no solo considera valores discretos, sino que también considera valores continuos. Para los atributos discretos se considera una prueba con todos los posibles valores que puede tomar el atributo y para los atributos continuos se realizaba una prueba binaria sobre estos datos.

3.3 CART

El algoritmo CART extrae su nombre *de Classification and Regression Trees*, término acuñado por Leo Breiman, es de gran relevancia y sirve de base para algoritmos posteriores como *Random Forest* o *Bagged Decision Tree*.

Se representa con un árbol binario, donde cada raíz tiene una variable (x), que se divide en dos ramas a través de una sentencia, normalmente numérica, a la que se le asigna un valor booleano (*True or False*) cada rama posteriormente se divide de la misma forma. Las hojas contienen una variable (y) que es usada para hacer una predicción.
En este árbol aparecen conceptos como el *índice de Gini*, el cual determina el índice de pureza en la separación de los datos en nodos. Este en un nodo *t* se determina con la siguiente formula:

$$g(t) = \sum_{J \neq i} p(j/t)\, p(i/t)$$

donde i y j son las categorías de la variable predictora y p es proporción.

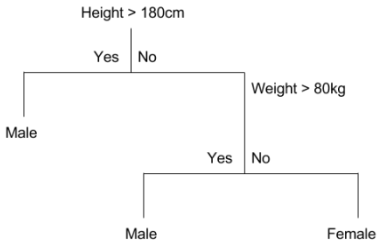


Figura 3: Ejemplo hipotético de árbol usando CART.

Ver referencia 3.

En el ejemplo descrito en la gráfica se evidencia la naturaleza booleana de los atributos de un árbol CART, y se evidencia la importancia de hacer una buena elección de variables para evitar generalizaciones erróneas, como que una persona que mida más de 180 cm siempre será hombre.

Para construir un modelo CART se debe seleccionar y adecuar los datos de entrada y separar los resultados de esas variables hasta que se cree un árbol adecuado. La selección de que variable usar y el punto de corte o separación se realiza a través de un *Greedy Algorithm* para minimizar el costo de correr el árbol.

3.4 CHAID

El acrónimo CHAID se refiere a *Chi-squared Automatic Interaction Detector*. Es uno de los métodos de clasificación propuesto por Kass, y se considera un descendiente de THAID. Este algoritmo construye árboles no binarios, a través del uso de un algoritmo bastante útil para el análisis de *data sets* de gran tamaño llamado Chi Square, el cual está dado por la fórmula: **((Real – Esperado)^2 / Esperado)^1/2**

El hecho de que trabaje con tablas de frecuencia multidireccional lo vuelve muy popular en el ámbito del mercadeo, en especial de estudios de segmentación de mercado.

El algoritmo crea vaticinadores categóricos dividiendo distribuciones continuas en un conjunto de categorías. Entonces el algoritmo itera los vaticinadores realizando pruebas, mezclando categorías y ajustando valores, escogiendo las ramas más significativas hasta el momento que no se puedan realizar más divisiones.

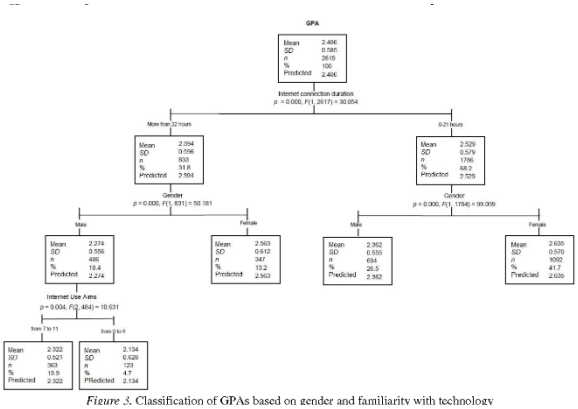


Figure 3. Classification of GPAs based on gender and familiarity with technology

Figura 4: Ejemplo de árbol usando CHAID.

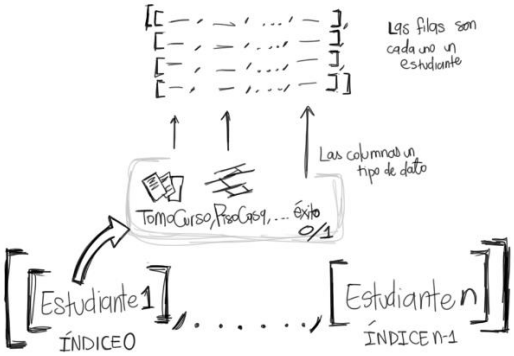
Ver referencia 5.

En el anterior ejemplo se evidencia un árbol CHAID en acción en un ambiente similar al que se efectuará en el proyecto en cuestión, analizando hábitos de estudiantes universitarios. Se aprecia que hay más variables involucradas que en el algoritmo analizado anteriormente, se ve las probabilidades estadísticas asociadas al vaticinador de cada rama, junto con su desviación.

Un principal inconveniente es el tamaño que los árboles pueden llegar a tener, no por un motivo computacional, pero de interpretación humana posterior de los árboles creados. También que necesita un volumen considerable de datos para funcionar correctamente.

4. ARREGLO DE OBJETOS NUMPY

La estructura en la cual se almacenarán los datos es un arreglo numpy en python de tipo object dtype.

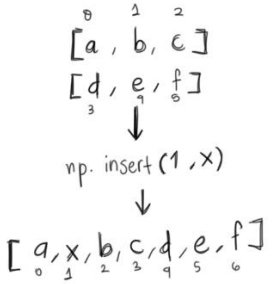


Gráfica 1: Arreglo numpy con datos de personas agrupados en cada element (element) de este.

4.1 OPERACIONES DEL ARREGLO NUMPY

Inserción de datos: np.insert(arreglo, obj,valores,eje)

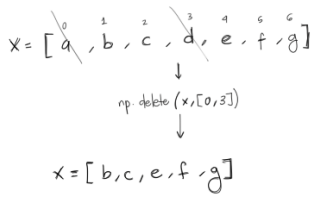
Se insertan los datos, recibe parámetros como el arreglo al que se aplica el cambio, el objeto que apunto a un subconjunto, los valores a insertar el arreglo y por último el eje seleccionado. Ejejmplo: np.insert(a, [1], [[1],[2],[3]], axis=1)



Gráfica 2: Inserción de un dato x en un arreglo de letras indicando solo la posición.

Borrado: numpy.delete(arr, obj, axis)

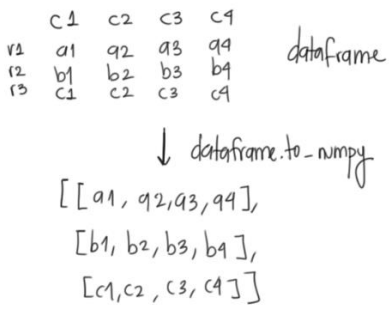
Este comando elimina las columnas o filas del arreglo a las cuales se apunte. Recibe el arreglo, el objeto que apunta a un subconjunto del arreglo y un eje de borrado. Ejemplo:arr = np.delete(arr, 2). Este comando elimina el ítem en el índice 2



Gráfica 3: Borrado de 2 datos de un arreglo de letras, indicando la posición de borrado.

Conversión de DataSet a arreglo de numpy: pandas.DataFrame.to_numpy(dtype,copy)

Este comando convierte un dataset de pandas en Python en un arreglo numpy. Tiene como argumentos el array a convertir, un tipo dtype y un booleano de si el resultado es una copia. Ejm: arr = df.to_numpy()



Gráfica 4: Conversión de un formato de pandas dataset a un arreglo numpy.

4.2 Criterios de diseño de la estructura de datos

Se decide usar Python gracias a las múltiples librerías y funciones afines al trabajo a ser realizado con los árboles de decisión.

Se realiza la lectura del archivo csv con Pandas, para garantizar un cortado certero de los datos, y por las facilidades que tiene para rellenar los datos sin contestar con el valor "desconocido" y prevenir así anomalías en la posterior construcción del árbol de decisión.

Luego este archivo se convierte a la estructura de array numpy, porque es mucho más liviano para trabajar los datos, consume menor memoria, ideal teniendo en cuenta la cantidad de estudiantes que tienen los sets de training data.

La razón de usar esta estructura es la habilidad para llamar los datos por categorías como columnas y filas y que es más liviano que el panda's dataframe, una opción similar.

En un punto también se consideró usar la opción de Listas de Python en vez del arreglo para guardar los datos leídos, la cual fue descartada ya que consume más memoria y tiene un comportamiento en tiempo de ejecución más lento.

4.3 Análisis de Complejidad

Método	Complejidad
Función view (Ver elemento)	O(1)
Crear array vacío (numpy.empty)	O(1)
Crear array con ceros (numpy.zeros)	O(n)
Crear array multidimensional con ceros (numpy.zeros)	O(nm), n= filas, m=columnas
Función Copy (Copiar array)	O(n)
Función Copy (Copiar array multidimensional)	O(nm) n= filas, m=columnas

Tabla 1: Tabla para reportar la complejidad.

4.4 Tiempos de Ejecución

Se miden los tiempos que toma hacer la lectura usando Panda's dataframe, la operación de construcción del arreglo numpy y por último una medición general de ambos procedimientos juntos.

	LECTURA DE CSV	CONSTRUCCIÓN ARRAY	LECTURA Y CONSTRUCCIÓN
Conjunto de datos 1	0,045s	0,002s	0,06s
Conjunto de datos 2	0,1s	0,009s	0,16s
Conjunto de datos 3	0,13s	0,011s	0,2s
Conjunto de datos 4	0,16s	0,014s	0,25s
Conjunto de datos 5	0,22s	0,021s	0,34s
Conjunto de datos 6	0,28s	0,028s	0,44s
Conjunto de datos 7	0,35s	0,037s	0,55s
Conjunto de datos 8	0,45s	0,048s	0,72s
Conjunto de datos 9	0,63s	0,068s	1s
Conjunto de datos 10	0,8s	0,091s	1,3s

Tabla 2: Tiempos de ejecución de las operaciones de la estructura de datos con diferentes conjuntos de datos

A continuación se compara temporalmente la lectura de datos a través de pandas dataframe y con csv reader en listas de Python.

	LECTURA CON PANDAS	LECTURA CON CSV READER
Conjunto de datos 1	0,045s	0.055s
Conjunto de datos 2	0,1s	0.181s
Conjunto de datos 3	0,13s	0.241s
Conjunto de datos 4	0,16s	0.505s
Conjunto de datos 5	0,22s	0.438s
Conjunto de datos 6	0,28s	0.632s
Conjunto de datos 7	0,35s	0.716s
Conjunto de datos 8	0,45s	0.986s
Conjunto de datos 9	0,63s	1.450s
Conjunto de datos 10	0,8s	1.982s

Tabla 3: Tiempos de lectura comparativos entre la lectura del dataframe de panda y la lectura en listas de Python con csv reader.

Las listas de Python son de tipo ArrayList, por lo cual son poco eficientes temporalmente. También se tiene en cuenta que con dataframe se tiene acceso a funciones complementarias de análisis que resultarán útiles en la posterior creación del árbol de decisión

Como medio de comprobación se hace una medición con listas de Python para comprobar experimentalmente la escogencia de la estructura de datos más eficiente.

	CONSTRUCCIÓN ARRAY	CONSTRUCCIÓN LIST
Conjunto de datos 1	0,002s	0.003s
Conjunto de datos 2	0,009s	0.014s
Conjunto de datos 3	0,011s	0,2s
Conjunto de datos 4	0,014s	0.018s
Conjunto de datos 5	0,021s	0.047s
Conjunto de datos 6	0,028s	0.069s
Conjunto de datos 7	0,037s	0.105s
Conjunto de datos 8	0,048s	0.125s
Conjunto de datos 9	0,068s	0.164s
Conjunto de datos 10	0,091s	0.197s

Tabla 4: Comparación de tiempos de ejecución de las operaciones de la estructura de datos entre el array Numpy y las Listas de Python

Como se puede apreciar en la tabla superior las listas de Python son una estructura más pesada que un arreglo, esto se justifica bajo el hecho de que en las listas se necesitan mas bytes para referenciar cada nuevo elemento que se agregue, mientras que el arreglo no.

4.5 Memoria

Se procede a hacer la medición de gasto de memoria con los mismos procedimientos anteriormente medidos.

	LECTURA DE CSV	CONSTRUCCIÓN ARRAY	LECTURA Y CONSTRUCCIÓN
Conjunto de datos 1	4.0664 MiB	2.3007 MiB	8.2656 MiB
Conjunto de datos 2	4.0039 MiB	7.1445 MiB	17.2890 MiB
Conjunto de datos 3	7.0429 MiB	9.0585 MiB	15.1601 MiB
Conjunto de datos 4	4.1289 MiB	11.6757 MiB	16.7656 MiB
Conjunto de datos 5	7.0390 MiB	16.7851 MiB	24.6445 MiB
Conjunto de datos 6	13.4609 MiB	21.4257 MiB	34.9648 MiB
Conjunto de datos 7	16.7460 MiB	28.3164 MiB	42.2968 MiB
Conjunto de datos 8	20.3203 MiB	36.4960 MiB	55.0156 MiB
Conjunto de datos 9	28.1406 MiB	51.7148 MiB	77.8632 MiB
Conjunto de datos 10	35.6484 MiB	67.0234 MiB	101.8164 MiB

Tabla 5: Consumo de memoria de la estructura de datos con diferentes conjuntos de datos

4.6 Análisis de los resultados

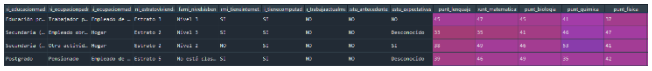
	LECTURA DE CSV	CONSTRUCCIÓN ARRAY	LECTURA Y CONSTRUCCIÓN
Tiempo (segundos)	0,045 - 0,8	0,002 - 0,09	0,06 - 1,3
Memoria (MiB)	4 - 36	2 - 67	8 - 102

Tabla 6: Análisis de los resultados obtenidos con la implementación de la estructura de datos.

5. ÁRBOL CREADO CON EL ALGORITMO C4.5

Se implementa un árbol de decision, en el cual, usando los datos de los resultados de las pruebas ICFES o saber 11 de varios estudiantes se predice con un grado de exactitud medible si un estudiante tendrá éxito o no en el examen saber pro.

Se leen los datos hacienda uso de panda's dataframe, se pulen los campos y llenan los espacios vacíos. Luego se convierten en arreglos de numpy para ser procesados (cuando son de tipo numérico).



Gráfica 5: Visualización en Spyder de un fragment del dataframe creado con uno de los dataset de prueba.

Por ultimo el árbol se forma usando los diccionarios de Python para guardar los datos y ramas del mismo.

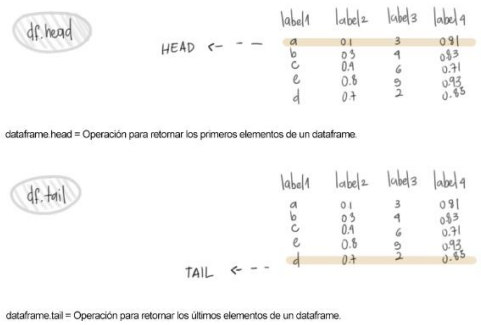
```
--- DataSet 0 - Depth 3 ---
{'punt_ingles <= 50.0': [{'punt_ciencias_sociales <= 49.0': [0,
                                                             {'punt_quimica <= 49.0': [0,
                                                                                       1]}]},
                         1]}
```

Gráfica 6: Diccionario de un árbol de profundidad 3, creado con la estructura de datos creada, usando el dataset0.

5.1 Operaciones de la estructura de datos

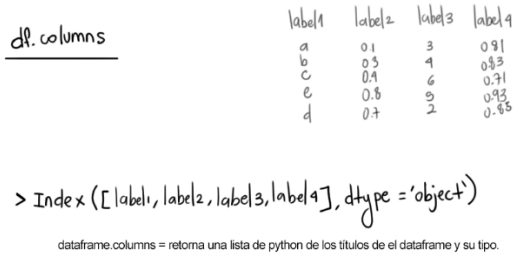
Entre las operaciones más útiles para procesar estas estructuras de datos se encuentran:

Selección de cabeza y de cola, útil para extraer un subconjunto conciso de un dataframe de larga extensión.



Gráfica 7: Selección de cabeza y cola de una estructura de dataframe en python.

Operación de selección de títulos. Llamado .columns, esta operación retorna los nombres de las columnas de un dataframe.



Gráfica 8: Ejemplo de selección de títulos en python.

5.2 Criterios de diseño de la estructura de datos

Para armar esta estructura se usa el algoritmo C4.5 para crear el árbol de decisión, entre el criterio de ganancia de Gini o entropía de Shannon se escoge la entropía, bajo el argumento de que es más apta para análisis explorativos y de tipo *unsupervised learning*, donde la calidad de los datos está mezclada y sus tipos presentan una alta variación, mientras que Gini es más apto para comprar datos de similitud mayor, y que están más organizados.

En este sentido, la entropía, a pesar de presentar una complejidad un poco mayor a Gini por la inclusión de un logaritmo en su fórmula, parece ser una mejor opción dada la cantidad de datos mezclados y "Deconocidos" que se tiene en el dataset.

Se usa la C4.5 sobre otras como la ID3 por su capacidad de manejar datos categóricos, o no numéricos, que no se pueden clasificar con una simple comparación de menor/mayor-qué.

La ventaja de usar *dicts* para guardar el resultado de crear el árbol es que se puede imprimir haciendo uso de *pretty print*, una función de Python que permite visualizar con facilidad los diferentes niveles del árbol, facilitando la lectura del resultado sin necesidad de plug-ins adicionales.

5.3 Análisis de la Complejidad

Al revisar los métodos y estructuras usadas se hallan las siguientes complejidades:

Método	Complejidad
Train_test_split:	O(n)
Check_purity:	O(n)
Create_leaf:	O(n)
Get_potential_splits:	O(nxm)
Calculate_entropy:	O(n)
Calculate_mse:	O(n)
Calculate_overall_metric:	O(n)
Determine_Best_Split:	O(n^2xm)
Split_Data:	O(n)
Determine_Type_Of_Feature:	O(nxm)
Decision_Tree_Algorithm:	O(n^2xm)
Predict_Example:	O(1)
Make_Prediction:	O(n)
Calculate_Accuracy:	O(n)
total	O(n^2xm)
* n es la cantidad de filas del dataset	
* m és la cantida de columnas del arreglo	

Tabla 7: Tabla para reportar la complejidad.

5.4 Tiempos de Ejecución y Uso de Memoria

Se toman medidas de tiempo y memoria en la construcción de árboles con los datasets otorgados por el docente.

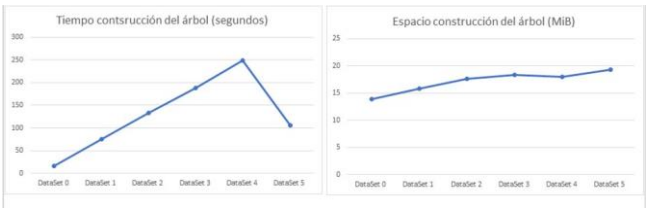
Se puede ver como la construcción de árboles tiene un tiempo máximo de 4 minutos al evaluar 135000 elementos de entrenamiento y 45000 de prueba.

	Tiempo contsrucción del árbol (segundos)	Espacio construcción del árbol (MiB)
DataSet 0	15.803	13.93
DataSet 1	76.005	15.80
DataSet 2	132.70	17.56
DataSet 3	187.61	18.32
DataSet 4	248.19	18.02
DataSet 5	106.13	19.35

Tabla 8: Tiempo y memoria de la construcción de árboles con el algoritmo usado.

5.6 Análisis de los resultados

Se puede ver como el tiempo de ejecución aumenta gradualmente para la cantidad de datos analizada, hay un valor un poco alejado de los otros al final, pero es porque se graficaron siguiendo el orden del nombre de los datasets, siendo el último set (dataset5) mas pequeño que sus predecesores causa una bajada en la curva.



Gráfica 9: Gráfica de Valores de procesamiento.

6. CONCLUSIONES

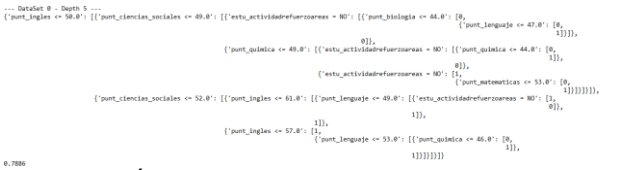
Se comprueba la importancia de crear árboles de decision, ya que las conjeturas e hipótesis que se tendrían de los resultados por parte de los integrantes del equipo se vieron confrontadas con resultados muy diferentes, que dejan en evidencia que el anális minucioso es muy importante para evitar la propagación de errores y estereotipos.

Un resultado muy importante es el descubrimiento de que el puntaje obtenido en inglés es una variable vital a la hora de dterminar el éxito en las pruebas saber Pro.

Otro resultado de vital importancia es el descubrimiento de que algunos datos categóricos al tener muchos items vacíos, que se llenan con "Desconocido" terminan llevando esta respuesta al árbol. Esto pasó con los datos de "Numero de Libros" y "Tiene Etnia", que tuvieron que ser eliminados para que el árbol no quedara con variables de "desconocido".

Gráfica 10: Árbol con todos los datos escogidos en un principio.

Versus el árbol editado, con menos datos categóricos y sin "Desconocido" en sus variables



Gráfica 11: Árbol editado.

Un problema que surgió durante la realización del código es que el algoritmo actual tiene problemas manejando árboles de mayor profundidad con estas cantidades de datos, sería un aspecto a mejorar en el futuro.

De continuar este Proyecto podría optimizarse para la creación de árboles mas grandes y complejos en menos

tiempo, además de crear un método que ayude a catgorizar la infleuncia de cada tipo de dato en el resultado final.

6.1 Trabajos futuros

En un futuro de realizarse un nuevo árbol de decisión ya se contaría con la experiencia de este proyecto y se podrían realizar mejoras estructurales de diseño frente a la primera iteración, además de presentarse la oportunid ad de comparar ambas ejecuciones y sus rendimientos.

Sería interesante aplicar esta misma estructura pero creada bajo el uso de Gini Index en vez de entropía de Shannon y comparar los resultados.

AGRADECIMIENTOS

Se agradece la ayuda de la universidad Eafit y sus recursos de infraestructura y acceso a herramientas digitales.

REFERENCIAS

1. López Takeyas, B (2015) Algoritmo ID3. recuperado en 07/02/2020 de: http://www.itnuevolaredo.edu.mx/takeyas/Apuntes/Inteligencia%20Artificial/Apuntes/IA/ID3.pdf
2. Wikipedia (2019) Árbol de decisión (modelo de clasificación ID3). Recuperado en 07/02/2020 de: https://es.wikipedia.org/wiki/%C3%81rbol_de_decisi%C3%B3n_(modelo_de_clasificaci%C3%B3n_ID3)
3. Espino J, Tijerina J, Cedano M, Amaya E, Pérez J, Chiñas A. (2015) Inteligencia Artificial: Algoritmo C4.5. recuperado en 08/02/2020 de: http://www.itnuevolaredo.edu.mx/takeyas/Apuntes/Inteligencia%20Artificial/Apuntes/tareas_alumnos/C4.5/C4.5(2005-II-B).pdf
4. Brownlee J. (2016) Classification And Regression Trees for Machine Learning. Machine Learning Mastery. Recuperado en 05/02/2020 de: https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/
5. StatSoft Inc (2013) Electronic Statistics Textbook. Tulsa. OK: StatSoft. Recuperado en 05/02/2020 de: http://www.statsoft.com/textbook/CHAID-Analysis
6. B. Bahar, K. Eylem (2015) Applying The CHAID Algorithm to Analyze How Achievement is Influenced by University Students' Demographics, Study Habits, and Technology. Recuperado en 06/02/2020 de: https://www.semanticscholar.org/paper/Applying-The-CHAID-Algorithm-to-Analyze-How-is-by-Baran-Kili%C3%A7/eafaa5f8c88192a4c586e5bb045c5c0e049980ed
7. Heisler S. (2017)). A Beginner's Guide to Optimizing Pandas Code for Speed. UPSIDE. Recuperado en 20/03/2020, de : https://engineering.upside.com/a-beginners-guide-to-optimizing-pandas-code-for-speed-c09ef2c6a4d6
8. Pedregosa S. (2013). Different ways to get memory consumption or lessons learned from ``memory_profiler``. Recuperado en 28/03/2020, de: http://fa.bianp.net/blog/2013/different-ways-to-get-memory-consumption-or-lessons-learned-from-memory_profiler/
9. Mantey S. (2020) Decision Tree Algorithm explained. Recuperado en 03/04/2020 de: https://www.sebastian-mantey.com/theory-blog/decision-tree-algorithm-explained-p1-overview