Predicción del éxito estudiantil con árboles de decisión

Samuel David Villegas Bedoya Universidad Eafit Colombia sdvillegab@eafit.edu.co	Julian Andres Ramirez Jimenez Universidad Eafit Colombia jaramirezj@eafit.edu.co	Miguel Correa Universidad Eafit Colombia macorream@eafit.edu.co	Mauricio Toro Universidad Eafit Colombia mtorobe@eafit.edu.co

RESUMEN

Este trabajo se centra en el diseño y desarrollo de un algoritmo basado en árboles de decisión que nos permita predecir el éxito académico de los estudiantes en la educación superior, más específicamente en las pruebas Saber Pro. Esto se hará teniendo en cuenta ciertos factores socioeconómicos y los resultados de las pruebas ICFES Saber 11. Es importante dar solución a este problema debido a que podemos identificar las variables que más influyen en que un estudiante pueda lograr un desempeño por encima de la media o no en las pruebas Saber Pro, y a futuro estos datos podrían ayudar a tomar decisiones gubernamentales que impactan sobre la calidad de la educación.

El algoritmo propuesto para abordar el problema es el CART, con este se construyó el árbol de decisión y luego se validó, con lo que obtuvimos una exactitud de 0.71, precisión de 0.46 y sensibilidad de 0.65 en las predicciones, además se evidenció que las variables que más importancia tienen al momento de que el árbol realice la predicción son el puntaje en matemáticas e inglés de las pruebas ICFES saber 11, para ciertos rangos de puntajes.

En cuanto a tiempos de ejecución se evidenció que el algoritmo gasta en promedio para el conjunto de datos más grande (135000 datos) alrededor de 40 segundos y el consumo de memoria para este mismo conjunto del árbol de decisión es de 667.75 MB.

Palabras clave

Árboles de decisión, aprendizaje automático, éxito académico, predicción de los resultados de los exámenes

1. INTRODUCCIÓN

Año tras año muchas de las universidades de todo el mundo buscan la excelencia académica en sus estudiantes, cada una lo hace a su manera, pero en cada país existen formas de ver si una universidad tiene cierto nivel académico en sus estudiantes, esto es por medio de pruebas estandarizadas, en nuestro país, por ejemplo, tenemos a las pruebas saber Pro. Estas pruebas las presentan la mayoría de los estudiantes, e incluso es requisito de graduación en la mayoría de las universidades. Por lo anterior un alto porcentaje de estudiantes de pregrado lo hacen, dando pie a pensar que puede ser un gran indicador de la calidad académica, de allí el interés de las instituciones de querer sacar un buen puntaje en estas pruebas.

Una de las maneras de buscar el mejoramiento es prediciendo los resultados, es decir, adelantarse a la prueba real, dado que podemos poner en evidencia los factores y variables que influyen en un resultado positivo o negativo, y a partir de esto buscar soluciones que no solamente aportan al resultado de la prueba, sino también al mejoramiento de la educación. Teniendo en cuenta lo anteriormente mencionado, el objetivo de nuestro proyecto es predecir los resultados y buscar aquellos factores que pueden influir en los resultados de las pruebas estandarizadas en las universidades de América latina, para así mejorar la calidad de educación de esta región. Esto lo pretendemos lograr mediante un algoritmo basado en árboles de decisión.

1.1. Problema

El problema por solucionar es realizar una predicción del éxito académico de los estudiantes en la educación superior en base a diferentes datos de estos, resolver este problema ayudaría a que se puedan tomar decisiones a tiempo por parte de las Instituciones de educación superior para el mejoramiento y así evitar malos resultados.

1.2 Solución

En este trabajo, nos centramos en los árboles de decisión porque proporcionan una gran explicabilidad, puesto que, son simples de entender y de interpretar, si el árbol no es muy grande se puede visualizar, se puede trabajar tanto con variables cuantitativas como cualitativas, se utiliza el modelo de caja blanca el cual le permite al ingeniero tener más control de los elementos matemáticos y lógicos del algoritmo. [9]

Evitamos los métodos de caja negra como las redes neuronales, las máquinas de soporte vectorial y los bosques aleatorios porque carecen de explicabilidad, debido a que puede ser complejo establecer un camino lógico que siguió el algoritmo para dar una respuesta específica. [10]

En este caso se eligió el algoritmo CART, este lo elegimos, dado que cumplía con lo que necesitábamos y además es el más sencillo de entender e implementar comparado con los demás algoritmos

1.3 Estructura del artículo

En lo que sigue, en la sección 2, presentamos el trabajo relacionado con el problema. Más adelante, en la sección 3, presentamos los conjuntos de datos y métodos utilizados en

esta investigación. En la sección 4, presentamos el diseño del algoritmo. Después, en la sección 5, presentamos los resultados. Finalmente, en la sección 6, discutimos los resultados y proponemos algunas direcciones de trabajo futuras.

2. TRABAJOS RELACIONADOS

2.1 Árboles de decisión para predecir factores asociados al desempeño académico de estudiantes de bachillerato en las pruebas Saber 11°

En este artículo se habla de que en base a un modelo de clasificación basado en árboles de decisión se detectaron factores asociados al desempeño académico de los estudiantes de grado undécimo de la educación media, en este caso se aplica el modelo para los estudiantes que presentaron las pruebas Saber 11 en los años 2015 y 2016. Para el modelo de clasificación utilizaron el algoritmo J48,el cual se basa e implementa el C4.5.,además la precisión que lograron con el algoritmo fue del 67%.[1]

2.2 Using Decision Trees to Understand Student Data

En este artículo se habla de la aplicación que se le dio a un algoritmo basado en árboles de decisión con el que a partir de datos de estudiantes universitarios se generaron gráficas para la comprensión humana, cuyo análisis permitía predecir la graduación, o mejor, comprender los factores que conducían a la misma y así poder generar conclusiones que aporten a futuras decisiones institucionales. Además se comparó este método de árboles de decisión con redes neuronales, máquinas vectoriales de soporte y regresión del núcleo, para comprobar su efectividad frente a estas. La precisión de este modelo estuvo alrededor del 64%.[2]

2.3 Predicción del rendimiento de estudiantes de ingeniería mediante árboles de decisión

En este artículo se plantearon generar un modelo predictivo utilizando árboles de decisión que buscaba pronosticar los desempeños de estudiantes de ingeniería, permitiendo identificar antes, a los estudiantes que posiblemente tuvieran más riesgo de fracasar, dando la posibilidad de hacer un acompañamiento a estos casos. El algoritmo utilizado fue el J48, el cual es una implementación en el lenguaje de programación Java del algoritmo C4.5, según los datos obtenidos la tasa positiva del modelo para la clase FAIL fue de 0.907, significando que el modelo estaba identificando con éxito aquellos estudiantes que probablemente fracasarán. [3]

2.4 Modelo predictivo de deserción estudiantil basado en árboles de decisión

En este artículo contrario a los otros trabajos relacionados, se buscaba encontrar aquellos factores que influyen en la deserción estudiantil, implementando varios algoritmos en base a árboles de decisión como lo fue el CART y este apoyándose en el algoritmo Hunt. En cuanto a la efectividad del algoritmo realizaron la curva Receiver Operating Characteristic (ROC) obteniendo como resultado un 94% de efectividad. [4]

3. MATERIALES Y MÉTODOS

En esta sección se explica cómo se recopilaron y procesaron los datos y, después, cómo se consideraron diferentes alternativas de solución para elegir un algoritmo de árbol de decisión.

3.1 Recopilación y procesamiento de datos

Obtuvimos datos del *Instituto Colombiano de Fomento de la Educación Superior* (ICFES), que están disponibles en línea en ftp.icfes.gov.co. Estos datos incluyen resultados anonimizados de Saber 11 y Saber Pro. Se obtuvieron los resultados de Saber 11 de todos los graduados de escuelas secundarias colombianas, de 2008 a 2014, y los resultados de Saber Pro de todos los graduados de pregrados colombianos, de 2012 a 2018. Hubo 864.000 registros para Saber 11 y 430.000 para Saber Pro. Tanto Saber 11 como Saber Pro, incluyeron, no sólo las puntuaciones sino también datos socioeconómicos de los estudiantes, recogidos por el ICFES, antes de la prueba.

En el siguiente paso, ambos conjuntos de datos se fusionaron usando el identificador único asignado a cada estudiante. Por lo tanto, se creó un nuevo conjunto de datos que incluía a los estudiantes que hicieron ambos exámenes estandarizados. El tamaño de este nuevo conjunto de datos es de 212.010 estudiantes. Después, la variable predictora binaria se definió de la siguiente manera: ¿El puntaje del estudiante en el Saber Pro es mayor que el promedio nacional del período en que presentó el examen?

Se descubrió que los conjuntos de datos no estaban equilibrados. Había 95.741 estudiantes por encima de la media y 101.332 por debajo de la media. Realizamos un submuestreo para equilibrar el conjunto de datos en una proporción de 50%-50%. Después del submuestreo, el conjunto final de datos tenía 191.412 estudiantes.

Por último, para analizar la eficiencia y las tasas de aprendizaje de nuestra implementación, creamos al azar subconjuntos del conjunto de datos principal, como se muestra en la Tabla 1. Cada conjunto de datos se dividió en un 70% para entrenamiento y un 30% para validación. Los conjuntos de datos están disponibles en

https://github.com/mauriciotoro/ST0245-Eafit/tree/master/proyecto/datasets .

	Conjunto de datos 1	Conjunto de datos 2	Conjunto de datos 3	Conjunto de datos 4	Conjunto de datos 5
Entrenamiento	15,000	45,000	75,000	105,000	135,000
Validación	5,000	15,000	25,000	35,000	45,000

Tabla 1. Número de estudiantes en cada conjunto de datos utilizados para el entrenamiento y la validación.

3.2 Alternativas de algoritmos de árbol de decisión

3.2.1 ID3

El algoritmo ID3 realizado por Quinlan implementa un sistema de árboles de decisión con dirección de arriba abajo, teniendo en cuenta lo siguiente:

Ejemplos: Se denomina ejemplos a aquellos datos que se utilizan para el entrenamiento del algoritmo, buscan aumentar la posibilidad de acierto de este, es decir entre más ejemplos mejor. Cada uno de estos se componen de atributos y valores.

Atributos: Se denomina atributo a aquellas características que componen a los datos, estos son los que se toman en cuenta a la hora de crear el árbol. Por ejemplo, si queremos predecir la posibilidad de tener un contagio de COVID un atributo podría ser Tener tos o fiebre.

Valores: Denominamos valores a los estados que pueden tomar los atributos, pueden ser cuantitativos y cualitativos. Por ejemplo, si queremos predecir la posibilidad de almorzar en un restaurante, un atributo puede ser si hay muchas personas y sus posibles valores pueden ser ninguna, algunas o muchas.

Hojas: Es aquel resultado que se tiene por cada una de las ramas. En algunos casos puede pasar que en varias ramas repitan el mismo resultado. Puede ser un resultado binario, es decir, si o no, verdadero o falso, etc. Aunque también se pueden usar más de dos estados.

Arcos: Los cuales contienen valores posibles del nodo inicial.

Entropía: Este concepto viene de la teoría de la información, que consiste básicamente en el nivel de desorden de los componentes de un sistema, en nuestro caso el nivel de variaciones que podamos tener a medida que agregamos nuestros atributos.

- Una muestra completamente homogénea tiene entropía 0.
- Una muestra igualmente distribuida tiene entropía 1.

Fórmulas

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Figura 1: Es la fórmula general para la entropía.

$$Entropia(s) = -Plog_2(P) - Nlog_2(N)$$

Figura 2: Es una variante de la fórmula, se utiliza más que todo para calcular la entropía original, donde P= #EjemplosPos/TotalEjemplos y N= #EjemplosNeg/TotalEjemplos.

Ganancia (Information Gain): El nivel de información que se gana en cada atributo. De este concepto depende el nivel de cada uno de los atributos, es decir, entre más ganancia tenga un atributo es ubicado en un nivel más alto.

Proceso

¿Cómo buscar el nodo principal?

- Se calcula la entropía del total
- Se divide el conjunto de datos en función de los diferentes atributos.
- Se calcula la entropía en cada rama y suma proporcionalmente las ramas para calcular la entropía del total.
- Se resta este resultado de la entropía original
- El resultado es la ganancia de información (Descenso de entropía).
- El atributo con mayor ganancia es seleccionado como nodo de decisión.

Otros pasos:

- Una rama con entropía 0 se convierte en hoja (Todos los casos clasificados).
- Si no es así, la rama debe seguir subdividiendo
- El algoritmo se ejecuta recursivamente hasta que se llegue a nodos hoja. [5]

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Figura 3: Datasets ejemplo golf.

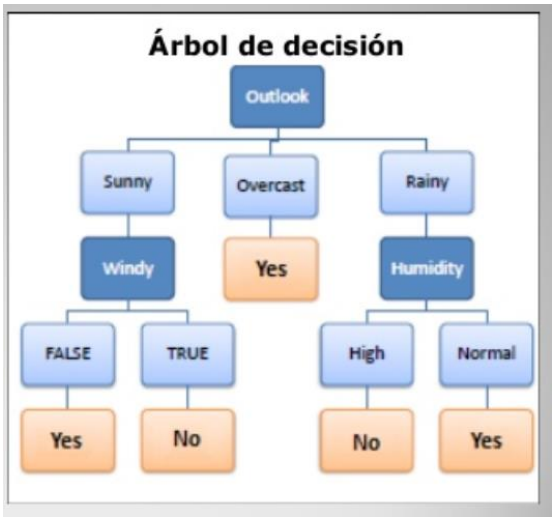


Figura 4: Árbol generado de los datasets mostrados en la figura 1 utilizando ID3.

Complejidad Espacio: Lineal/ Polinómico

Complejidad Tiempo: Polinómica/Exponencial [6]

3.2.2 C4.5

El algoritmo C4.5 podríamos llamarlo como una actualización del algoritmo anterior llamado ID3, por ende, se basa en los mismos conceptos como lo son: La entropía y la ganancia de información. Pero introduce nuevos conceptos como:

Split Info (SI): Permite generar nodos hijos con la menor impureza posible.

$$SI(S) = -\sum_{i=1}^{N} \frac{S_i}{S} \left(log\left(\frac{S_i}{S}\right)\right)$$

Figura 5

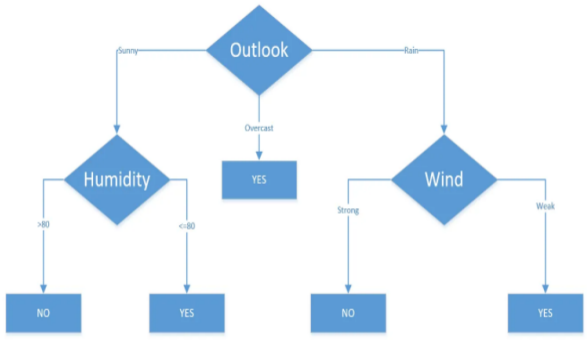
Ratio (GR): Es la razón entre el IG que teníamos por ejemplo en el ID3 y el nuevo concepto SI.

$$GR(S,A) = \frac{IG(S,A)}{SI(S)}$$

Figura 6

Eliminamos el concepto de ganancia como ya lo conocíamos dado que nos generaba un problema, Al momento de evaluar los atributos el algoritmo ID3 favorecía a los atributos con gran cantidad de valores. Para arreglar este asunto Quinlan establece la propuesta del Ratio (GR).

También se introduce un nuevo concepto como lo es **La poda**, esta tiene como función eliminar y reemplazar por nodos de hoja a aquellas ramas que tengan una tasa de error más alta establecido por una regla de decisión.



Árbol de decisión generado por C4.5

Figura 7: Es un ejemplo del árbol generado por C4.5 en comparación con el árbol hecho en el algoritmo ID3.

En cuanto a la complejidad no se encontraron registros de cambio en memoria o tiempo, pero si mejoraron luego con

C5.0/See5 que es una versión mejorada del algoritmo en cuestión.

3.2.3 CART

CART es el acrónimo de Classification and Regression Trees(Árboles de clasificación y regresión). El algoritmo CART es un método que utiliza datos históricos para construir árboles de clasificación o de regresión los cuales son usados para clasificar o predecir nuevos datos.[7]

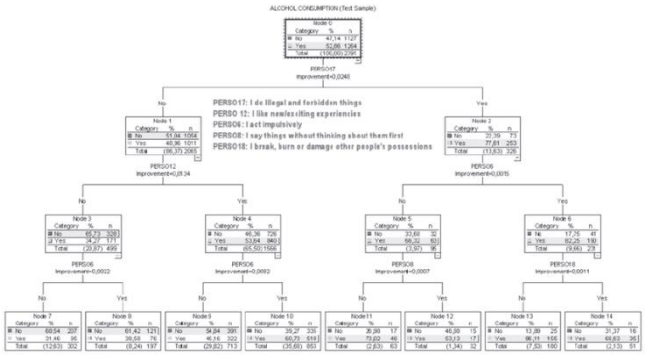


Figura 8: Árbol de clasificación generado por el algoritmo CART (Breiman et al., 1984)

3.2.4 QUEST

QUEST es el acrónimo de Quick, Unbiased, Efficient Statistical Tree (Árbol Estadístico Eficiente, Insesgado y Rápido). El procedimiento que sigue este algoritmo básicamente es el siguiente: "primero se elige la mejor variable predictora cuyo objetivo es que el número de categorías que poseen las variables no afecte a la elección de la mejor variable, para realizar después la mejor segmentación de la variable que ha seleccionado" [8]

(Para este algoritmo en específico no encontramos una figura)

4. DISEÑO DE LOS ALGORITMOS

En lo que sigue, explicamos la estructura de los datos y los algoritmos utilizados en este trabajo. La implementación del algoritmo y la estructura de datos se encuentra disponible en Github[1].

4.1 Estructura de los datos

La estructura de datos utilizada es un árbol de decisión binario, este consiste en una serie de condiciones que dividen una entrada hasta el punto de dar un resultado. La entrada va tomando diferentes caminos hasta llegar a la meta, los caminos en cada decisión sólo pueden ser dos, de allí su nombre binario, el nombre de sus partes puede variar, pero aquí tenemos un ejemplo:

Raíz (Nodo principal): Es el nodo de más alto nivel, se busca que sea la condición que tenga más relevancia entre el conjunto de datos.

Ramificaciones: Son los posibles valores que puede tomar una condición, en este caso sí y no.

Hojas: Son las condiciones que pueden ser derivadas de la raíz o de otras hojas, se caracterizan porque también tienen subdivisiones como el nodo raíz.

Sub Hoja: Son condiciones producto de las divisiones de las hojas.

Nodo terminal o resultado: Es el valor resultante de las divisiones anteriores.

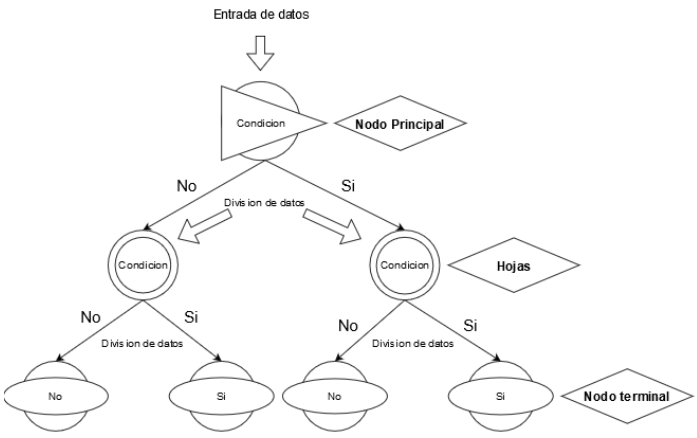


Figura 9: Estructura básica de árbol de decisión binario.

[1] https://github.com/sdvillegab/ST0245-001/tree/master/proyecto

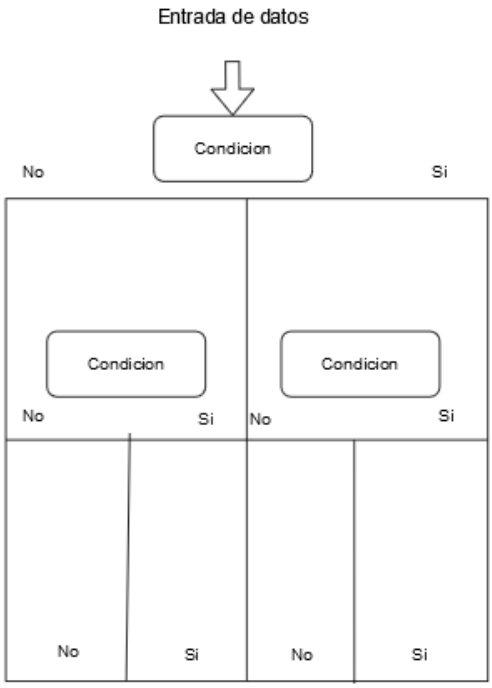


Figura 10: Alternativa de visualización para un árbol de decisión.

4.2 Algoritmos

4.2.1 Entrenamiento del modelo

El algoritmo utilizado es el CART este consiste en realizar un árbol de decisión con los siguientes pasos.

- Separar las reglas o condiciones que se utilizan para la realización del árbol.
- Hallar el índice de Gini ponderado de las condiciones existentes.
- Encontrar el índice de Gini ponderado menor, correspondiente a la condición que mejor divide los datos.
- Separar los datos por los datos que cumplen con la condición principal y los que no.
- Con cada conjunto de datos separado se realiza nuevamente una búsqueda del índice de Gini ponderado menor entre las condiciones faltantes, este proceso se repite hasta que en la columna resultante estén todos los datos de un mismo valor.

4.2.2 Algoritmo de prueba

En el algoritmo de prueba se realizó el siguiente proceso:

- Se realizó la lectura de la data.
- Se ingresó al modelo cada uno de los datos, guardando el resultado del modelo.
- Se compara cada uno de los resultados del modelo con los resultados reales y se van sumando cada acierto y desacierto para realizar al final el porcentaje de acierto del modelo, y así saber qué tan preciso resultó el modelo diseñado.

Usa terminal	Juega videojuegos	Tiene mas disco	Gusto
Si	Si	Si	Si
Si	Si	No	No
Si	No	No	Si
No	No	No	No
Si	No	Si	Si
No	No	Si	No
No	Si	Si	No
Si	Si	No	No
No	No	Si	No
No	No	No	No
Si	Si	No	No
Si	Si	Si	Si
No	Si	Si	No
Si	No	Si	Si
Si	Si	No	No
Si	Si	Si	Si
No	Si	No	No

Figura 11: Data para la realización de ejemplo sobre si una persona usará el SO Linux o no, por medio del algoritmo CART.

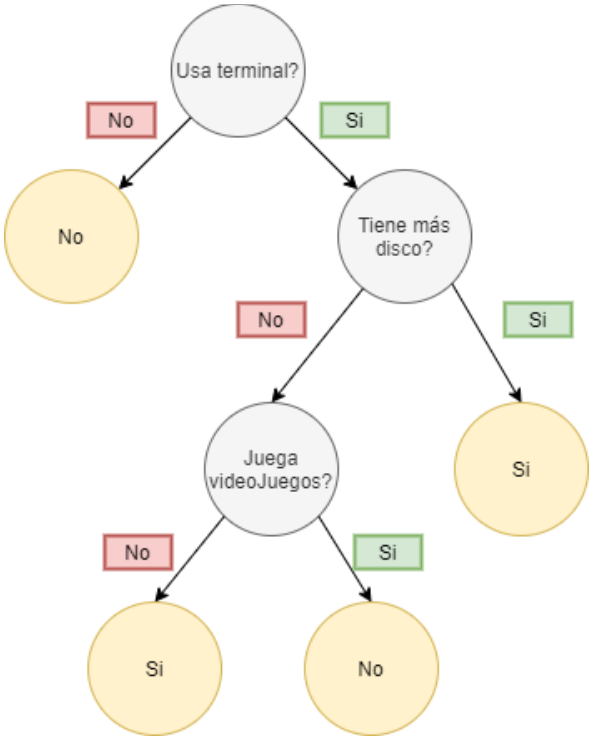


Figura 12: Árbol resultante del estudio de la figura 11 con la utilización del algoritmo CART.

4.3 Análisis de la complejidad de los algoritmos

El análisis en notación O lo hicimos en base a varias ideas:

1. Al crear el árbol binario estamos gastando un tiempo Log(M), puesto que multiplicamos el problema se divide cada vez que hacemos la recursión, se debe tener en cuenta que este tiempo se puede ver reducido debido a que no se realiza todos los casos y que los datos se van reduciendo en cada recursión, es decir, los datos pueden encontrar su caso de parada rápido e irse en algunos casos solo por un camino, por la naturaleza del algoritmo CART.
2. Se realizó el algoritmo utilizando la famosa frase "divide y vencerás" en este caso se intentó dividir los métodos por funcionalidad teniendo un método principal para la creación del árbol, este método accionaba los demás, cada vez que se realizaba el método los otros métodos también se hacían, también se tomaba en cuenta los ciclos es decir la complejidad de estos métodos debía ser multiplicada por la complejidad del principal.
3. En el caso de la validación tenemos una cantidad M de datos para validar e ingresamos estos datos al árbol, este nos arroja un resultado en O(H) puesto solo elige un camino cada recursión, y en el peor de los casos bajará hasta la altura del mismo árbol.

Algoritmo	**La complejidad del tiempo**
Entrenar el árbol de decisión	O(V*N*M*Log(M))
Validar el árbol de decisión	O(M*H)

Tabla 2: Complejidad temporal de los algoritmos de entrenamiento y prueba. siendo M=> la cantidad de datos, filas o estudiantes, N=> la cantidad de variables o columnas, V =>la cantidad de valores (variables! =valores), H=> la altura del árbol.

Algoritmo	**Complejidad en memoria**
Entrenar el árbol de decisión	O(V*M*N+A)
Validar el árbol de decisión	O(M*N+A)

Tabla 3: Complejidad de memoria de los algoritmos de entrenamiento y prueba. Donde *M*=>La cantidad de datos, filas o estudiantes, N=>La cantidad de variables o columnas, A=> el número de nodos del árbol.

4.4 Criterios de diseño del algoritmo

El primer problema fue saber que necesitábamos el funcionamiento de un algoritmo de clasificación, la elección de este se hizo en base a la facilidad de implementación y desarrollo, que tuviera como ítem fundamental el índice de gini y cumpliera con los tipos de clases previstos, este fue el algoritmo CART ya que cumplía con todos los items necesarios.

El segundo problema fue saber dónde guardaremos la data, necesitábamos una estructura de datos lineal, dinámica, que nos permitiera hacer recorridos fácilmente, en tiempos cortos, la elección fue un ArrayList<String []> este cumplia con los parámetros necesarios, hay que tener en cuenta que se necesitaba guardar filas y columnas, en las columnas ya se tenía un tamaño fijo puesto que anteriormente se habían elegido las variables y valores necesarios, por eso la utilización de String[] en cambio las filas no se podían determinar, dependen de la cantidad de estudiantes, por esa razón se utilizó ArrayList<>.

El tercer problema era cómo saber si un valor ya fue validado es decir si ya es un nodo, para esto necesitábamos una estructura que nos almacenará los valores ya validados y también que nos permitiera buscar, añadir, y que no ingresara elementos repetidos en un tiempo constante, la elección fue HashSet() esta estructura nos permite realizar lo anterior muy fácilmente.

Con base en el problema anterior también necesitábamos guardar los valores de las variables mientras recorremos la data, estas no se podían repetir, para ello se utilizó la estructura TreeSet() que cumplía con lo anterior. Nota: También se podía realizar con HashSet().

La elaboración del árbol se tenía que hacerla manual, es decir, realizar clase la Árbol y la clase Nodo, necesitábamos que los nodos nos guardarán un valor que nos permitiera luego realizar los tests, este valor es de tipo String, esta variable tiene la estructura variable- valor puesto que si ponemos solo valor sería ambiguo debido a que dos variables pueden tener un mismo valor, por ejemplo: Si, No, " ".

Como se mencionó anteriormente el algoritmo se basa en estructuras y con algunas más añadidas queda así:

1. ArrayList<String[]> (Data)
2. Variable [] (Variables)
3. Árbol y Nodo (Clases creadas)
4. TreeSet<String> (Valores)

   Nota: Cada variable tenía un TreeSet con sus respectivos valores

5. HashSet<String> (Valores Validados)

El proceso que se realiza en el algoritmo juntos a las estructuras es el siguiente:

1. Se cargan los datos en el ArrayList

2. Se separan las variables a utilizar en un vector de tipo Variable, en este caso serán 30.
3. Se buscan entre los datos los posibles valores de las variables anteriores.
4. Se realiza el conteo, este proceso es el que se encarga de contar de cada valor la cantidad de 0 y 1 de quienes cumplen con el valor, y la cantidad de 0 y 1 de quienes no cumplen el valor.
5. Se realizan los cálculos del gini de cada valor.
6. Se elige el valor con el gini más bajo.
7. Se pone el valor como nodo principal y se añade en el HashSet<String>.
8. Se divide la data para quienes cumplen el valor y quienes no. Si alguna parte de la data () cumple las siguientes condiciones se llama al método crear Árbol, las condiciones son:
    a. No todos los resultados son los mismos es decir la última columna tiene valores diferentes.
    b. No todos los valores entre los datos están en el HashSet<String> es decir hay valores por validar aún.
    c. La cantidad de datos es igual o mayor que 10.

Si se llega a incumplir alguno de estas condiciones el nodo será terminal, es decir, será una hoja y tendrá el resultado del valor que tenga más datos relacionados.

9. El proceso se realiza hasta que se encuentren todos los nodos hoja.

5. RESULTADOS

5.1.1 Evaluación del modelo en entrenamiento

A continuación, presentamos las métricas de evaluación de los conjuntos de datos de entrenamiento en la Tabla 3.

	Conjunto de datos 1	***Conjunto de datos 2***	***Conjunto de datos 3***	***Conjunto de datos 4***	***Conjunto de datos 5***
Exactitud	0.86	0.86	0.86	0.86	0.91
Precisión	0.89	0.88	0.88	0.88	0.93
Sensibilidad	0.83	0.83	0.83	0.83	0.89

Tabla 3. Evaluación del modelo con los conjuntos de datos de entrenamiento.

5.1.2 Evaluación de los conjuntos de datos de validación

A continuación, presentamos las métricas de evaluación para los conjuntos de datos de validación en la Tabla 4.

	Conjunto de datos 1	***Conjunto de datos 2***	***Conjunto de datos 3***	***Conjunto de datos 4***	***Conjunto de datos 5***
Exactitud	0.86	0.86	0.86	0.86	0.71
Precisión	0.88	0.88	0.88	0.88	0.73
Sensibilidad	0.84	0.83	0.83	0.83	0.65

Tabla 4. Evaluación del modelo con los conjuntos de datos de validación.

5.2 Tiempos de ejecución

	Conjunto de datos 0	***Conjunto de datos 1***	***Conjunto de datos 2***	***Conjunto de datos 3***	***Conjunto de datos 4***
Tiempo de entrenamiento	3.5 s	11.4 s	20.2 s	29.7s	38.7s
Tiempo de validación	0.032 s	0.095 s	0.164 s	0.239 s	0.314 s

Tabla 5: Tiempo de ejecución del algoritmo *CART para cada conjunto de datos.*

5.3 Consumo de memoria

Presentamos el consumo de memoria del algoritmo y del árbol de decisión binario, para diferentes conjuntos de datos, en la Tabla 6.

	Conjunto de datos 0	***Conjunto de datos 1***	***Conjunto de datos 2***	***Conjunto de datos 3***	***Conjunto de datos 4***
Consumo de memoria	73.86 MB	139.12 MB	220.01 MB	259.9 MB	350.9 MB

Tabla 6: Consumo de memoria del árbol de decisión binario y del algoritmo para diferentes conjuntos de datos.

6. DISCUSIÓN DE LOS RESULTADOS

Para este proyecto se descartaron diferentes variables que poseía el conjunto de datos debido a que podían generar sesgos en el modelo y seria antiético incluirlas, además es

importante mencionar que estamos trabajando con seres humanos, y esto implica que las variables externas pueden alterar el comportamiento de los individuos, por lo tanto, las predicciones que haga el modelo pueden fallar debido a lo anterior. La exactitud que se obtuvo fue de 0.71, precisión 0.46, sensibilidad 0.65, por lo que se mencionó anteriormente no es necesario que el modelo sea perfecto, dado que los humanos son tan variables, por lo que un porcentaje de aciertos de al menos 60% en las predicciones ya se puede considerar apropiado, en nuestro caso fue del 71%.

En cuanto al consumo de memoria y tiempo, es notorio que tener un exponencial en tiempo no es lo más optimo, pero en este caso es lo que nos propone el algoritmo CART, y en cuanto al consumo de memoria es el óptimo, dado que cargar los datos consume cierta memoria y luego construir el árbol consume incluso menos que la carga de los datos.

En este caso el modelo tiene más aplicabilidad para la ayuda a estudiantes con baja probabilidad de éxito, puesto que dada la variabilidad que se puede presentar en un individuo por el hecho de ser humano, es mucho más viable ayudar a los que estarán por debajo del promedio para que mejoren a futuro que para dar una beca. Además, es importante tener en cuenta que para dar una beca se deben analizar otros aspectos adicionales de una forma más detallada como lo son el estrato, ingresos económicos y otras variables relacionadas, más que solo el aspecto académico.

6.1 Trabajos futuros

El algoritmo realizado cumple con lo mínimo pedido para este proyecto, pensamos que se podría mejorar en la medida en que se tenga más datos, y se busque una implementación que permita generar más exactitud por ejemplo usando bosques aleatorios.

AGRADECIMIENTOS

Agradecemos a nuestras familias por todo el apoyo emocional que nos brindaron durante el desarrollo del proyecto.

REFERENCIAS

1. Timarán,R y Caicedo,J y Hidalgo,A . Árboles de decisión para predecir factores asociados al desempeño académico de estudiantes de bachillerato en las pruebas Saber 11°. *Revista De Investigación, Desarrollo E Innovación* 9, 2 (2019). DOI:http://dx.doi.org/10.19053/20278306.v9.n2.2019.9184

2.Murray, E. Using Decision Trees to Understand Student Data. 2015.

3. R. R. Kaba y R. S. Bichkar. Performance Prediction of Engineering Students using Decision Trees, International Journal of Computer Applications (0975 – 8887), *36*(11). 8-12.

4. Cuji, B., Gavilanes, W. y Sanchez, R. Modelo predictivo de deserción estudiantil basado en arboles de decisión. Revista Espacios, *38*(55). 17- 26.

5. Sancho, F. (Enero 2013). Arboles decisión id3 Recuperado 16 Agosto, 2020, de https://es.slideshare.net/FernandoCaparrini/arboles-decision-id3

6. Iglesia, J.L. Descubriendo la Inteligencia Artificial. (2017, 11, 3). Nº 177: IA Probabilidad - Árboles de Decisión ID3 02 [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=PxPKZA87EXU

7. Díaz Sepúlveda, J. and Correa, J., . Comparación entre árboles de regresión CART y regresión lineal. *Comunicaciones en Estadística*, 2013, 6(2), p.175.

8.Parra,F. Estadística y Machine Learning con R. 2019.

9 y 10. Parada Torralba, P. Toda la verdad, y nada más que la verdad, sobre los algoritmos caja negra de Inteligencia Artificial. Recuperado de http://www.pascualparada.com/algoritmoscajanegra/