DISEÑO DE UN ALGORITMO PARA PREDECIR EL ÉXITO EN LAS PRUEBAS SABER PRO

Sebastian Castaño Orozco	Dennis Castrillón Sepúlveda
Universidad Eafit	Universidad Eafit
Colombia	Colombia
scasta31@eafit.edu.co	dcastri9@eafit.edu.co

Miguel Correa	Mauricio Toro
Universidad Eafit	Universidad Eafit
Colombia	Colombia
mcorrea@eafit.edu.co	mtorobe@eafit.edu.co

RESUMEN

El siguiente trabajo tiene como fin la construcción de un algoritmo basado en árboles de decisiones, que permita predecir el éxito de un estudiante de pregrado en las pruebas Saber Pro, teniendo como definición de éxito, la obtención de una nota superior al promedio de su cohorte.

El trabajo se enmarca en la necesidad de asignar una importancia cuantitativa respecto a los aspectos para tener en cuenta en la preparación en torno a la prueba y la necesidad de brindar una herramienta de análisis en torno al proceso global de preparación, presentación y obtención de resultados de la prueba Saber Pro.

El algoritmo propuesto entonces entrena bajo un conjunto de datos n, luego valida generando el árbol con otro conjunto de datos de tamaño n y calcula la exactitud, evaluando para cuantos estudiantes se predijo correctamente el resultado de sus pruebas. Este algoritmo se realizó bajo árboles de decisión CART. Los tiempos de ejecución del algoritmo son altos debido a la complejidad de las operaciones internas, sin embargo, se consiguen exactitudes cercanas al 80% para conjuntos de datos de 45000 estudiantes lo que representa una buena cifra para la primera versión del algoritmo y la utilización de árboles de decisión.

1. INTRODUCCIÓN

Un reto continuo en la metodología de enseñanza y aprendizaje en los pregrados de las distintas universidades es ofrecer modelos de educación que permitan la obtención de resultados positivos por parte de los estudiantes en las pruebas realizadas por el estado en los últimos semestres de la carrera.

Por tanto, predecir qué factores pueden influir en mayor proporción en la probabilidad de tener éxito en los resultados de las pruebas Saber Pro (en el caso de Colombia) puede ser un mecanismo óptimo de toma de decisiones relacionadas con el sistema educativo.

1.1. Problema

Se requiere predecir el éxito de un estudiante en las pruebas Saber Pro de acuerdo con sus características sociodemográficas y académicas, a saber, edad, ingresos de sus padres, pregrado, resultados en la prueba Saber 11, género, estrato, entre otras, a través de un algoritmo de árboles de decisión.

La resolución de este problema significaría un avance que impactaría directamente en la toma de decisiones relativas al proceso social y educativo por el que atraviesa un estudiante previo a enfrentarse a las pruebas Saber Pro en aras de mejorar los resultados y la probabilidad de éxito en el futuro.

1.2 Solución

En este trabajo, nos centramos en los árboles de decisión porque proporcionan una gran explicabilidad [12]. Los métodos de caja negra como las redes neuronales, las máquinas de soporte vectorial y los bosques aleatorios porque carecen de explicabilidad [13].

Para este trabajo se decide desarrollar el árbol de decisión a través del algoritmo C4.5 debido a que es un algoritmo que puede trabajar con bases de datos que poseen datos perdidos o errados, crea un árbol más generalizado y no cae en sobre ajustes, puede trabajar con variables continuas y nominales.

Además, es un algoritmo que surgió como mejora al algoritmo ID3, por tanto, corrige las falencias de este último y crea ciertas ventajas. De igual manera, es un algoritmo que posee gran cantidad de información en la web y su funcionamiento se puede entender de manera relativamente fácil.

A nivel general, el algoritmo C4.5 se encarga de calcular una ganancia y un radio de ganancia para cada variable de cada atributo. Para los valores continuos, calcula ganancias de acuerdo con ciertos rangos y convierte este tipo de variable en nominal creando rangos para menores o iguales a cierto valor y mayores a este mismo. Posteriormente, selecciona el atributo con el mayor radio de ganancia y crea el árbol de decisión.

2. TRABAJOS RELACIONADOS

2.1 Predicción del desempeño de un estudiante a través de Machine Learning

Este trabajo, realizado como parte de la tesis doctoral del estudiante Murat Pojon de la universidad de Tampere en Finlandia consistió en predecir los resultados en un examen para una base de datos de 480 estudiantes pertenecientes a la universidad de Jordania, teniendo como referencia 17 características de cada estudiante, entre las que se incluía género, nacionalidad y veces que el estudiante alzó la mano durante clase, entre otras.

La predicción del desempeño por árboles de decisión se realizó con el algoritmo de particionamiento recursivo (CART) y obtuvo una precisión del 91.7%. Es pertinente resaltar que Pojon realiza una comparación entre varios algoritmos, para los cuales el algoritmo CART obtiene el segundo valor más cercano a los resultados reales, por debajo del algoritmo de clasificación de Naïve Bayes.

2.2 Predicción de resultados mediante minería de datos y métodos de clasificación

Dorina Kababchieva de la universidad de Sofía en Bulgaria, procesa los datos de 10330 estudiantes descritos por 20 parámetros entre los que se encuentra género, puntaje en el examen de admisión y semestre, entre otros, con el fin de predecir los resultados en un examen de acuerdo con estos parámetros. La predicción se realiza a través del algoritmo C4.5 y se obtiene una precisión del 63.1%. [2]

Cabe destacar que la precisión alcanzada presenta un valor menor, en parte debido a la gran cantidad de estudiantes analizados.

2.3 Arboles de decisión para predecir la aprobación académica de estudiantes

Josip Mesarić y Dario Šebalj crearon un modelo que permitiera la clasificación de estudiantes en una de dos categorías, dependiendo del desempeño que tuvieran al final del curso académico, además de identificar qué factores influían su desempeño. Este modelo se basa en la información extraída de los estudiantes después de que completaran el primer curso académico y la recolección de datos a partir de estudiantes que cursaban su segundo año académico.

Se compararon diferentes algoritmos para la construcción de los árboles de decisión, y a partir de un modelo estadístico se determinó el algoritmo que tenía una mayor precisión, que en este caso fue usando REPTree que tuvo una tasa de clasificación del 79%, pero este árbol no tuvo tanto éxito clasificando ambas clases, por lo que se hizo un promedio de dos modelos, de esta forma se aumentó la tasa de clasificación apoyándose del modelo de algoritmo J48.

2.4 Predecir el rendimiento de los estudiantes usando algoritmo de árboles de clasificación y regresión

.El rendimiento académico de los estudiantes siempre va a ser una preocupación de las partes que están interesadas en la educación y en el desarrollo de las personas, por lo que se busca innovar y establecer métodos que permitan a la persona mejorar más en dichos ámbitos. Teniendo como base toda la información recolectada a partir de los datos que se almacenan en los sistemas de registro, de esta forma se identifican patrones de comportamiento

En este estudio se utilizó un algoritmo de clasificación de minería de datos CART, para analizar los datos de la actividad de los estudiantes y predecir su desempeño académico.

El modelo CART clasificó correctamente 167 estudiantes que reprobaron el curso, pero clasificaron erróneamente a otros 3 que no aprobaron la clase (clasificó correctamente 98,2% de los casos). El modelo también clasificó correctamente 182 estudiantes que no fueron reprobados (clasificó correctamente 100.0% de los casos). La precisión general de la clasificación es, por tanto, el promedio ponderado de estos dos valores (99,1%).

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 gradudados 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 wEafit/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 Algoritmo ID3

El algoritmo ID3, inventado por Ross Quinlan, se basa en la búsqueda de hipótesis o reglas dado un conjunto de ejemplos, el cual deberá estar conformado por una serie de tuplas de valores, cada uno de ellos denominados atributos, en el que uno de ellos (de tipo binario) será el atributo por clasificar. Los elementos de este algoritmo son los nodos que contienen los atributos, arcos que contienen valores posibles del nodo padre y hojas que clasifican el ejemplo como positivo o negativo.[5]

ID3 comienza con el conjunto original como el nodo raíz. En cada iteración del algoritmo, se itera a través de todos sin usar el atributo del conjunto y calcula la entropía de ese atributo. A continuación, selecciona el atributo que tiene la entropía más pequeña para posteriormente dividir el conjunto entre este atributo y dar lugar a los subconjuntos de datos.[6]

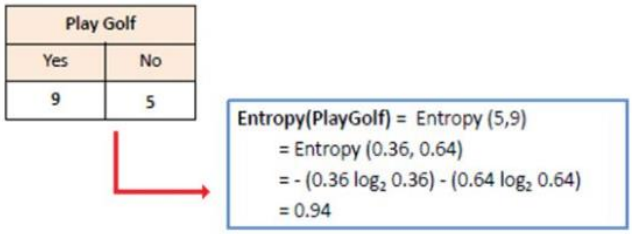

Figura 1. Ejemplo de algoritmo ID3

3.2.2 Algoritmo C4.5

C4.5 es un algoritmo usado para generar un árbol de decisión. Fue desarrollado por Ross Quinlan en 1993 y es una extensión del algoritmo ID3, tiene como ventajas el manejo de datos perdidos y la posibilidad de trabajar con datos continuos.

El algoritmo considera todas las pruebas posibles que pueden dividir el conjunto de datos y selecciona la prueba con la mayor ganancia de información. Para cada atributo discreto, se considera una prueba con n resultados, siendo n el número de valores posibles que puede tomar el atributo. Para cada atributo continuo, se realiza una prueba binaria. En cada nodo, el sistema debe decidir qué prueba escoge para dividir los datos. [7]

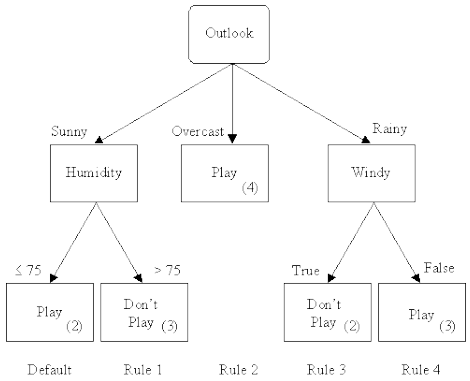

Figura 2. Ejemplo de algoritmo C4.5

3.2.3 Algoritmo CART

El algoritmo CART fue diseñado por Breiman, con este algoritmo, se generan árboles de decisión binarios, lo que quiere decir que cada nodo se divide en exactamente dos ramas.

Este modelo admite variables de entrada y de salida nominal, ordinal y continua, por lo que se pueden resolver tanto problemas de clasificación como de regresión.

Se basa en la idea de impureza. CART selecciona el corte que conduce al mayor decrecimiento de la impureza. Así se consiguen descendientes homogéneos en la variable respuesta Y.

CART propone segmentar la base de datos hasta obtener una estructura de árbol lo más compleja posible. Un nodo se declara terminal sólo si su tamaño es inferior a un umbral preestablecido (normalmente muy pequeño). La complejidad de un árbol se mide por el número de nodos terminales. A continuación, se poda la estructura de árbol maximal que se ha obtenido. Una rama del nodo t de un árbol T está formada por él y todos sus descendientes. Podar la rama en t consiste en eliminar todos los descendientes del nodo t.

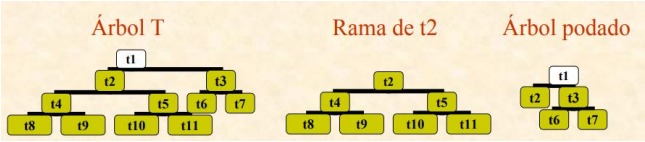

Figura 3. Representación proceso CART

El proceso de poda se apoya en la siguiente medida: Combina el riesgo o coste de predicción y la complejidad. El primer sumando mide el riesgo de T (tasa de error si el problema es de clasificación o la suma de las varianzas residuales si es de regresión). El segundo sumando penaliza las estructuras de árbol complejas. El parámetro $\alpha \geq 0$ se denomina parámetro de complejidad

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}|$$

Figura 4. Medida para la poda maximal

Se realiza de una manera inteligente, eliminando las ramas más débiles. La idea es encontrar subárboles que minimicen Rα (T).

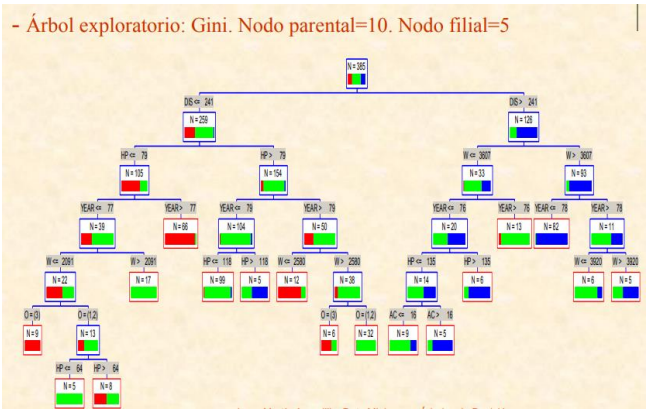

Figura 5. Ejemplo consumo vehículos con CART

3.2.4 Algoritmo CHAID

Procede del ámbito de la Inteligencia artificial. Desarrollado por Kass a principios de los años 80. CHAID considera todos los cortes posibles en todas las variables. Selecciona el corte que da el menor p-valor asociado a una medida de contraste estadístico. Si la variable criterio es categórica la medida es la χ 2 de Pearson. Si es continua la medida es la de la prueba de la F. La búsqueda de la variable y el corte óptimo se lleva a cabo en dos fases: merge (fusión de categorías) y 4plit (selección de la variable de corte).

Fase merge; Agrupa estados o valores de las variables explicativas. Para cada variable, agrupará los estados de cuya unión se obtenga el de menor significación estadística del contraste; siempre que ésta supere un umbral αmerge, fijado de antemano

Fase Split: De la fase merge se toma la agrupación en la variable con contraste más significativo (menor p-valor ajustado). Si la significación estadística es inferior a un mínimo 4plit prefijado, se toma dicha agrupación como partición del nodo.

El criterio de parada CHAID: Se fija de antemano por el experimentador. Depende de: El nivel Split, número de niveles de la estructura de árbol, un umbral mínimo para el tamaño de los nodos descendientes.

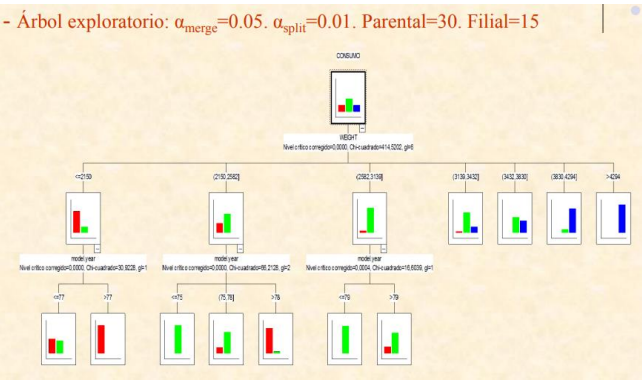

Figura 6. Ejemplo consumo vehículos con CHAID

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]. (github.com/scasta31/ST0245-003/proyecto).

4.1 Estructura de los datos

¿Qué es y cómo está constituido un árbol de decisión?

Un árbol de decisión es un modelo de predicción en el cual dado un conjunto de datos se crean diagramas de construcciones lógicas basados en reglas que sirven para representar y categorizar una serie de condiciones que ocurren de manera sucesiva para la solución de un problema.

Este tipo de diagramas está formado por los siguientes tipos de elementos:

- Nodo Raíz. Este nodo de nivel superior representa el objetivo final o la gran decisión que se está intentando tomar.
- Las ramas, que provienen de la raíz, representan diferentes opciones o cursos de acción que están disponibles al tomar una decisión en particular.
- Nodo de hoja. Los nodos foliares, que están unidos al final de las ramas, representan posibles resultados para cada acción.

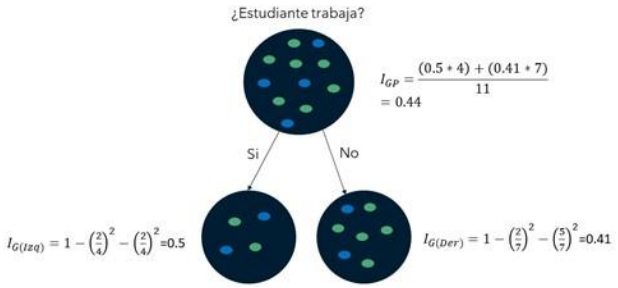

Figura 7: Un árbol de decisión binario para predecir Saber Pro basado en los resultados de Saber 11. Cálculo de la impureza de Gini para el árbol.

4.2 Algoritmos

El algoritmo C4.5 se encarga de calcular la ganancia y el radio de ganancia para cada atributo según la siguiente fórmula, siendo A el atributo en cuestión:

$$Gain(A) = \sum -P(i) \cdot \log_2 P(i)$$

$$= -P(Yes) \cdot \log_2 P(Yes) - P(No) \cdot \log_2 P(No)$$

$$SplitInfo(A) = -\sum \frac{|Dj|}{|D|} * \log_2 \frac{|Dj|}{|D|}$$

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}$$

Luego, es posible crear una tabla que muestre como es el comportamiento de cada atributo de acuerdo con su ganancia y radio de ganancia calculados:

Attribute	Gain	GainRatio
Wind	0.049	0.049
Outlook	0.246	0.155
Humidity <> 80	0.101	0.107
Temperature <> 83	0.113	0.305

Figura 8: Ejemplo de tabla de ganancia y radio de ganancia para cada atributo para el ejemplo de predicción para jugar Golf según el clima.

Finalmente, se procede a seleccionar el atributo con mayor ganancia (en la mayoría de los casos este atributo corresponde también al valor con mayor radio de ganancia) y este sería el nodo raíz del árbol de decisión en cuestión, generando los posibles eventos para cada decisión y los subniveles siguientes en los que se encontrarían el resto de los atributos y sus posibles eventos.

[1] http://www.github.com/ ????????? /proyecto/

4.2.1 Entrenamiento del modelo

El algoritmo se encarga de asignarle ganancia a cada variable posible de cada atributo, en este caso, los atributos nominales seguirían siendo binarios y los continuos se establecerían rangos con el fin de calcular cada ganancia para cada atributo y encontrar el atributo con mayor ganancia.

Posteriormente, para este atributo se generan los posibles eventos que concluyen en los posibles resultados según cada acción. Luego, se construye el árbol de decisión teniendo como nodo raíz el atributo con mayor ganancia y generando las subdivisiones binarias de acuerdo con los posibles eventos de cada nivel.

El algoritmo se encarga de para cada elemento presente en la base de datos evaluar como sería el árbol de decisión y según las variables de cada atributo de este clasificarlo como exitoso o no exitoso.

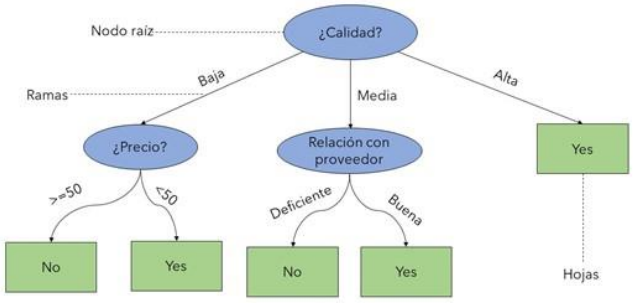

Figura 9: Entrenamiento de un árbol de decisión binario usando *C4.5.* En este ejemplo, mostramos un modelo para predecir si uno debe o no adquirir un material específico en una compañía, dependiendo de la calidad del material, precio y relación con el proveedor.

4.2.2 Algoritmo de prueba

Con el fin de probar el modelo generado a través del algoritmo C4.5 para la predicción del éxito de estudiantes en las pruebas saber Pro, se crea un algoritmo que se encarga de evaluar para cada dato perteneciente a la base de datos para la cual se desee probar el modelo, el comportamiento de cada dato.

Este algoritmo, establecerá a través de una serie de condiciones de if y else, la estructura top-bottom del árbol de acuerdo con el atributo con mayor ganancia, seguido por los posibles desenlaces (ramas) para cada nivel y sus posibles eventos, generando al final de las condiciones una salida binaria que sería la predicción de cada posible estudiante según los atributos anteriormente mencionados.

```
def findDecision(Outlook, Temperature, Humidity, Wind)
   if Temperature&lt;=83:
      if Outlook == 'Rain':
         if Wind == 'Weak':
            return 'Yes'
         elif Wind == 'Strong':
            return 'No'
      elif Outlook == 'Overcast':
         return 'Yes'
      elif Outlook == 'Sunny':
         if Humidity&gt;65:
            if Wind == 'Strong':
               return 'No'
            elif Wind == 'Weak':
               return 'No'
   elif Temperature&gt;83:
      return 'No'
```

Figura 10: Ejemplo de algoritmo de prueba usando *C4.5.* En este ejemplo, mostramos un ejemplo de prueba para predecir si se debe jugar al golf o no, según el clima.

Con este valor de predicción para el grupo de estudiantes analizados, se procedería a comparar el número de estudiantes que se predijo tendrían éxito en la prueba y los que no, con los valores de éxito entregados en la base de datos correspondientes a cada estudiante, estableciendo así, el porcentaje de acierto/error del árbol de decisión creado basado en el algoritmo C4.5.

4.3 Análisis de la complejidad de los algoritmos

El calculo de las complejidades se calculo para cada operación de acuerdo con el peor de los casos que consiste en insertar o buscar un atributo especifico para un estudiante especifico, o básicamente acceder y luego realizar una operación con una posición (i,j) especifica de la matriz de datos. Únicamente se realiza para estas 3 operaciones ya que son las operaciones principales del algoritmo.

Algoritmo (operación)	La complejidad del tiempo
Insertar	O(n*m)
Buscar	O(n*m)
Borrar	O(n)

Tabla 1: Complejidad temporal de los algoritmos de entrenamiento y prueba, siendo n las columnas de la matriz de datos y m las filas.

4.4 Criterios de diseño del algoritmo

El algoritmo se realizó bajo el criterio de acceder a los datos de la manera mas optima posible, por ello al leer los datos se almacenan directamente en un arreglo de arreglos (matriz), esto ya que esta estructura de datos permite organizar de mejor manera el conjunto y en Python se puede visualizar en el explorador de variables si el algoritmo está almacenando los datos de manera óptima.

Además, organizar los elementos en un arreglo de arreglos permite borrar las columnas de atributos que no se deseen evaluar según los criterios de evaluación y ver la matriz luego de la operación.

En una matriz entonces, basta con saber las posiciones para acceder a los datos y, dado que en Python una lista es un vector unidimensional, una lista de listas es un arreglo multidimensional. Por tanto, almacenar los datos en esta estructura de datos resulta más conveniente que almacenarlos por ejemplo en una pila, en la que tendríamos que iterar elemento por elemento en varias pilas para acceder a un elemento específico con un atributo específico.

5. RESULTADOS

5.1 Evaluación del modelo

En esta sección, presentamos algunas métricas para evaluar el modelo. La precisión es la relación entre el número de predicciones correctas y el número total de datos de entrada. Precisión. Es la proporción de estudiantes exitosos identificados correctamente por el modelo y estudiantes exitosos identificados por el modelo. Por último, Sensibilidad es la proporción de estudiantes exitosos identificados correctamente por el modelo y estudiantes exitosos en el conjunto de datos.

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 2.

	Conjunto de datos 1	***Conjunto de datos 2***	***Conjunto de datos 3***
Exactitud	0.87	0.82	0.78

Tabla 2. 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 3.

	Conjunto de datos 1	***Conjunto de datos 2***	***Conjunto de datos 3***
Exactitud	1	0.87	0.78

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

5.2 Tiempos de ejecución

Calcular el tiempo de ejecución de cada conjunto de datos en Github. Medir el tiempo de ejecución 100 veces, para cada conjunto de datos, e informar del tiempo medio de ejecución para cada conjunto de datos.

	Conjunto 1		Conjunto 2		Conjunto 3	
Operación	Test	Train	Test	Train	Test	Train
Read	6.15	45.37	21.62	84.73	76.03	181.25
Search	0.09	0.47	0.51	1.32	1.51	5.03
Insert	0.09	0.48	0.49	1.38	1.47	4.39

Tabla 4: Tiempo en segundos de ejecución del algoritmo para diferentes conjuntos de datos.

	Conjunto 1		Conjunto 2		Conjunto 3	
	Test	Train	Test	Train	Test	Train
Cantidad datos	5000	15000	15000	45000	45000	135000

Tabla 5: Cantidad de datos por conjunto.

5.3 Consumo de memoria

Presentamos el consumo de memoria del árbol de decisión binario, para diferentes conjuntos de datos, en la Tabla 7. Cabe resaltar que para conocer el espacio utilizado en memoria del dataset en la RAM se utilizó la librería sys, esta permite conocer cuál es el total de recursos que se destina a una variable específica.

	Conjunto 1		Conjunto 2		Conjunto 3	
Tipo	Test	Train	Test	Train	Test	Train
Consumo en MB	21,52	62,46	62,46	180,64	180,64	587,02

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

5.4 Complejidad en el tiempo

Presentamos el consumo en tiempo de manera gráfica, se presenta el tiempo de ejecución del algoritmo en segundos para cierta cantidad n de datos.

Figura 11: Complejidad en el tiempo.

6. DISCUSIÓN DE LOS RESULTADOS

Los resultados obtenidos en términos de exactitud son apropiados dependiendo del conjunto de datos a analizar y su tamaño, sin embargo, para conjuntos grandes de 45000 estudiantes, por ejemplo, se consiguen exactitudes cercanas al 80%, es decir, para 4 de cada 5 estudiantes se consigue generar predicciones correctas, siendo este valor una buena cifra dependiendo de la aplicación que se busque para el algoritmo y la precisión que se requiera.

En cuanto al sobreajuste del modelo se buscó evaluar variables generales que permiten realizar predicciones valederas sobre el desempeño futuro del estudiante, buscando entonces que el modelo se entrenara y lograra aprender conceptos generales sobre el conjunto de datos que está estudiando con el fin de poder aplicarlo a cualquier conjunto y obtener resultados óptimos en términos de exactitud.

En cuanto al consumo de tiempo, es importante resaltar que el algoritmo tarda mucho en ejecutarse y además tiende a crecer de manera exponencial lo cual no es recomendable cuando se busca obtener algoritmos rápidos o que se estabilicen en el tiempo luego de una cantidad n de datos evaluados. Por ende, se concluye que el consumo de tiempo no es óptimo.

Finalmente, el algoritmo puede aplicarse para dar becas o ayudar a estudiantes con baja probabilidad al éxito ya que permite no solo predecir sino establecer parámetros que permiten estudiar los atributos o variables que impactan en mayor magnitud el desempeño de los estudiantes.

6.1 Trabajos futuros

En el futuro, la implementación del algoritmo se buscaría hacerse con una complejidad menor, lo que permita evaluar grandes cantidades de datos en menor tiempo, además de optimizar el consumo de memoria mediante alternativas para la lectura y carga de datos. Sin embargo, como primera versión del algoritmo, presenta buenos resultados y una exactitud apropiada.

Usar un bosque aleatorio podría ser una solución que permitiría correr de manera eficiente bases de datos más grandes y manejar adecuadamente datos perdidos, los cuales son muy comunes en bases de datos similares a las trabajadas en este trabajo. Quizás, otra solución podría ser implementar algunas librerías especializadas en el tema de bases de datos en Python, permitiendo así reducir los tiempos de algunas operaciones y el del algoritmo en general.

AGRADECIMIENTOS

Esta investigación se realizó bajo la supervisión y el apoyo del profesor de la asignatura y los monitores de esta, gracias a ello se pudo aplicar los conocimientos aprendidos durante el semestre y poner a prueba estos bajo el reto de entender el tema de árboles de decisión y la construcción de un algoritmo práctico para la predicción del desempeño de los estudiantes en las pruebas Saber Pro según algunos atributos conocidos previamente.

REFERENCIAS

1. Pojon, M. Using Machine Learning to Predict Student Performance. Retrieved August 14, 2020, from University of Tampere: https://trepo.tuni.fi/bitstream/handle/10024/101646/GRADU-1498472565.pdf?sequence=1.
2. Kabakchieva, D. Predicting Student Performance by Using Data Mining Methods for Classification, *Sciendo*. Retrieved August 14, 2020, from Versita, Bulgarian Academy of Sciences: https://content.sciendo.com/view/journals/cait/13/1/article-p61.xml
3. Anónimo. ¿Qué es el algoritmo ID3. Retrieved August 14, 2020, from Quora: https://es.quora.com/Qu%C3%A9-es-el-algoritmo-ID3.
4. Wikipedia. Algoritmo ID3 - ID3 algorithm. Retrieved August 14, 2020, from Qwe: https://es.qwe.wiki/wiki/ID3_algorithm
5. Blogspot. Minería de Datos - C4.5. Retrieved August 14, 2020, from Blogspot: https://mineriac45.blogspot.com/
6. https://web.fdi.ucm.es/posgrado/conferencias/JorgeMartin-slides.pdf CART y CHAID
7. https://bookdown.org/content/2274/agrupacion-de-la-informacion.html CART
8. https://en.wikipedia.org/wiki/Chi-square_automatic_interaction_detection CHAID
9. Árbol de decisiones: ejemplos de ventajas y pasos a seguir. Retrieved October 11, 2020, from https://retos-directivos.eae.es/arbol-de-decisiones-ejemplos-de-ventajas-y-pasos-a-seguir/.

10. Árbol de decisiones: ejemplos de ventajas y pasos a seguir. Retrieved Febrero 24, 2020, from Retos Directivos: https://retos-directivos.eae.es/arbol-de-decisiones-ejemplos-de-ventajas-y-pasos-a-seguir/

13. Arboles de decision y Random Forest. Retrieved Noviembre 16, 2018, from Johanna Orellana Alvear: https://bookdown.org/content/2031/ensambladores-random-forest-parte-i.html